Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Karmen Dykstra

Predicting Protein Producibility:

Binary classification of recombinant proteins produced in filamentous fungi

Master's Thesis
Espoo, January 7, 2016

Supervisor: Prof. Juho Rousu
Advisor: Mikko Arvas, Ph.D.

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author:	Karmen Dykstra		
Title:	Predicting Protein Producibility: Binary classification of recombinant proteins produced in filamentous fungi		
Date:	January 7, 2016	**Pages:**	78
Major:	Machine Learning and Data Mining	**Code:**	T-61
Supervisor:	Prof. Juho Rousu		
Advisor:	Mikko Arvas, Ph.D.		

Recombinant protein synthesis aims to produce specific protein products of interest in living cells. However, protein production is subject to failure, and thus the successful development of a computational tool to predict protein sequence success prior to laboratory experimentation would save time and resources. We demonstrate the ability of an SVM trained on protein amino acid composition to predict successful protein production in a dataset of sequences tested in the host species *Trichoderma reesei*. We found that predictive models generalize well between two species of filamentous fungi, and furthermore that 50 training sequences are sufficient to train a model that yields an AUC of over .7. We introduced novel predictive features using protein domains detected with the InterProScan tool, which were modestly successful in the predictive task but whose addition did not improve over the use of amino acid composition alone. Experiments applying semi-supervised SVM formulations to the predictive task did not yield significant improvement, most likely because the spatial distribution of data points under the chosen numeric representations did not conform to the assumptions of the semi-supervised models. We explored the species of origin and enzyme function of sequences from the UniProt SwissProt database predicted to be successful by the trained SVM models, and showed that models trained with an RBF kernel were the most conservative in terms of the number of predicted successes.

Keywords:	binary classification, SVM, protein, filamentous fungi, semi-supervised
Language:	English

Acknowledgements

Thank you to my supervisors, Mikko Arvas and Juho Rousu, for your guidance and patience while completing this thesis.

Thank you to my sister and her family, Ursa, Juki, and Maria, for all the good food and always cheering me up.

And thank you to Michael. *You're* the best.

Espoo, January 7, 2016

Karmen Dykstra

Contents

Chapter 1

Introduction

Proteins are ubiquitous biological molecules that perform many diverse functions - typically around 60% of the organic matter in a cell consists of proteins [11]. Recombinant protein synthesis, the technical word for the artifical production of specific proteins using genetic engineering, has significant applications in a wide range of industries. For example, a major breakthrough in the treatment of diabetes was made when protein synthesis techniques were used to artificially produce the human insulin protein in *Escherichia coli* in 1978 [20]. Of the many applications of recombinant protein synthesis, example products relevant to modern research include antibodies to treat various diseases [7] and enzymes for use in biofuel production [40].

Recombinant protein synthesis involves cloning a gene from a donor species and inserting it into the host species, where, if successful, protein synthesis will take place and result in the fully functional form of the protein. When researchers seek to produce a protein, they must first select the genetic sequence to use. A number of issues can occur during the synthesis of recombinant proteins and hence it is often unknown, prior to lab experiments, whether or not a given sequence will result in a fully functional protein in the host organism. Since protein production is time-consuming and costly, it is desirable to develop a computational tool that would screen the candidate genes prior to lab experimentation. Such a tool would take candidate sequences as input and indicate which of the genes would be successful or unsuccessful in the specified host. The successful development of this tool would save researchers time and resources by increasing the success rate of protein production experiments.

Researchers have expended significant effort to develop such tools for predicting successful protein production in the bacterial host *Escherichia coli*. However, little work has been done to predict protein production in other host organisms, including filamentous fungi hosts such as *Aspergillus niger* and *Trichoderma reesei*. Filamentous fungi, commonly known as moulds, are of interest as production hosts

for a number of reasons including their inexpensive growth [57] and capacity to produce and secrete proteins in large quantities, as indicated by the production of some endogeneous proteins in amounts as high as 100 g/L [47]. Also, unlike *E. coli*, they have the necessesary components for glycosylation [15], a biochemical process required for e.g. producing human antibodies.

Recently van den Berg et al. introduced a publicly available dataset of over 300 *A. niger* sequences, and used these to train the first computational tool to successfully predict protein production in a filamentous fungi host [55]. We investigate this tool in relation to a smaller dataset of almost 50 proteins produced in *T. reesei*, as made available for this thesis by the VTT Technical Research Centre of Finland.

1.1 Problem statement

Our primary goal is to develop an effective computational tool that uses amino acid sequences as input to predict the successful production of proteins in *Trichoderma reesei*. Related to this goal, we address the following questions: How do existing computational tools perform on the *T. reesei* dataset? Can we improve the predictor by enriching the protein representation, or by making use of unlabeled sequences? How many labeled protein sequences are necessary to train an effective model? What kind of proteins does the model predict as "successful"?

1.2 Thesis structure

Chapter 2 gives an overview of the biology relevant to the thesis topic, including the basics of protein composition and synthesis, recombinant protein production, and associated challenges. Chapter 3 continues with a summary of possible modeling approaches to address issues in recombinant protein production, a brief introduction to the machine learning methods used in our experiments, and the definition of evaluation metrics. Chapter 4 introduces the datasets used in the experiments, our numeric representations of the sequences, and a detailed description of the experiments. Chapter 5 gives results and discussion separated by experiment, and finally Chapter 6 contains concluding remarks.

Chapter 2

Biological Background

Proteins are biological molecules present in all living cells. In fact, about two thirds of the organic matter in a typical cell consists of proteins [11]. Proteins are directly responsible for most metabolic processes, and also serve transport, structural, and regulatory functions. Because proteins are useful in a wide range of applications, researchers in biotechnology have developed methods for cost-effective, targeted protein production through the use of recombinant DNA technology. This engineered synthesis of protein products in microbial cell systems is in contrast to previous protein isolation methods, in which kilograms of plant and animal tissue were necessary to isolate small quantities of protein product [46]. In filamentous fungi, recombinant protein methods has been applied to produce a range of fungal proteins useful for food, feed, textiles, and leather treatment [60], and *T. reesei* has been shown to be a promising host for pharmaceutical production [34].

This section gives a brief overview of the core processes and challenges involved in recombinant protein production. Many details are omitted for brevity, but are covered thoroughly in most modern molecular biology textbooks (e.g. [21]). The information in Sections 2.1 and 2.2 was drawn from [11] and [35], and Sections 2.3 and 2.4 were written by referencing [21]. We start by reviewing the basics of protein composition and synthesis.

2.1 Protein composition

All proteins contain one or more polypeptide chains, which are sequences of amino acid units joined by peptide bonds. The number of amino acids in a protein can range from about 20 [53] to over 1,000 [21]. An amino acid is a biological molecule that contains a core carbon atom linked to amine (-NH_2) and carboxylic acid (-COOH) functional groups, a hydrogen atom, and an additional side-chain, called an "R-group," which varies between amino acids in size, structure, and electric

Amino Acid	3-Letter Code	1-Letter Code	Physical Properties
Alanine	Ala	A	hydrophobic
Arginine	Arg	R	basic
Asparagine	Asn	N	neutral polar
Aspartic acid	Asp	D	acidic
Cysteine	Cys	C	hydrophobic
Glutamic acid	Glu	E	acidic
Glutamine	Gln	Q	neutral polar
Glycine	Gly	G	—
Histidine	His	H	basic
Isoleucine	Ile	I	hydrophobic
Leucine	Leu	L	hydrophobic
Lysine	Lys	K	basic
Methionine	Met	M	hydrophobic
Phenylalanine	Phe	F	hydrophobic
Proline	Pro	P	hydrophobic
Serine	Ser	S	neutral polar
Threonine	Thr	T	neutral polar
Tryptophan	Trp	W	hydrophobic
Tyrosine	Tyr	Y	neutral polar/ hydrophobic
Valine	Val	V	hydrophobic
Aspartic acid or Asparagine	Asx	B	
Glutamic acid or Glutamine	Glx	Z	
Unspecified amino acid		X	

Figure 2.1: Amino acid names, abbreviations and physical properties. Copied from [11].

charge. An exception is proline, which has -NH attached to its core carbon instead of -NH_2. Amino acids can be categorized based on their side chain as basic, acidic, hydrophobic, or polar and uncharged. There are 20 amino acids ubiquitously found in protein sequences prior to post-synthesis modification. The single letter code, three letter code, full name, and R-group classification for each of the 20 amino acids are given in Figure 2.1.

Two or more amino acids link together to form a protein via peptide bonds, covalent bonds formed by the removal of water from the carboxyl group (-COOH) of one amino acid and the amino group (-NH_2) of the other. Thus the first amino acid in a synthesized polypeptide chain retains its free amino group and is called the N-terminus of the chain. The last amino acid, left with a free carboxyl group, is called the C-terminus.

2.2 Protein synthesis

The sequence of nucleotides in long, double-stranded deoxyribonucleic acid (DNA), coiled into a structure called a chromosome, determines the sequence of amino acids

in synthesized proteins. In a process called transcription, ribonucleic acid (RNA) polymerase sequentially joins ribonucleotides together according to the sequence information from a particular portion of DNA, to produce a single strand of messenger RNA (mRNA). A DNA sequence that codes for a strand of mRNA is called a gene, and in most cases genes code for proteins. Short nucleotide sequences in the DNA, called promoter and terminator regions, indicate where the transcription process begins and where it ends. To initiate transcription, RNA polymerase binds to a promoter region of DNA that occurs upstream from the sequence copied to mRNA. The transcriptional terminator, a sequence downstream from the area for mRNA synthesis, defines where RNA polymerase stops transcription.

In prokaryotic organisms, cells do not contain the cellular nucleus or separate cell compartments, which are features of the cells of eukaryotic organisms. Bacteria such as *E. coli* are prokaryotic, and plants, mammals, and fungi are eukaryotic. In prokaryotes, each strand of mRNA encodes for the entire target protein. In eukaryotes, sequences of coding DNA, called exons, are interrupted by sequences of non-coding DNA, called introns, which contain sequence information excluded from the final protein sequence. Thus, once the entire gene is transcribed by RNA polymerase, in eukaryotic cells the introns are removed and the exons joined together for the final mRNA strand used to make the protein chain.

A sequence of three nucleotides in the mRNA, called a codon, codes for one amino acid in the protein sequence. RNA contains four different nucleotides, so there are $4^3 = 64$ possible codons. Since there are 64 codons and 20 amino acids, multiple codons represent the same amino acid. Due to the 3:1 ratio of nucleotides to amino acids, different start locations in reading a gene will yield different amino acid sequences. Consider the sequence GAAAUGUAU. If we read starting at the first nucleotide, we get the sequence GAA | AUG | UAU, which gives the amino acid sequence Glu | Met | Tyr. If we read from the second nucleotide, we get G | AAA | UGU | AU which yields – | Lys | Cys | –. Finally starting at the third, we get GA | AAU | GUA | U which yields – | Asn | Val | – . Each different way of reading the gene is called a reading frame. Because nucleotides code for amino acids in triplets, there are three possible reading frames. An open reading frame (ORF) is a sequence of 50 or more codons that occurs without a "stop codon," which triggers the end of translation. In the recombinant protein literature a gene that encodes for a protein of interest is often referred to as an ORF.

In the translation process, transfer RNAs (tRNAs), ribosomes, and a large number of protein factors join together amino acids to form the protein sequence according to the information in the mRNA. The process is initiated by the binding of a small and large ribosomal subunit to the 5' end of the mRNA, named for the 5' phosphate group on that end. Amino acids to form the protein are brought by tRNAs, where each tRNA is structured so that it carries a single type of amino

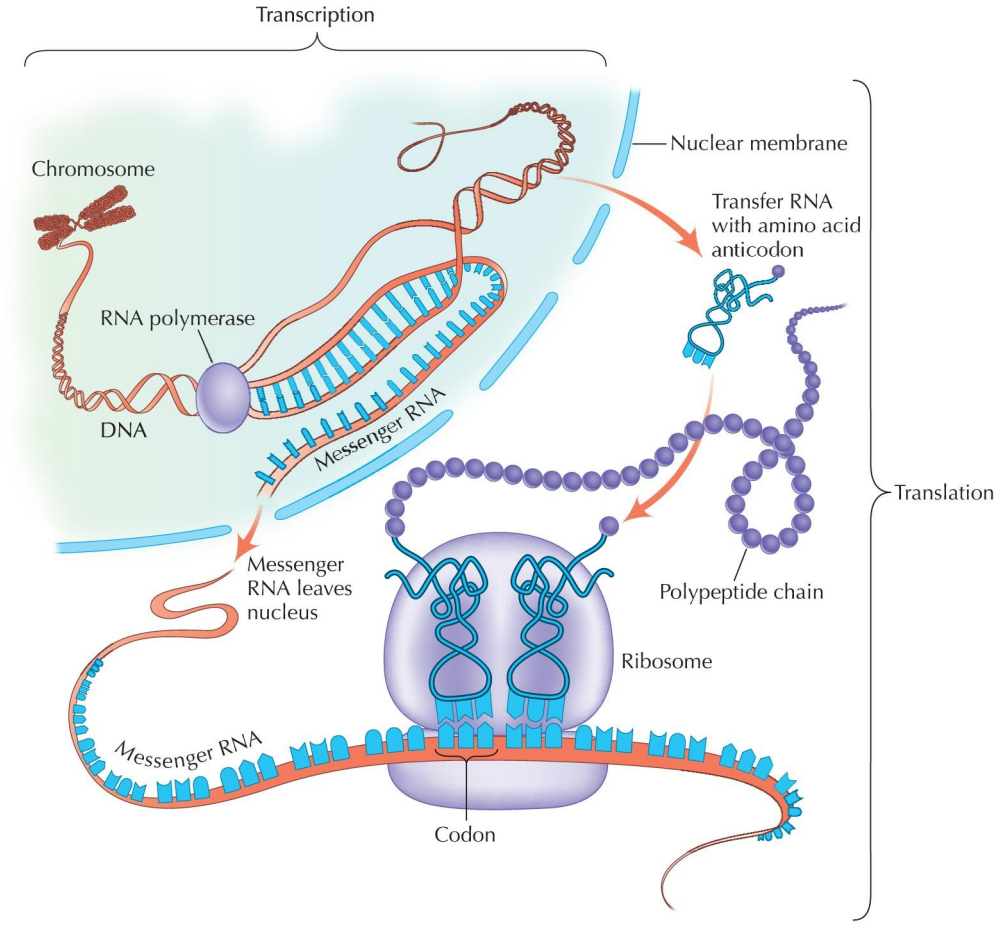


Figure 2.2: The transcription and translation processes of protein synthesis in eukaryotic organisms. Copied from [5].

acid. The ribosome catalyzes the formation of peptide bonds between the amino acids transported by tRNA. The growth of the amino acid chain continues until a stop codon is encountered in the mRNA. A single mRNA chain can be translated by a number of ribosomes simultaneously.

Proteins undergo a number of modifications upon completion of translation. The methionine on the N terminus is cleaved off in most proteins, and in eukaryotes the protein may be cleaved to make several discrete proteins with different functions. Phosphate groups, lipids, carbohydrates, or other groups may be added for certain protein functionalities.

Once synthesized, the polypeptide chains in a protein must be folded into their correct structure for the protein to achieve full functionality. Four structural levels are recognized in proteins: primary structure, the linear sequence of amino acids; secondary structure, the folding or coiling of the original structure by hydrogen bonds; tertiary structure, additional folding of the secondary-structure polypeptide chain to form its final 3D shape; and quaternary structure, the possible joining of several polypeptide chains to give the final structure of the protein. Some proteins only have one polypeptide chain and thus do not have a quaternary structure.

2.3 Protein secretion

Filamentous fungi are appealing as industrial host species because of their natural ability to secrete large quantities of proteins into the extracellular media, which makes the protein product easier to extract than if it remained within the cell walls. Since our datasets of interest are from sequences produced in filamentous fungi, this thesis focuses on secreted proteins.

Both bacteria and eukaryotic cells have specialized machinery for protein secretion. Proteins destined for secretion are tagged with a signal sequence at the N-terminus, which consists of about 20 amino acids. The signal sequence is cut off after export and hence is not present in the mature protein. Although there is little homology between signal sequences in different secreted proteins, the signal sequence has a general pattern of two to eight positively charged amino acids, then a long stretch of hydrophobic amino acids, followed by an amino acid with a short side chain just before the cleavage site.

Proteins targeted for secretion undergo a specific sequence of steps, many of which are general to eukaryotic cells. The following overview is taken from [47]. First the signal sequence at the N-terminus of secreted proteins is recognized by the signal recognition particle (SRP) and translation is paused. The ribosome-mRNA-peptide complex is transported to the membrane of the endoplasmic reticulum (ER), where the chain is simultaneously translated and translocated into the ER lumen, the space enclosed by the ER membrane. Upon translation, protein folding

takes place, usually with the help of various chaperones and foldases. In some cases, in a process called glycosylation, a carbohydrate side chain is attached to the protein by the oligosaccharyltransferase complex in the ER membrane. If the proteins are misfolded or aggregated, they are removed from the ER and degraded by the ER-associated protein degradation (ERAD) system. If folded correctly, the proteins undergo transport in membrane vesicles with specific protein coats, first to the Golgi apparatus and then to the cell membrane.

2.4 Recombinant protein production

The goal of recombinant protein production is to produce a protein in a host species by synthetically incorporating genetic information from a species of origin. If the host and originator are the same species, the protein is called homologous, and if they are different, the protein is called heterologous. Recombinant protein production requires the gene that codes for the target protein product and an autonomous piece of DNA, called a vector, that introduces the gene into the host species. Once the gene coding for the target protein is successfully inserted into the vector, creating a "clone," the clone is inserted into the host cells. Cells that lack the clones producing the target protein are separated from successfully transformed cells, from which the final protein product is harvested after an appropriate incubation period. Intracellular proteins can be gathered by breaking open or "lysing" the cell walls, or secreted proteins can be gathered directly from the growth media.

The most common form of vector is a self-replicating ring of DNA called a plasmid, originally found in bacteria. The typical plasmid contains an origin of replication that controls plasmid proliferation, a promoter under the control of a regulatory sequence either contained in the plasmid or in the host species, and a gene for antibiotic resistance that facilitates the selection of successfully transformed cells. Many vectors have been developed, plasmid and otherwise, and in modern biotechnology the experimentalist chooses from a range of commercially available options [46].

The promoter to use in the vector should be strong, so that it induces a high level of transcription of the target gene, but should not be uncontrolled so that production of the protein product exhausts the host cell. The increased metabolic burden of producing the target protein can already favor the proliferation of cells that do not produce the synthetic protein, resulting in transformed cells being overtaken by those that are non-producing. Several well-suited promoters have been discovered and are routinely used in recombinant protein production. Common choices include the T7 promoter in *E. coli* [46], the cellobiohydrolase I promoter in *T. reesei* and the glucoamylase A promoter in *A. niger* [60].

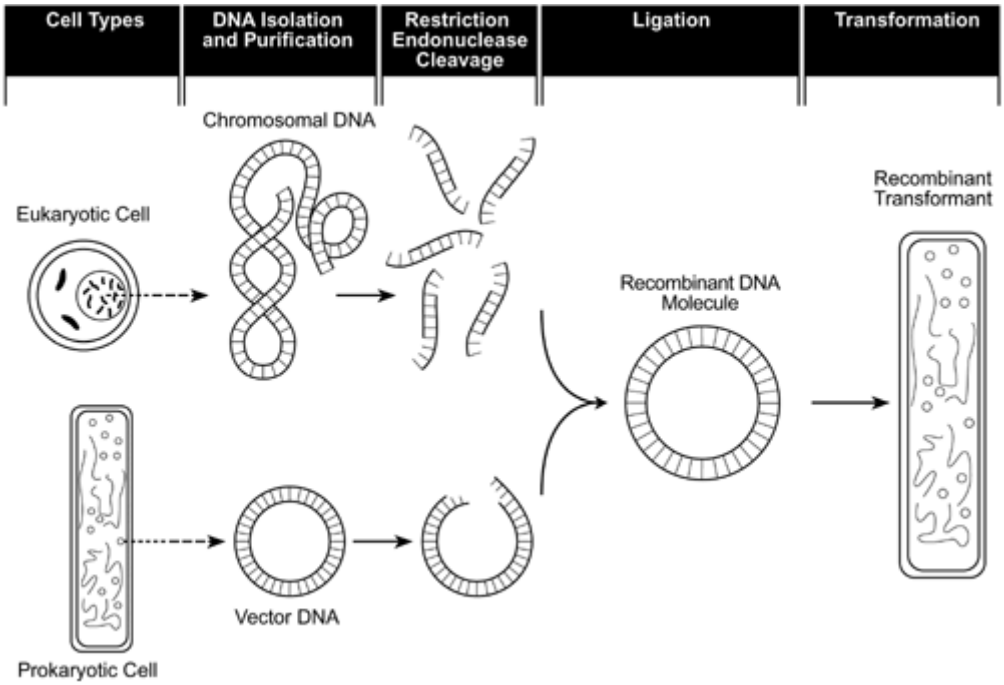


Figure 2.3: Illustration of the basic steps of applying recombinant DNA technology to incorporate a gene from a eukaryotic organism into a prokaryotic host. Copied from [25].

The gene coding for the target protein historically had to be isolated from the donor organism by the use of restriction enzymes. Now, however, efficient methods exist for direct synthesis of a DNA sequence, which can then be "amplified" by Polymerase Chain Reaction (PCR) to create many copies of the gene for insertion into vectors. Restriction endonucleases are then used to cut the vector and gene at specific nucleotide sequences, and DNA ligase to join the gene and vector together.

In *E. coli*, clones are inducted into the host organism by electroporation, in which a short pulse of electricity is administered to cells suspended in media. The brief shock creates transient pores through which the vectors are taken into the cells. Since the desired uptake of the target DNA does not occur in every case, vectors are cleverly engineered so that successfully transformed cells can be selected by testing for antibiotic resistance or observing cell color.

Trichoderma reesei and *Aspergillus niger*, the host species of interest in this thesis, are both filamentous fungi. Unlike *E. coli* and yeasts, which are unicellular, filamentous fungi are multicellular organisms that grow in long, branching strands called hyphae. The process of protein synthesis is different in prokaryotes and eukaryotes, and there are reasons for favoring a eukaryotic host for some proteins. For example, proteins produced by higher eukaryotic organisms, like humans, might require O- or N-linked glycosylation to function correctly, the machinery for which does not exist in *E. coli* [42].

In typical filamentous fungi protein production experiments, clone creation and selection is first carried out in *E. coli*. Clones are then filtered from successfully transformed colonies and inducted into the host species. Plasmids that self-replicate in the final filamentous fungi host are not used [60], but rather the vectors integrate into the host chromosome. DNA uptake in filamentous fungi is accomplished through electroporation, transfer of the DNA by *Agrobacterium*, removal of the walls of the host cells, or biolistic methods where the DNA is shot into the cell on tungsten or gold particles.

2.5 Challenges in recombinant protein production

The synthesis of recombinant proteins can fail for a variety of reasons at multiple steps in protein synthesis, and in the case of no observable protein product in the media, it is challenging to identify *where* the production process failed.

It is possible that the target gene is not transcribed, or transcribed in very small quantities. This can be caused by improper environmental conditions. For example, mRNA for one of the major cellulases produced by *Trichoderma reesei*, CBHI, is expressed in quantities thousands of times greater in media containing

cellulose than media containing glucose, which represses cellulase expression [30]. Similarly, differences in pH can cause low or high yields, and experimental choices are complicated by the fact that appropriate environmental conditions for high production vary depending on the protein of interest [60].

Even if transcription proceeds, problems can occur at the translation and folding stages. If the codons in the target gene are significantly different than the host genome, there may be a deficiency of the necessary tRNAs for translation of the protein product [46]. One approach to this issue is to adapt the target gene codon composition to match the codon use of the host organism, but there are additional complications. For example, some codon sequences cause translational pausing, which is beneficial for correct protein folding.

After translation the protein may not be folded correctly, which causes subsequent intracellular degradation, or clumping with other proteins to form unusable aggregates called "inclusion bodies" (IBs) [56], which is especially common and problematic in *E. coli*. While it is possible to denature aggregates and then refold the proteins, the refolding process requires protein-specific conditions and is undesirably costly [41]. One solution for this issue is to fuse the target gene with the sequence for a different protein that facilitates solubility, creating a "fusion protein" [46]. Fusion proteins are also created to prevent the breakdown of proteins by proteases and facilitate purification of the target protein [21], as well as aid successful secretion of the protein product [42].

In eukaryotic organisms, protein translation and folding of secreted proteins in the ER requires a number of chaperones, cochaperones, and cofactors [21]. If the gene is transcribed at high levels, there may be a shortage of the necessary components for proper folding. When unsuccessfully folded proteins accumulate in the ER of filamentous fungi they trigger the unfolded protein response (UPR), which upregulates the transcription of foldases and transport proteins [42]. There have been attempts to increase ER throughput by increasing expression of the UPR regulator gene, *hac*, the results of which vary from having a positive to negligible impact [42]. Similarly, the expression of protein disulfide isomerase, an instrumental enzyme for forming disulfide bonds for protein folding and stability, has been artificially boosted by its inclusion in recombinant vectors. In some cases this resulted in impressive gains in yeast expression systems [21]. While modifications of this kind can improve yields, there is no way to guarantee the correct translation and folding of recombinant proteins.

In addition, protein products are vulnerable to degradation by intracellular and extracellular proteases natively produced by the host organism. This issue has been mainly addressed through strain engineering, wherein the genes for problematic proteases are deleted from the genome of the host organism. For example, the BL21 strain of *E. coli* lacks the gene for the Lon protease that degrades many foreign

proteins, and the gene for OmpT, a protease that acts on extracellular proteins [46]. In addition to deleting protease genes, strain engineering of *T. reesei* has involved deletion of some the major cellulases that compose the bulk of secreted proteins, the removal of which has favored the production and secretion of other proteins [47].

The production of the target protein may hinder the growth and survival of the host cells. In some cases this is caused by the added metabolic burden from producing the recombinant protein, and in other cases the target protein is toxic to the host cell [46]. In addition, since filamentous fungi are multi-cellular organisms, cells that do not successfully integrate the foreign DNA can separate from the integrated cells in heterokaryosis, resulting in a non-producing segment. Due to the production of recombinant proteins imposing additional metabolic burden on producing cells, selection can favor non-producing or low-producing cells [60].

Thus there are numerous complex and multifaceted problems that can occur in recombinant protein production, which hinder the production of active, soluble proteins. Computational methods have been developed to address some of these issues.

2.6 Numeric representation of proteins

Researchers have applied computational methods to predict the presence of protease cleavage sites in protein sequences [51], the intra- or extra-cellular destinations of proteins [32], and protein solubility upon expression in *E. coli* [24]. These applications share a common problem structure with the primary aim of this thesis, the prediction of successful protein production in filamentous fungi. In each case, protein sequences are used as input to train machine learning models, or create scoring schemes, for the prediction of the property of interest in unseen protein sequences. Protein sequences come in the form of strings of characters – for example, the string "MAY" indicates the sequence of methionine, alanine, and tryptophan. However, statistical and machine learning methods require a numeric representation of the protein sequences, which raises the question of how transform the character sequence to numbers in a way that best represents the relevant biological information.

The most direct numeric representation is the replacement of each amino acid with a numeric equivalent. This can be accomplished in a binary encoding of each amino acid by placing a 1 in a vector of 20 zeros. Under this representation the amino acid alanine could be encoded by the vector $[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$, and for a sequence of length N the resulting feature representation is of size $20 \times N$. This representation has been used, for example, for the purpose of predicting protease cleavage sites [45, 58]. However, many machine learning

methods require a fixed-length input vector, and proteins vary dramatically in sequence length. Thus the use of the binary coding scheme and similar direct-replacement representations are well-suited to situations that use a fixed-length subsequence, such as the sliding window methods frequently used in cleavage site prediction. However, they are impractical for prediction problems where the entire sequence is relevant, such as our goal of predicting successful recombinant protein production.

Often a number of physicochemical properties of the protein are used. For example, isoelectric point (pI) is a highly relevant feature frequently included for predicting the solubility of proteins upon expression in *E. coli* [22, 39, 43]. The isoelectric point is the characteristic pH at which a molecule has no net electric charge, and for a protein sequence can be calculated using the Henderson-Hasselbalch equation [35]. There are also direct measurements of hydrophobicity, such as the Goldman, Engelman, Steitz (GES) scale, based on the relative energies involved in interactions of polar and charged amino acid side chains with water [17]. The commonly used grand average of hydropathy (GRAVY) measurement is the sum of hydropathy values of each constituent amino acid, divided by the length of the protein [33]. Additional features include the protein sequence length, molecular weight, and average charge of the constituent amino acids.

Protein sequences are also commonly represented by composition measures. For example, the amino acid composition counts the number of each type of amino acid, and divides each count by the total number of amino acids in the sequence [1]. Protein composition can similarly be calculated for the 400 possible dipeptides, sequences of two amino acids, or the 8,000 possible tripeptides, sequences of three amino acids. It should be noted that calculating the composition of dipeptides and tripeptides is more computationally expensive than single amino acid composition, and does not necessarily improve performance [50]. Sometimes the composition is calculated of each *group* of amino acids, as specified by their type of side chain (Table 2.1). For example, one might calculate the proportion of aromatic or polar amino acids. If specific amino acid subsequences are known to be associated with the predictive problem, their occurrence may also be measured, as with the "reduced alphabet sets" in Idicula-Thomas et al. [29].

Numeric features may be calculated directly from the DNA sequence of the protein. One simple measure, the codon composition, is calculated exactly like the amino acid composition, except that the counted units are codons. The Codon Adaptation Index (CAI), a representation that is unique to the DNA sequence, compares codon occurrence in the genes of interest to codon use in a reference

[1] In fact, analyzing proteins by their amino acid composition has a long and rich history. Already in 1964, Fisher sought correlations between amino acid composition and size based on observations of the tendency of hydrophobic amino acids to fold into the interior of the 3D protein structure [18].

set of genes [48]. For example, Boettner et al. calculated the CAI using a set of highly expressed yeast genes as the reference set to predict the soluble expression of malaria proteins in a yeast host [9].

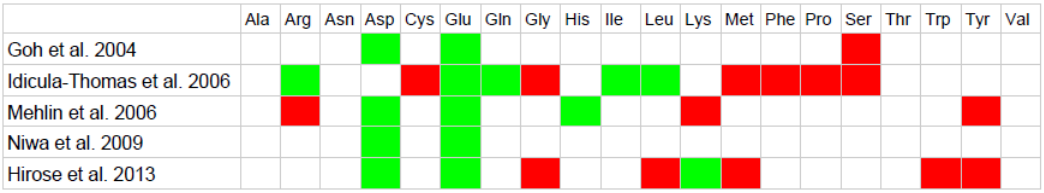


Figure 2.4: Comparison of relationships observed between high content of individual amino acids and protein solubility in *E. coli*, as described by Goh et al. [22], Idicula-Thomas et al. [29], Mehlin et al. [39], Niwa et al. [43] and Hirose et al. [27]. A green box indicates a positive relationship between amino acid content and solubility, a red box indicates a negative relationship, and a white box indicates no observed relationship. Three letter codes for amino acids are given in Figure 2.1.

As of yet, there is no consensus on the best numeric representation to use for protein prediction tasks. Researchers may select a combination of numeric representations, and the chosen measures tend to vary between studies. Even when the same representations are used across studies, conclusions about their predictive contributions do not always agree. For example, Figure 2.4 shows how the observed relationships between individual amino acids and protein solubility in *E. coli* vary across five studies. While higher aspartate and glutamate content are consistently associated with solubility, the roles of lysine and arginine are contentious. Differences in experimental conditions and the type of proteins in question could be responsible for discrepancies between studies [29]. A lack of consensus on numeric protein representation invites the independent exploration of representations for each prediction task.

For guidance as to what numeric representations to use in this thesis, we look to previous work in which van den Berg et al. [54] investigated the efficacy of different protein representations for the filamentous fungi protein prediction problem. 638 recombinant proteins tested for expression and secretion in *A. niger* were used to compare the performance of different classifiers and protein sequence representations to predict protein producibility. From their experiments with the protein features of amino acid composition, amino acid groups, and local and global hydrophobicity measures, they concluded that composition of asparagine, the set {arginine, lysine}, tyrosine, and the presence of hydrophobic peaks were the best-discriminating features of successfully versus unsuccessfully produced proteins.

A subsequent paper addressed the same task, but focused on a thorough comparison of a greater number of sequence-based protein representations [55], the

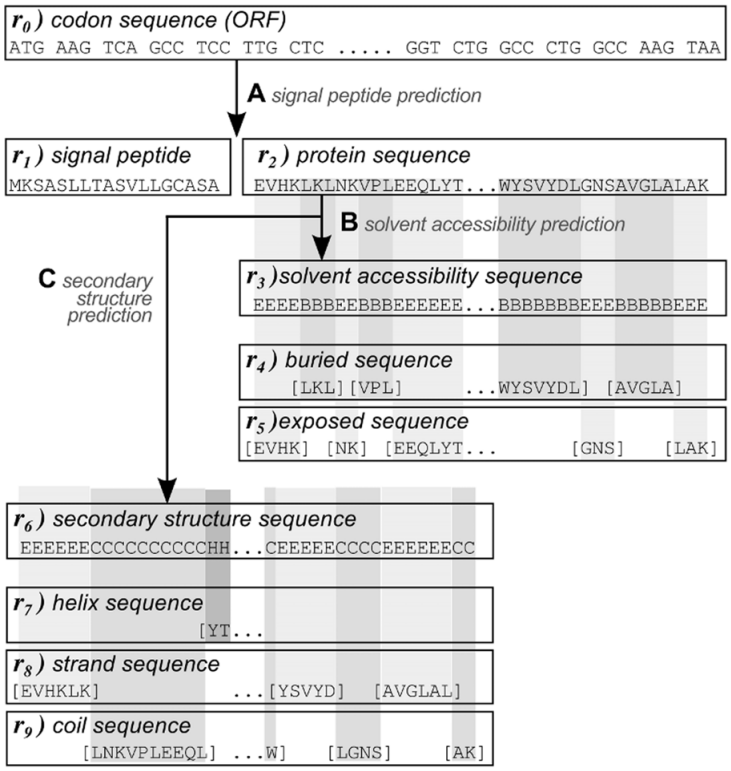


Figure 2.5: Illustration of the different protein sequence representations from which sequence composition was calculated in previous work on protein production in filamentous fungi. (E) and (B) respectively indicate an exposed or buried amino acid, and (E), (C) and (H) represent amino acids in strand, random coil, or helix formations, respectively. This figure is copied from [55].

extraction of which is illustrated in Figure 2.5. They considered multiple sequence representations of each protein: the original DNA sequence, the predicted signal peptide, the amino acid sequence, the predicted solvent accessibility of each amino acid, and the predicted secondary structure. They calculated the normalized composition of each sequence, defined as the count of each of the individual elements, divided by the length of the sequence. The counted "elements" were codons in the DNA sequence, and amino acids in the signal peptide and protein sequences. For the solvent accessibility sequence the proportions of amino acids predicted as buried or exposed were calculated, as well as the amino acid compositions of the buried and exposed units. Similarly, the proportion of elements predicted to be in a helix, random coil or strand formation from the secondary structure sequence were calculated, as well as the amino acid composition of the units predicted to be in helix, strand, or random coils.

For each protein Van den Berg et al. also calculated codon usage, signal peptide length, protein sequence length, the codon adaptation index, and isoelectric point. They included features based on groupings of amino acids, which were defined by amino acid side chain, or optimized specifically for the task using a feed-forward cluster selection procedure. Van den Berg et al. also performed feature selection on the aforementioned features using a combination of t-tests and forward selection to yield a combined set of features, which was tested in addition to the individual feature types. Finally, spectrum kernels [36] were used to calculate the pairwise similarities between sequences based on the occurrence of k-length amino acid subsequences, with $k = 2, 3, 4, 5$.

The table displaying the predictive results for each protein representation is reproduced in Figure 2.6. Out of feature representations derived from the mature amino acid sequence, normalized amino acid composition and feature selection performed the highest for both the homologous and heterologous classifiers. The thorough performance comparison of sequence representations made by van den Berg et al. makes a useful starting point for the experiments described in this thesis.

Table 2. Prediction performance scores (AUROC).

features	*hom→hom*	*het→het*	
Composition-based features			
f_0	**0.85**	0.70	*ORF codon composition*
f_1	0.66	0.51	*signal peptide AA composition*
f_2	0.83	0.70	*mature protein AA composition*
f_3	0.68	0.51	*buried-exposed composition*
f_4 (f_4')	0.80 (0.80)	0.65 (0.64)	*buried AA composition*
f_5 (f_5')	0.82 (0.78)	0.64 (0.65)	*exposed AA composition*
f_6	0.62	0.57	*helix-strand-coil composition*
f_7 (f_7')	0.68 (0.70)	0.60 (0.57)	*helix AA composition*
f_8 (f_8')	0.70 (0.72)	0.61 (0.57)	*strand AA composition*
f_9 (f_9')	0.80 (0.80)	0.65 (0.65)	*coil AA composition*
f_{10}	0.80	0.63	*AA clusters composition*
f_{11}	0.83	0.67	*optimized AA clusters comp.*
Sequence-derived features			
f_{12}	0.64	0.54	*codon usage*
Selected features			
f_{14}	0.84	**0.75**	*feature selection*

Figure 2.6: AUC (defined in Section 3.6) performance of various numeric representations, copied from [55]. *hom* and *het* indicate the performance for a classifier trained and tested on homologous or heterologous sequences, respectively. The table has been cropped to exclude the results of the spectrum kernel representation, which never exceed .82 for the homologous predictor or .63 for the heterologous.

Chapter 3

Computational Background

The primary objective of this thesis is to apply computational methods to predict the success of recombinant protein production in filamentous fungi. This section reviews computational modeling approaches to address recombinant protein production and motivates our use of support vector machines (SVMs). We then describe the basics of SVMs, kernel functions, semi-supervised extensions of the SVM, and the performance metrics used in our analyses. We rely heavily on [49] for Section 3.2 and 3.3.

3.1 Modeling approaches

Computational modeling for recombinant protein production can roughly be divided into two main branches – metabolic modeling of the host cell, and predictive modeling of protein properties.

3.1.1 Metabolic models

As described in Section 2.5, the environmental conditions and the strain of the host organism have profound impacts on the production of the protein of interest. Metabolic modeling uses a formal mathematical framework to represent metabolic processes within the host cell. The model can then be exploited to suggest modifications that optimize for the production of the desired end-products [23]. In the more commonly used stoichiometric modeling of metabolic networks, the modeler first defines the cellular stoichiometric equations that affect the process of interest, where stoichiometric equations describe constraints on the chemical reactions. One common method of analysis, flux-balance analysis (FBA) uses linear programming to find the optimal flux configuration for the model, where a "flux" is the mass of a particular element in the metabolic network [44]. Different configurations of the

model can be explored to gain a deeper understanding of metabolic tradeoffs, and to compare current strain production to the maximum predicted amount of the desired products, which represents a theoretical maximum yield. As an example, Driouch et al. used stoichiometric modeling to compare network fluxes between a strain of *Aspergillus niger* engineered to make a recombinant protein, and an unmodified wild-type strain [16].

Kinetic models, another form of metabolic models, differ from stoichiometric models in their inclusion of not only the reactions themselves, but also the kinetics of the reactions. This allows for the inclusion of regulatory mechanisms in the model, and processes can be modeled with greater accuracy. However, data collection for kinetic models and optimization are also more expensive [2].

One advantage of metabolic modeling is the detailed analysis of the intracellular processes required for protein production, which contrast with the machine learning methods used for predicting protein properties. While metabolic models are able to suggest points in the metabolic network where engineering efforts would be most beneficial, if a prediction is negative in the case of the case of current protein prediction models, the model cannot suggest *where* in the process the failure occurs.

In this thesis, we choose to apply machine learning methods that do not involve modeling cellular metabolism. This is primarily because (1) the available data is better-suited to binary classification as it does not include measures of intra- and extra-cellular metabolites beyond the protein product of interest and (2) our approach has the advantage of requiring no prior knowledge about the protein of interest beyond its amino acid sequence.

3.1.2 Protein-related prediction

Applications of computational modeling that focus on protein properties, as touched upon in Section 2.6, include prediction of protein solubility [24], prediction of protease cleavage sites [51], and prediction of protein destination [32]. While the predictive goals differ, these applications share the same general problem setup. Figure 3.1 presents a high-level overview of the modeling process. The protein strings of the training dataset are taken as input, various "features" are calculated from the amino acid sequence, and these features, along with the class labels of the training data, are used to train a predictive model. The trained model can then be used to predict the class label of a new protein whose label is unknown.

The application most similar to the subject of this thesis is the prediction of solubility upon expression in *E. coli*. In both cases the goal is binary classification, which is the prediction of the input to belong to one of two classes – soluble or insoluble, or in our case, produced or non-produced. Both applications are also broadly interested in the correct transcription, translation, and folding of the re-

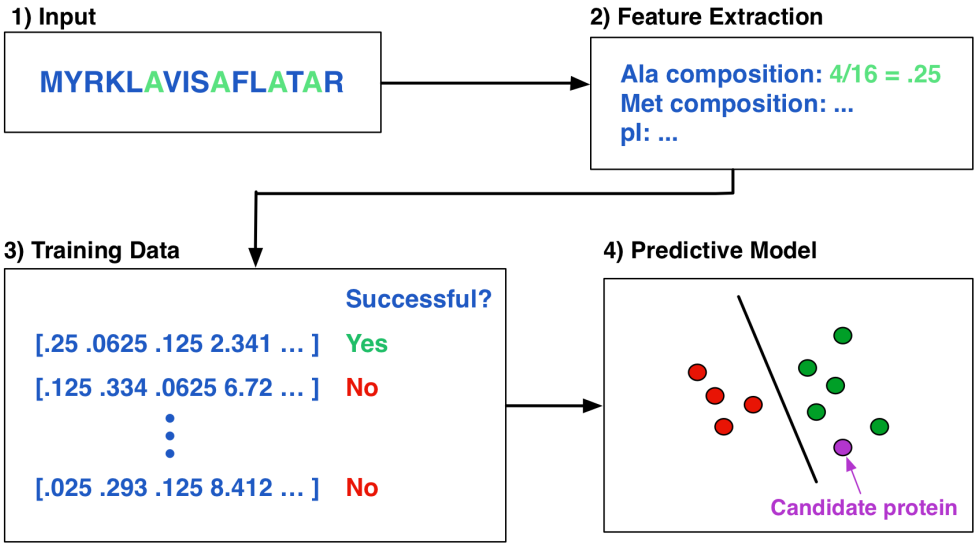


Figure 3.1: Visual overview of training a model to predict protein properties. Numeric representations are extracted from the input protein sequences, which are then used as input to statistical and machine learning methods. The fitted model can be used to predict properties of new sequences.

combinant gene. For this reason, and the fact that predicting protein production in filamentous fungi is a new problem with few related works, we discuss computational approaches to the solubility in *E. coli* modeling problem.

Once extraction of the numeric representation of the proteins has been completed, as described in Section 2.6, statistical tests can be applied to filter out features of the proteins that are informative to the prediction problem. This step is motivated by the possibility that some of the calculated protein features are irrelevant or redundant. To test for the significance of individual features, the calculated values for one group, e.g the soluble proteins, are compared to the values calculated for the other group, e.g. insoluble proteins. The t-test, Mann-Whitney U test, and Kruskal-Wallace test have all been applied to select features for predicting solubility in *E. coli* [9, 26–28]. For example, Hirose and Naguchi used a t-test to select 50 features with distributions significantly different between the insoluble and soluble protein datasets [27]. In the literature the choice of statistical test is rarely justified, which is problematic because the efficacy of the test may depend on the validity of assumptions made on the underlying data distribution [61]. A statistical test may be used as the primary data analysis method, as done by Boettner et al. to identify features related to the characteristic of interest [9], but this does not provide a way to classify new protein sequences, which is our primary aim.

There are a number of classification methods commonly used in machine learning that serve well for protein prediction problems, with implementations readily available in statistical software packages. These include logistic regression, decision trees, linear discriminant analysis (LDA), artificial neural networks (ANNs), k-nearest neighbors (kNN) classifiers, random forest (RF) classifiers, and support vector machines (SVMs). A review of these methods is beyond the space constraints of this thesis, but detailed descriptions of these methods can be found in Bishop's *Pattern Recognition and Machine Learning* [8].

SVMs have been shown to perform highly for both the solubility in *E. coli* and protein production in filamentous fungi prediction tasks. SVMs performed favorably for protein solubility prediction when compared with ANNs and kNN classifiers [29, 38], and RFs [27]. In the only work to date that compared machine learning methods for predicting protein production in filamentous fungi, SVMs performed the best when compared to LDA, kNN, and Naive Bayes [54]. SVMs have several strong points, including few parameters that provide a simple means to adjust the flexibility of the model, a global optimum, efficient optimization methods, and the advantage of offering both linear and non-linear classifiers under the same framework through choice of the kernel function.

Previous work by van den Berg et al. established the efficacy of applying a linear kernel SVM to predict protein producibility in *Aspergillus niger* [55]. The same paper also introduced a publicly available web-server implementation of their predictive tool, HIPSEC [1], and a publicly available dataset of 345 homologous *A. niger* proteins tested for successful production. In this thesis we show that this model generalizes well to proteins produced in *Trichoderma reesei*, and seek to improve the model through inclusion of additional protein features and semi-supervised methods.

3.2 Support Vector Machines (SVMs)

A support vector machine (SVM) is a classification method that maps the input data into a high-dimensional feature space and then learns a linear decision surface in this space. SVMs became a widely used pattern recognition model after 1995, when Cortes and Vapnik extended a similar, previously defined model so that it could handle non-linearly-separable data [14].

A *hyperplane* is a vector of one less dimension than its ambient vector space [59]. A one-dimensional line is a hyperplane in two-dimensional space, a two-dimensional plane is a hyperplane in three-dimensional space, and so on for larger dimensions. An SVM selects a hyperplane through the feature space so that the

[1] http://helix.ewi.tudelft.nl/hipsec/home.py

examples lying on one side of the hyperplane are assigned a different class label than those on the opposite side. The hyperplane is selected so that it minimizes the classification error of the training examples and maximizes the space between examples of either class, called the *margin*. The latter criterion is included to improve generalization of the model to unseen examples.

The formal definition of the SVM from [14] is as follows. We are given a set of L training pairs $\mathcal{L} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)\}, \mathbf{x} \in \mathbb{R}^n, y \in \{-1, 1\}$. An SVM learns a decision function

$$f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x}) + b \tag{3.1}$$

where $\phi(\cdot)$ is the chosen feature map. To return a binary prediction, the value returned by Equation 3.1 is thresholded at 0. The weights $\mathbf{w}$ and intercept b are learned by minimizing

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{L}\xi_i \tag{3.2}$$

subject to

$$y_i f(\mathbf{x}_i) \geq 1 - \xi_i, \quad i = 1, \ldots, L$$

$$\xi_i \geq 0, \quad i = 1, \ldots, L$$

In the above equations, ξ_i is the hinge loss function H on sample i, defined as

$$H(y_i f(\mathbf{x}_i)) = \max(0, 1 - y_i f(\mathbf{x}_i))$$

The parameter C in Equation 3.2 represents a trade-off between the margin, controlled by $||\mathbf{w}||$, and the size of the error terms ξ_i. In practice C is selected by cross-validation. Figure 3.2 illustrates the learned decision function for different values of C and different kernel functions (discussed in the next section) on a toy dataset. Small values of C result in a wide margin and a less accurate classification of the training examples. Larger values of C yield a small margin and a stricter classification of training examples.

Optimization is carried out over the quadratic programming problem formed from the dual form of the function, resulting from the Lagrangian [14]. Introducing the Lagrange multipliers (a.k.a. dual variables) $\{\alpha_i, \ldots, \alpha_L\}$, the learned decision function can equivalently be written

$$f(\mathbf{x}) = \sum_{i=1}^{L} y_i \alpha_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle + b.$$

In fact, the inner product of the feature vectors $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle$ are the only input needed to learn the decision function. This is the kernel function, discussed in the following section.

3.3 Kernels

We use the following definition of a kernel function from [49]:

A *kernel* is a function κ that for all $\mathbf{x}, \mathbf{z} \in X$ satisfies

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$$

where ϕ is a mapping from X to a feature space F

$$\phi : \mathbf{x} \to \phi(x) \in F$$

and $\langle \cdot, \cdot \rangle$ is the inner product operator. In order for this to work, the feature space F must have a corresponding inner product, defined as a real-valued symmetric bilinear (linear in each argument) map $\langle \cdot, \cdot \rangle$, that satisfies $\langle \cdot, \cdot \rangle \geq 0$.

For the vector space $\mathbb{R}^n$ the standard inner product is given by

$$\langle \mathbf{x}, \mathbf{z} \rangle = \sum_{i=1}^{n} x_i z_i.$$

Certain machine learning models, including SVMs, only require as input the inner product between feature vectors and not the feature vectors themselves. This means that, as opposed to first calculating the feature map $\phi(\mathbf{x})$, and then its inner product with other training points, we can directly calculate the inner products using the original input $\mathbf{x}$. The kernel is then a special tool to bypass computationally expensive calculations in a high dimensional feature space, while still *implicitly* representing each input vector in many dimensions.

In this thesis we consider three commonly-used kernel functions for use in the SVM classification model: the linear kernel, Gaussian or "radial basis function" (RBF) kernel, and polynomial kernel.

The linear kernel is defined by the inner product of the original real-valued vectors:

$$\kappa(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{n} x_i z_i = \langle \mathbf{x}, \mathbf{z} \rangle$$

This means that the feature map implicitly used for calculating the linear kernel is the same as the original vector, or equivalently, $\phi(\mathbf{x}) = \mathbf{x}$.

For $\sigma > 0$, the RBF kernel is defined as

$$\kappa(\mathbf{x}, \mathbf{z}) = \exp\left(\frac{-||\mathbf{x} - \mathbf{z}||^2}{2\sigma^2}\right)$$

In most Gaussian kernel implementations, the kernel is instead parameterized by γ, where

$$\gamma = \frac{2}{\sigma^2}$$

Large values of γ correspond with "peaky" Gaussian kernels, while small values of γ produce wider Gaussian kernels that, if decreased far enough, result in a constant function. The standard method to select the γ parameter is by cross-validation, and this is also the method used in this thesis.

A polynomial kernel with degree k is defined as

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^k$$

Figure 3.2 visualizes the form of the decision functions learned using these three different kernel functions.

3.4 Transductive Support Vector Machines (TSVMs)

Transductive support vector machines (TSVMs) extend the SVM framework to make use of unlabeled examples to improve generalization accuracy [12]. Beyond maximizing the margin for labeled data points (as in fully supervised SVMs), TSVMs also seek to maximize the margin for unlabeled data. At a conceptual level, the model is formulated to place the hyperplane in "gaps" between the classes, which are elucidated by the labeled *and* unlabeled data.

Keeping the same notation from Section 3.2, with the addition of U unlabeled examples $\mathcal{U} = \{\mathbf{x}_{L+1}, \ldots, \mathbf{x}_{L+U}\}$, the objective function can be formulated as minimizing

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{L}\xi_i + C^*\sum_{i=L+1}^{L+U}\xi_i^*$$

subject to

$$\begin{aligned} y_i f(\mathbf{x}_i) &\geq 1 - \xi_i, \quad i = 1, \ldots, L \\ |f(\mathbf{x}_i)| &\geq 1 - \xi_i, \quad i = L+1, \ldots, L+U \\ \xi_i &\geq 0, \quad i = 1, \ldots, L \\ \xi_j^* &\geq 0, \quad i = L, \ldots, L+U \end{aligned}$$

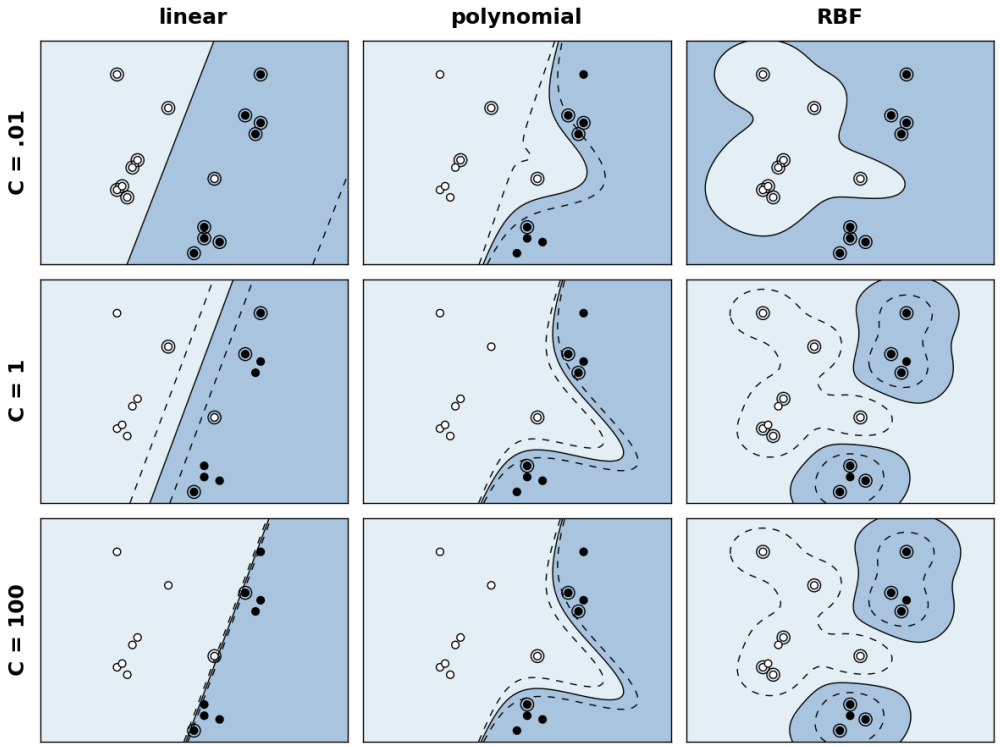


Figure 3.2: Two-dimensional illustration of the decision function learned by an SVM trained on two-dimensional toy data. Columns from left to right use linear, polynomial ($k = 2$), and RBF ($\gamma = 2$) kernels. Rows from top to bottom use a C value of .01, 1, and 100. The hard line shows the classification boundary, and dashed lines indicate the margin boundaries ($f(\mathbf{x}) = \pm.5$). Support vectors are distinguished from other training points by an extra circle drawn around them. Figure adapted from the scikit-learn toolbox documentation[3].

Original code by Gaël Varoquaux http://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html.

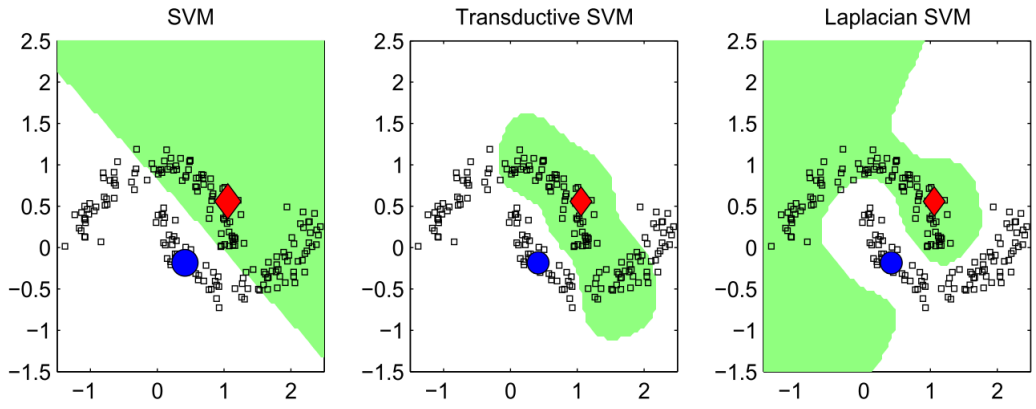


Figure 3.3: Illustration of the classification function learned by the different SVM formulations on a toy dataset. The red and blue data points are the only labeled examples included, and the rest are used as unlabeled data. The green and white background indicates the area belonging to each class. Copied directly from [6].

One problem with the above formulation is that in a high dimensional space with a small number of labeled examples, it is possible to classify all of the unlabeled data as belonging to a single class with a large margin, even though this leads to poor performance in most cases. To prevent this, a balancing constraint is added that enforces the average classification value of the unlabeled data to be equal to the average class label of the labeled data. The constraint can be written as follows:

$$\frac{1}{U}\sum_{i=L+1}^{L+U} f(x_i) = \frac{1}{L}\sum_{i=1}^{L} y_i$$

We use the implementation referenced in [12] and freely available online [4]. They solve the optimization problem by decomposing the loss function into the sum of two distinct loss functions, which enables the use of the "Concave-Convex Procedure" for optimization [62].

3.5 Laplacian Support Vector Machines (LapSVMs)

The Laplacian support vector machine (LapSVM) is an alternate SVM formulation that also uses unlabeled data, but in a different manner than the TSVM described in the previous section. While the TSVM discourages the appearance of unlabeled

[4] http://mloss.org/software/view/19/

data inside the classification margin, the LapSVM encourages unlabeled examples that are geometrically close together to have the same class label.

In the formulation from [6], the parameters γ_A and γ_I are used instead of the C parameter described in Section 3.2. In a standard SVM formulation, $C = \frac{1}{2L\gamma}$ where L is the number of labeled examples and γ is a weight on the norm term in the optimization problem. Note that as opposed to placing C parameters on the slack variable penalty terms, the LapSVM places γ-type parameters on norm terms, that are inversely related to C.

The LapSVM objective is to minimize

$$\frac{1}{l}\sum_{i=1}^{L}\xi_i + \gamma_A||\mathbf{w}||^2 + \frac{\gamma_I}{(u+l)^2}\mathbf{f}L_G\mathbf{f}$$

subject to

$$\begin{aligned} y_i f(\mathbf{x}_i) &\geq 1 - \xi_i, \quad i = 1, \ldots, L \\ \xi_i &\geq 0, \quad i = 1, \ldots, L \end{aligned}$$

In the above formulation L_G is the graph Laplacian, calculated by $L_G = D - W$, where W is the adjacency matrix formed from the data and D is a diagonal matrix formed by $D_{ii} = \sum_{j=1}^{l+u} W_{ij}$. The adjacency matrix is a binary matrix of size $L + U \times L + U$ where $W_{ij} = 1$ if $\mathbf{x}_i$ is connected to point $\mathbf{x}_j$, and equals 0 otherwise. $\mathbf{f} = [f(\mathbf{x}_1, \ldots, f(x_{l+u})]^T$ is the vector of predicted values over all of the data points. Note that $\mathbf{f}L_G\mathbf{f}$ can be expanded

$$\mathbf{f}L_G\mathbf{f} = \sum_{i,j=1}^{L+U}(f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 W_{ij}$$

In this formulation it is clearer that if the labels assigned to $\mathbf{x}_i$ and $\mathbf{x}_j$ by the classification function f are different, and they are "close" as defined by the data adjacency matrix W, then the value of the objective function will increase. This encourages "close" data points to be assigned the same class label.

A visual illustration of the SVM, TSVM, and LapSVM formulations on a toy dataset is shown in Figure 3.3.

We make use of the implementation from [52], which is publicly available online[5]. The data adjacency matrix was found using k-Nearest Neighbors with $k = 10$.

[5]http://www.dii.unisi.it/ melacci/lapsvmp/

3.6 Performance measures

Define $Y = \{y_i\}_{i=1}^{l}$ s.t. $y_i \in \{-1, 1\}$ as the true class assignment of each example and $\hat{Y} = \{\hat{y}_i\}_{i=1}^{l}$ s.t. $\hat{y}_i \in \{-1, 1\}$ as the predicted class assignment. We make use of the following abbreviations:

1. TP: True positives, the number of instances where the true class assignment is positive and the predicted class assignment is positive as well

$$\text{count}\{\, i \,|\, \hat{y}_i = y_i \,,\, y_i = 1\}$$

2. FN: False negatives, the number of instances where the true class assignment is positive and the predicted class assignment is negative

$$\text{count}\{\, i \,|\, \hat{y}_i \neq y_i \,,\, y_i = 1\}$$

3. TN: True negatives, the number of instances where the true class assignment is negative and the predicted class assignment is negative as well

$$\text{count}\{\, i \,|\, \hat{y}_i = y_i \,,\, y_i = -1\}$$

4. FP: False positives, the number of instances where the true class assignment is negative and the predicted class assignment is positive

$$\text{count}\{\, i \,|\, \hat{y}_i \neq y_i \,,\, y_i = -1\}$$

5. False Positive Rate (FPR): the fraction of negative examples that are incorrectly predicted to be positive examples

$$\frac{FP}{TN + FP}$$

6. True Positive Rate (TPR): the fraction of positive examples that are correctly predicted to be positive examples

$$\frac{TP}{FN + TP}$$

7. True Negative Rate (TNR): the fraction of negative examples that are correctly predicted to be negative examples

$$\frac{TN}{FP + TN}$$

We can then define the performance measures listed below. All measures range in value from 0 to 1.0:

1. Accuracy: the fraction of labeled examples that are correctly classified by the model
$$\frac{TP + TN}{TP + FP + TN + FN}$$
2. Precision: the fraction of positive class predictions that are correct
$$\frac{TP}{TP + FP}$$
3. Recall: the fraction of positive examples that are correctly classified by the model
$$\frac{TP}{TP + FN}$$
4. Balanced accuracy: a measurement of accuracy that gives equal emphasis to each class regardless of the number of examples from each class
$$.5 * (TPR + TNR)$$

Another measure we use is the **area under the receiver operating characteristic curve** (AUC). Consider again our true labels $\mathbf{y} = \{y_i\}_{i=1}^{l}$ s.t. $y_i \in \{-1, 1\}$, but now define our predicted values as real valued numbers, so that $\hat{\mathbf{y}} = \{\hat{y}_i\}_{i=1}^{l}$ s.t. $\hat{y}_i \in \mathbb{R}$. With many classifiers, including SVMs, we define a thresholding classification function, $c(\hat{y})$ that converts the real-valued prediction to a discrete class label. Considering a threshold t we can define the classification function as:

$$c(\hat{y}) = \begin{cases} 1, & \text{if } \hat{y} > t \\ -1, & \text{if } \hat{y} \leq t \end{cases}$$

A good threshold returns predictions $\hat{\mathbf{y}}$ that, when evaluated against the ground truth, give a *high* True Positive Rate (TPR) and *low* False Positive Rate (FPR). If we set the threshold t equal to the maximum predicted value, $\max(\hat{\mathbf{y}})$, then all examples are given a label of -1, and both TPR and FPR are equal to 0. In the other extreme, if we set t to the minimum predicted value, $\min(\hat{\mathbf{y}})$, then all examples are given a label of $+1$ and both the TPR and FPR are at their maximum value of 1.0. The receiver operating characteristic curve (ROC) plots the false positive rate on the x-axis and the true positive rate on the y-axis as a function of a decreasing classification threshold. If a threshold exists that can

perfectly separate the two classes, then the classifier performs optimally and AUC is equal to 1.0. AUC thus has the advantage of being independent of the selection of the specific threshold t chosen to binarize the predicted labels. We can raise and lower precision and accuracy by choosing a different value of t. AUC can also be analyzed visually by directly plotting the ROC curve.

Chapter 4

Materials and methods

4.1 Data

Four labeled datasets were used in this thesis, with basic statistics given in Table 4.1. The *Aspergillus niger* dataset consists of 345 homologous sequences. It was first introduced in [54], and then shared publicly in [55]. According to [54], each sequence was inserted into a vector with the strong glucoamylase promoter (P_{GlaA}), and the modified cells were grown in shake flasks, filtered, and put on SDS-PAGE gel, where the detection of a visible band in the gel was defined as successful production.

Both the heterologous and homologous datasets tested in the host species *Trichoderma reesei* are internal to VTT Technical Research Centre of Finland Ltd. The heterologous dataset of ORFs tested in the host *T. reesei* consists of 19 sequences from a variety of species. These were tested at different times over the course of many years, with different experimental protocols. The homologous dataset consists of 31 sequences, which were selected for their high likelihood of being secreted according to previous proteomics experiments [4], and include known highly produced cellulases such as CBHI and CBHII. All 31 sequences were deemed to be successfully produced according to their detection in protein gels.

For the *T. reesei* heterologous sequences, an additional Northern Blot analysis was conducted, which tests the amount of mRNA produced for each sequence. Sequences without an adequate amount of detected mRNA were excluded from analysis, as this suggests that these sequences failed at the transcription stage.

All datasets were filtered using CD-HIT [19, 37] with all parameters at default values except a 60% sequence identity threshold, which was calculated as the number of amino acids in alignment divided by the length of the shorter sequence. The *T. reesei* combined dataset was created by first merging the unfiltered heterologous and homologous datasets, and then filtering using CD-HIT. This filtering step

was included to avoid an over-representation of one protein family in the datasets, which could bias the model to perform well on only a particular type of protein.

Dataset	Host species	Seq species of origin	Size	Positive	Negative
A. niger	*A. niger*	*A. niger*	345	178	167
T. reesei homologous	*T. reesei*	*T. reesei*	31	31	0
T. reesei heterologous	*T. reesei*	various	19	10	9
T. reesei combined	*T. reesei*	various	49	40	9

Table 4.1: Basic statistics of datasets used in experiments.

For semi-supervised methods we extracted unlabeled protein sequences from the protein databases of the October 2013 version of UniProt [13]. We specifically drew unlabeled protein sequences from UniProt TrEMBL [13], the computer-annotated section of the UniProt Knowledgebase. TrEMBL contains translations of all coding regions in the DDBJ/EMBL/GenBank nucleotide databases, and protein sequences extracted from the literature or submitted to UniProtKB, which are not yet integrated into Swiss-Prot. We used TrEMBL instead of Swiss-Prot because more protein sequences are available in TrEMBL, and we did not need the functional annotations provided in Swiss-Prot for semi-supervised experiments. Swiss-Prot from the same UniProt version was used to test for enrichment of EC numbers and organism taxonomies in the positive predictions of the model.

4.2 Feature selection

We applied two different numeric representations of the protein sequences: amino acid composition and InterProScan features.

We calculated the normalized amino acid composition of each protein sequence for each of the "standard" amino acids (see Figure 2.1). Define n_i as the number of times that amino acid i occurs in the input protein sequence. Then each feature vector can be defined as $\mathbf{x} \in \mathbb{R}^{20} = \{x_1, \ldots, x_{20}\}$ where

$$x_i = \frac{n_i}{\sum_{j=1}^{20} n_j} \tag{4.1}$$

These were the highest performing sequence features tested on the *A. niger* dataset in [55].

InterProScan is "a tool that combines different protein signature recognition methods into one resource"[1] [31]. InterProScan takes a protein amino acid sequence as input and returns protein signatures detected by the following methods:

[1] http://www.ebi.ac.uk/Tools/pfa/iprscan5/

1. **BlastProDom**: Scans the families in the ProDom database. ProDom is a comprehensive set of protein domain families automatically generated from the UniProtKB/Swiss-Prot and UniProtKB/TrEMBL sequence databases using psi-blast.

2. **FPrintScan**: Scans against the fingerprints in the PRINTS database. These fingerprints are groups of motifs that together are more potent than single motifs by making use of the biological context inherent in a multiple motif method.

3. **HMMPIR**: Scans the hidden markov models (HMMs) that are present in the PIR Protein Sequence Database (PSD) of functionally annotated protein sequences, PIR-PSD.

4. **HMMPfam**: Scans the hidden markov models (HMMs) that are present in the PFAM Protein families database.

5. **HMMSmart**: Scans the hidden markov models (HMMs) that are present in the SMART domain/domain families database.

6. **HMMTigr**: Scans the hidden markov models (HMMs) that are present in the TIGRFAMs protein families database.

7. **ProfileScan**: Scans against PROSITE profiles. These profiles are based on weight matrices and are more sensitive for the detection of divergent protein families.

8. **HAMAP**: Scans against HAMAP profiles. These profiles are based on weight matrices and are more sensitive for the detection of divergent bacterial, archaeal and plastid-encoded protein families.

9. **PatternScan**: PatternScan is a new version of the PROSITE pattern search software which uses new code developed by the PROSITE team.

10. **SuperFamily**: SUPERFAMILY is a library of profile hidden Markov models that represent all proteins of known structure.

11. **HMMPanther**: A library of hidden Markov models representing families of genes.

12. **Gene3D**: A database of protein domain structure annotations for protein sequences.

13. **Phobius**: Predicts transmembrane topology and signal peptides

14. **Coils**: Calculates the probability that the input sequence will adopt a coiled-coil conformation

We used all InterProScan signatures that were detected in at least two labeled proteins in the union of the *A. niger* and *T. reesei* combined datasets. This amounted to 391 signatures, the names of which are given in Appendix 7. A binary feature vector $\mathbf{x} \in \{0,1\} = \{x_1, \ldots, x_{391}\}$ was constructed for each protein sequence by placing a 1 if the signature was detected in the protein, and a 0 otherwise.

4.3 Feature combination and kernel choice

We evaluated the predictive performance of an SVM trained with amino acid (AA) composition only, InterProScan only, or a combination of these features, and a linear, polynomial, or RBF kernel.

We trained and tested two models - one trained on the *A. niger* dataset and the other trained on the *T. reesei* combined dataset. For experiments using the *A. niger* dataset, we used 10-fold cross-validation to train and evaluate the model, and for the *T. reesei* combined dataset we used 5-fold cross-validation. 5-fold cross-validation was used instead of 10 for the *T. reesei* dataset so that there were a greater number of examples for evaluation in each fold. The cross-validation splits were the same across all choices of features and kernels to allow for direct comparison.

The SVM C parameter and kernel parameters were set using a 3-fold inner cross-validation loop over the training data for each fold. The parameters that resulted in the highest average balanced accuracy across the inner cross-validation loops were selected. In these experiments the C parameter and γ parameter for the RBF kernel were set using a grid search over 10^i for $i = \{-3, -2, -1, 0, 1, 2, 3\}$, and the k parameter for the polynomial kernel was set with a grid search over $\{2, 3, 4, 5, 6\}$.

In addition to testing InterProScan and AA composition features individually, we tested whether a combination of these features would boost performance. We tested concatenation of the features prior to calculating the kernel, and the element-wise product of the two kernels. Prior to taking the product, each kernel κ was normalized so that

$$\hat{\kappa}(\mathbf{x}, \mathbf{z}) = \left\langle \frac{\phi(\mathbf{x})}{||\phi(\mathbf{x})||}, \frac{\phi(\mathbf{z})}{||\phi(\mathbf{z})||} \right\rangle = \frac{\kappa(\mathbf{x}, \mathbf{z})}{\sqrt{\kappa(\mathbf{x}, \mathbf{x})\kappa(\mathbf{z}, \mathbf{z})}} \tag{4.2}$$

In normalizing the kernels we aimed to give both sets of features approximately equal emphasis despite differences in magnitude.

4.4 Training dataset size

Given that laboratory time and resources are needed to gather the data necessary to train a predictive model, it is of interest *how many labeled sequences* are necessary to achieve a desired performance. To investigate this, we calculated predictive performance as a function of a decreasing number of labeled training examples.

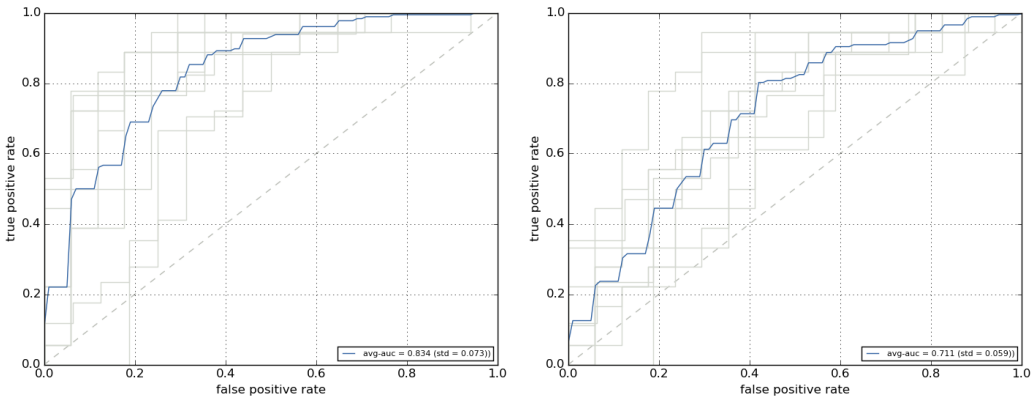
This experiment was carried out on the *A. niger* dataset. We first assigned the data to 10 random cross-validation folds. For each fold, we held the test examples constant for all iterations, but decreased the number of training examples by 10 in every iteration. Define $L_0 \in \mathcal{L}$ as the training examples originally assigned to a particular fold. At every iteration i we took a random subsample of the training examples used in the previous iteration, $L_i \in L_{i-1}$, such that the number of training examples was 10 less than the previous iteration, or more formally $|L_i| = |L_{i-1}| - 10$. So if for the first iteration, $|L_0| = n$, then $|L_1| = n - 10$, $|L_2| = n - 20$, and so on until there were no fewer than 10 training examples. In order to separate the effect of fewer training examples from the number of examples present from each class, the subsample at every iteration was taken so that the ratio of positive to negative training examples was the same as the previous iteration.

This experiment was repeated for 3 different randomized cross-validation splits of the data. We ran the experiment for both RBF and linear kernels because of the high performance of these kernels on the *A. niger* data (Table 5.2). C and γ parameters were selected separately for each fold using a second 5-fold cross-validation loop within the training data, such that the parameters maximized the average accuracy across the 5 inner folds. Parameter selection was performed once in the first iteration using all labeled training data, and these parameters were used for all the following iterations.

4.5 Unlabeled sequence selection for semi-supervised experiments

Semi-supervised methods use a combination of labeled and unlabeled data. Since modern sequencing technologies have made a wealth of protein sequences publicly available, the question immediately arises of *how to select* the protein sequences that form the unlabeled dataset.

As a first step, we tested the use of the entire *Aspergillus niger* genome, consisting of $14,097$ sequences, in conjunction with the labeled *A. niger* data. We trained a linear kernel TSVM using 10-fold cross-validation in a labeled-only condition wherein we used only the labeled *A. niger* sequences for training and testing, and a semi-supervised condition in which we used the labeled sequences combined



(a) labeled data only (b) with *A. niger* genome

Figure 4.1: TSVM 10-fold cross-validation ROC curves for the *A. niger* dataset trained on amino acid composition features. Gray lines show the ROC curve for individual folds, and the blue line is the average ROC curve across folds. Figure (a) shows the results using only the labeled data, and (b) shows results with all labeled data combined with the unlabeled *A. niger* genome sequences.

with all 14,097 unlabeled sequences from the *A. niger* genome. The C^* parameter was kept at its default value of $C^* = \frac{LC}{U}$ where L and U are the number of labeled and unlabeled examples, respectively. Figure 4.1 (a) shows the ROC curves when using the labeled *A. niger* data only and (b) shows the ROC curves when training with the labeled *A. niger* data plus all genome sequences. These figures show how the inclusion of the sequences from the *A. niger* genome hurts model performance, decreasing from an average AUC of .83 to .71.

Given the lack of improvement afforded by the use of the *A. niger* genome, we sought alternative unlabeled sequences. We conducted initial selection experiments using the *A. niger* dataset and the UniProt SwissProt database from October of 2013, consisting of over 500,000 unlabeled protein sequences. We used the Basic Local Alignment Search Tool (BLAST) [3] [10] to search for unlabeled sequences, with the idea that sequences relevant to our prediction problem could be identified by homology to the labeled sequences. BLAST finds pairwise local alignments between sequences and also returns an "e-value" that describes the significance of the alignment compared to a random model.

We first performed a BLAST of the *A. niger* dataset against the UniProt SwissProt sequences, and selected all sequences that were within a BLAST e-value of 10 to the labeled dataset. This returned 19,845 sequences. A histogram of these sequences by BLAST e-value to the closest labeled sequence is shown in Figure 4.2. There are a greater number of proteins with weaker homology to the

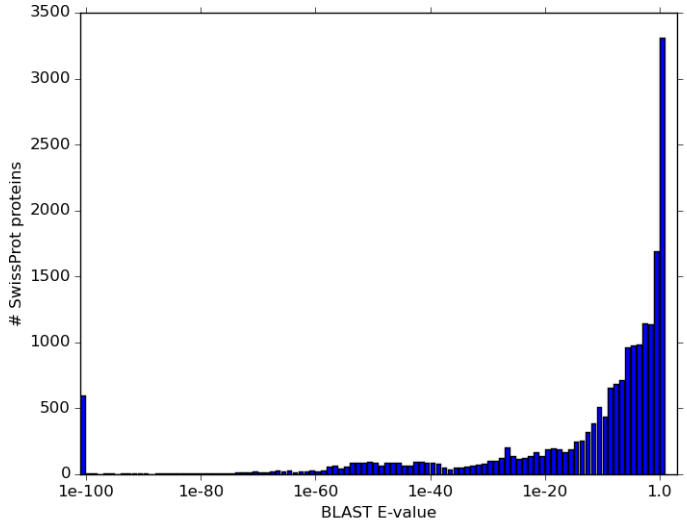
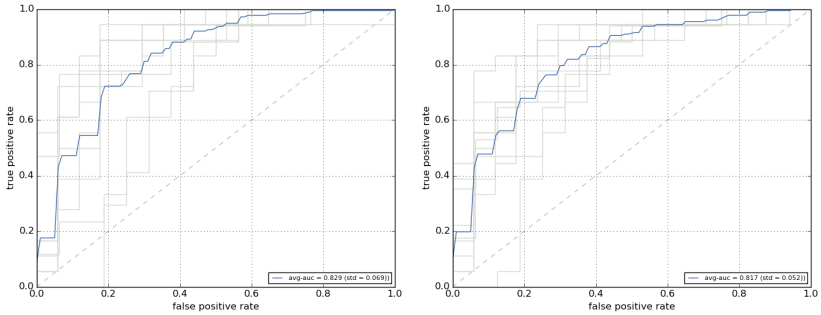


Figure 4.2: Histogram of number of sequences vs. smallest BLAST e-value to the labeled *A. niger* sequences.

labeled sequences, as shown by the increase in number of sequences as the e-value increases.

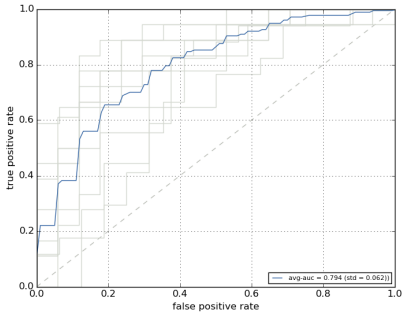
We conducted experiments to determine the effect of sequence distance in BLAST e-value space on performance using the TSVM model. We separated the 19,845 unlabeled SwissProt sequences selected in the previous step into the following 5 bins according to their e-value: 0–1E-60, 1E-60–1E-20, 1E-20–1E-1, 1E-1–1E0, and 1E0–1E1. We then performed 5 rounds of 10-fold cross-validation with a linear kernel TSVM. In each round we used the same labeled *A. niger* examples for training, but different unlabeled sequences. In the first round we randomly sampled 1000 unlabeled sequences from the first e-value bin, and used these as the unlabeled sequences. In the second round, we used 500 randomly selected from the first bin and 500 from the second bin. In the third, we randomly sampled 333 sequences each from the first 3 bins, and so on for the fourth and fifth round. Thus in each round the number of unlabeled sequences was kept constant, but the maximum BLAST e-value was increased. All 1000 unlabeled sequences were used for training each cross-validation fold. The resulting ROC curves are shown in Figure 4.3. Based on Figure 4.3, the highest performance of the TSVM model resulted when using sequences from the first two bins. We decided to limit unlabeled sequences to those with BLAST e-value of $\leq$ 1E-20.

In order to have access to a larger number of unlabeled sequences for future experiments, we switched from using the UniProt SwissProt database to the UniProt

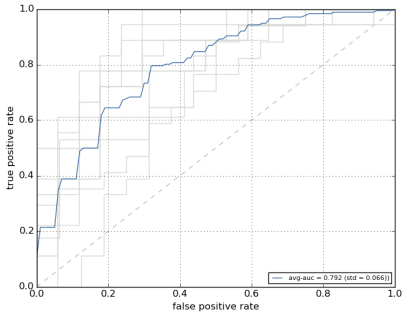


(a) e-value < 1E-60

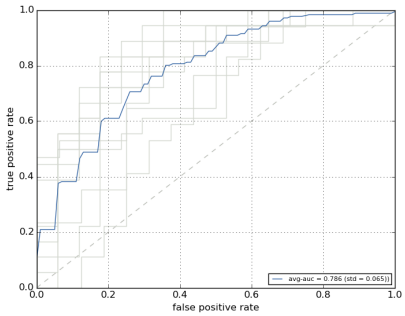
(b) e-value < 1E-20



(c) e-value < 1E-1



(d) e-value < 1E0



(e) e-value $\leq$ 1E1

Figure 4.3: ROC curves for 10-fold cross-validation of linear TSVM trained on *A. niger* dataset using amino acid composition features. Gray lines show ROC curves for each fold, and the blue line shows the average ROC across folds. 1000 unlabeled sequences were used during training for all figures a-e, but for each figure the threshold on the BLAST e-value was increased. The BLAST e-value was given by a BLAST of unlabeled sequences against labeled *A. niger* sequences.

TrEMBL database, which contains over 54 million sequences. The e-value is sensitive to the number of sequences in the database, so to maintain consistency with the BLAST e-value binning experiments on the SwissProt database, we specified that the size of the database for e-value calculations for the TrEMBL sequences be set equal to 542,901, which is the number of sequences in the SwissProt database. The BLAST of the labeled *A. niger* and *T. reesei* sequences against the TrEMBL database returned 307,301 unlabeled candidate TrEMBL sequences for the *A. niger* dataset and 72,679 sequences for the *T. reesei* dataset.

4.6 TSVM and LapSVM experiments

We compared the performance of two semi-supervised SVM formulations, the Transductive SVM (Section 3.4) and Laplacian SVM (Section 3.5), and two different sampling procedures to select the unlabeled sequences to use during training. Both procedures sampled from the TrEMBL sequences with BLAST e-value of $\leq$ 1E-20 when aligned with the labeled sequences, the selection of which is described in the previous section. We tested both a linear kernel and radial basis function (RBF) kernel.

In the "random" sampling method, we selected the unlabeled sequences to use for training by random selection from the pool of TrEMBL sequences with small BLAST e-value. That is, to construct the "random 500" dataset, for every cross-validation fold we randomly selected 500 sequences from the unlabeled TrEMBL sequences, and these sequences plus the labeled data were used during training for that fold. The same procedure was conducted for the "random 2000" dataset, except in this case 2000 sequences were selected per fold.

The "close" method sampled TrEMBL sequences that were close to labeled sequences in amino acid content. Rather than selecting sequences close to *any* labeled sequence, we sought to distribute the unlabeled sequences somewhat evenly by selecting the k nearest neighbors of each labeled training sequence, as defined by Euclidean distance between amino acid content vectors. Defining s as the input sample size and N as the number of labeled training sequences, the number of nearest-neighbors to select for each labeled sequence was set equal to $k = \texttt{ceil}(\frac{s}{N})$. The number of sequences to sample s was set to either 500, for "close 500," or 2000, for "close 2000." Since the number of sequences to sample per labeled protein was rounded up using the *ceil* function, in some cases the total number of selected sequences exceeded 500 or 2000.

Training and testing was completed using the same cross-validation splits as used in Section 4.3. For the *A. niger* data, we used all labeled examples for training in one condition, and a randomly sampled subset of 10 labeled training examples in another condition. The second situation was to test for improvements when few

labeled training examples are available. As in Section 4.4, the 10 labeled examples were sampled so that the proportion of negative and positive examples was the same as in the full labeled dataset.

The C parameter for the SVM (see Section 3.2) and γ parameter for the RBF kernel (see Section 3.3) were set once using all labeled examples. The best parameters for each cross-validation fold were selected as the parameters that maximized the average balanced accuracy within an inner 3-fold cross-validation loop. These parameter values, set once for each dataset, were applied for all transductive experiments, which used the same cross-validation splits as the labeled-data-only experiments. In addition, the parameter values set in this manner using all labeled examples of the *A. niger* dataset were applied to the model trained on only 10 labeled examples. Since the Laplacian SVM uses an alternate formulation of the SVM objective function, the parameter γ_A was set to the equivalent C value, or $1/(2 \cdot C)$ (Section 3.5).

The regularization terms for the TSVM, C^* (see Section 3.4), and LapSVM, γ_I (see Section 3.5), were selected independently for every transductive experiment. For each fold the C^* or γ_I was selected that maximized average accuracy across an inner 3-fold cross-validation loop. A grid search over 10^i for $i \in \{-3, -2, -1, 0, 1, 2, 3\}$ was performed to select C^* and a grid search over {500, 50, 5, .5, .05, .005, .0005} was used to select γ_I. The differences in the latter grid are due to the inverse relationship between C and γ_I (see Section 3.5).

4.7 Enrichment tests

The model evaluations in Section 4.3 give estimates for how well we can predict the success of available labeled data, primarily composed of sequences from the native *T. reesei* and *A. niger* genomes. In this section we sought to understand what sequences are predicted to be "producible" at a wider scale, by analyzing model predictions on the entire Uniprot SwissProt database of protein sequences. We tested for enrichment of "successful" predictions in taxonomic classification of the organism from which the sequence originates, and in enzyme function as specified by Enzyme Commission (EC) numbers.

We tested six variants of a fully-supervised SVM trained using amino acid composition – models trained using a linear, polynomial, and RBF kernel and either the *A. niger* or *T. reesei* dataset. We applied these six models to predict the producibility of the 542,901 sequences in the October 2013 UniProt SwissProt database. For each model, parameters were selected to maximize the balanced accuracy averaged across 5 cross-validation folds. The real-valued predictions on the SwissProt sequences were thresholded at 0, so that positive values were considered predictions of "success," and negative of "failure."

SwissProt includes taxonomic classification of the organism from which each protein sequence originates. An example entry from the database for a plant protein sequence is *Eukaryota; Viridiplantae; Streptophyta; Embryophyta; Tracheophyta,Spermatophyta; Magnoliophyta; eudicotyledons; Gunneridae, Pentapetalae; rosids; fabids; Fabales; Fabaceae; Papilionoideae,Fabeae*. To separate sequences by the classification of their origin species, we defined taxonomic categories by the fifth level of their taxonomic classification, where each level is separated by semi-colons. In the example sequence taxonomy, the sequence would belong to the "Tracheophyta" category.

47% of the sequences in SwissProt also have an Enzyme Commission (EC) number. The EC number system, assigned by the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology (NC-IUBMB)[2] describes the reactions catalyzed by the protein. In tests for enrichment by EC class, we separated sequences by the third number in their EC number. As an example, cellulases have an EC number of 3.2.1.4, so in our enrichment tests they would be counted in the "3.2.1" category.

To test for a significantly high number of predicted "successes" per category, we used the hypergeometric statistical test, implemented as `phyper` in the R programming language. Given a sample population, this tests whether a sub-sample of items, taken without replacement, contains a larger-than-expected proportion of items of a particular type. In our case, the "sample population" is the SwissProt database, the sub-sample taken consists of all sequences with a positive prediction, and the "type" tested for significant presence in the sub-sample is either the taxonomy of the organism of origin or the sequence EC number.

We only considered EC number or taxonomic categories that contained at least 20 sequences in SwissProt, to avoid deeming a category as "enriched" with positive predictions if there were very few examples from which to draw this conclusion.

[2] http://www.chem.qmul.ac.uk/iubmb/enzyme/

Chapter 5

Results and Discussion

5.1 Performance of existing tools

There are many existing tools to predict protein solubility upon expression that are primarily trained on proteins tested in an *E. coli* host (see Section 3.1.2). As of yet there is little discussion in the literature about how well prediction tools can generalize between host organisms. We investigated whether tools aimed at solubility prediction in *E. coli* can also be applied to predict the secretion of recombinant proteins in filamentous fungi. Specifically, we applied two reputedly high-performing *E. coli* solubility prediction tools, `ccSOL omics` and `PROSO II`, on the aformentioned VTT *T. reesei* datasets of 19 homologous and 31 heterologous proteins, and the *A. niger* dataset of 345 proteins [55].

The resulting accuracies are shown in Table 5.1. The prediction accuracies using `HIPSEC` [55], the previously developed tool for predicting protein secretion in filamentous fungi, are shown for comparison.

Table 5.1: Accuracies for tool predictions on filamentous fungi secretion datasets

	A. niger [55]	*T. reesei* hom	*T. reesei* het
`ccSOL omics` [1]	.50	.45	.63
`PROSO II` [50]	.52	.68	.53
`HIPSEC` [55]	.80	.81	.84

While `ccSOL omics` and `PROSO II` have reputedly high performance on large datasets of proteins produced primarily in *E. coli*, they do not perform well in predicting the production and secretion of homologous and heterologous proteins in filamentous fungi. This establishes the need for prediction tools that are tailored to the desired host organism. Interestingly, the `HIPSEC` predictor performs very

well on the *T. reesei* data, which motivates our use of the data and methods used to train the `HIPSEC` model.

5.2 Feature combination and kernel choice

The performance metrics of AUC and balanced accuracy are shown for the model trained on the *A. niger* dataset in Table 5.2, and the metrics for the model trained on the *T. reesei* combined dataset are shown in Table 5.3. To calculate the balanced accuracy the classification function, given in Equation 1.1 in Section 3.2, was thresholded at 0. The labels "aacomp," "interproscan," "concat," and "prod" respectively refer to amino acid (AA) composition features only, InterProScan features only, concatenation of the features prior to kernel calculation, and their normalized kernel product. The type of kernel is shown in parentheses, and the highest AUC achieved for each dataset is bolded.

The *A. niger* column in Table 5.2 shows the average performance of the model on the left-out test data, averaged across the 10 cross-validation folds. The standard deviation from the average across folds is given in parentheses. The *T. reesei* column in Table 5.2 shows the performance when the model, trained on the *A. niger* data, predicts the labels of the *T. reesei* dataset. Table 5.3 shows the average performance across folds under the *T. reesei* column, and the generalization to the *A. niger* data under the *A. niger* column.

The best performance on the *A. niger* data was .84 AUC when training with the *A. niger* data (Table 5.2) and .75 when training with the *T. reesei* combined data (Table 5.3). With both training datasets, the combination of a linear kernel with amino acid (AA) composition features gave the best results on the *A. niger* data. However, the performance using AA composition and a kernel choice other than linear were nearly equal to the performance of the linear kernel. In fact, the choice of kernel had little effect on performance for the *A. niger* dataset.

For the *T. reesei* data, the highest performance when training with the *A. niger* data was .89 AUC (Table 5.2), and an average of .80 AUC when training with the *T. reesei* data. In both cases, the highest achievement on the *T. reesei* dataset was with a polynomial kernel. It is unclear why the polynomial kernel performed slightly better on the *T. reesei* data than using a linear or RBF kernel. It could be related to the significant class imbalance in the *T. reesei* dataset. With over 4/5 of the dataset belonging to the positive class, the boost on the *T. reesei* dataset would make sense if the polynomial kernel shape admits more positive predictions.

The models trained using InterProScan features (as described in Section 4.2) did not perform as well as those trained with only AA composition. However, the InterProScan features on their own were informative for prediction on both datasets - a linear kernel model with only the InterProScan features trained on the

Table 5.2: Performance of feature/kernel pairs for model trained with *A. niger* data. *A. niger* and *T. reesei* column headers indicate the dataset used for evaluation. The *A. niger* column shows the cross-validation average with its standard deviation in parentheses.

	A. niger		*T. reesei*	
	AUC	Bal Acc	AUC	Bal Acc
aacomp (linear)	**.84 (.08)**	.76 (.09)	0.86	0.84
interproscan (linear)	.74 (.12)	.67 (.12)	0.83	0.68
concat (linear)	.75 (.10)	.68 (.08)	0.78	0.60
prod (linear)	.78 (.11)	.70 (.08)	0.81	0.76
aacomp (rbf)	**.84 (.07)**	.77 (.08)	0.83	0.73
interproscan (rbf)	.75 (.12)	.67 (.12)	0.74	0.69
concat (rbf)	.75 (.11)	.67 (.11)	0.80	0.61
prod (rbf)	.76 (.10)	.68 (.11)	0.76	0.62
aacomp (polynomial)	.82 (.09)	.77 (.09)	**0.89**	0.88
interproscan (polynomial)	.75 (.11)	.68 (.12)	0.78	0.60
concat (polynomial)	.74 (.11)	.68 (.11)	0.79	0.60
prod (polynomial)	.77 (.11)	.70 (.10)	0.82	0.69

Table 5.3: Performance of feature/kernel pairs for model trained with *T. reesei* combined data. *A. niger* and *T. reesei* column headers indicate the dataset used for evaluation. The *T. reesei* column shows the cross-validation average with its standard deviation in parentheses.

	A. niger		*T. reesei*	
	AUC	Bal Acc	AUC	Bal Acc
aacomp (linear)	**0.75**	0.69	.64 (.22)	.61 (.19)
interproscan (linear)	0.67	0.51	.58 (.24)	.59 (.21)
concat (linear)	0.73	0.51	.64 (.35)	.70 (.18)
prod (linear)	0.72	0.51	.60 (.23)	.56 (.19)
aacomp (rbf)	0.74	0.67	.71 (.26)	.59 (.20)
interproscan (rbf)	0.51	0.51	.52 (.22)	.47 (.03)
concat (rbf)	0.51	0.51	.60 (.24)	.54 (.11)
prod (rbf)	0.51	0.51	.59 (.29)	.47 (.03)
aacomp (polynomial)	0.74	0.70	**.80 (.17)**	.61 (.19)
interproscan (polynomial)	0.69	0.51	.53 (.31)	.64 (.21)
concat (polynomial)	0.72	0.50	.55 (.28)	.59 (.21)
prod (polynomial)	0.72	0.52	.60 (.17)	.64 (.18)

A. niger data yielded an average AUC of .74 on the *A. niger* data, and an AUC of .83 on the *T. reesei* data (Table 5.2). Given that both the InterProScan and AA composition features performed well on the prediction problem on their own, it is surprising that their combination did not surpass the performance of amino acid composition alone. Taking the linear kernel SVM trained and tested on the *A. niger* data as an example, AA composition alone achieved .84, InterProScan alone achieved .74, their concatenation achieved .75 and their product, .78. This effect of InterProScan features on their own performing the worst, their combination with AA composition performing slightly better, and finally AA composition alone performing the best, occurred across nearly all combinations of dataset and kernel type.

One possibility is that InterProScan features do not provide information that is not available in the amino acid composition alone – i.e. that no new information is added. To investigate this possibility, we compared the predictions on the *A. niger* dataset made by the linear kernel SVM trained using only InterProScan features, and the linear kernel model trained using only AA composition. There were 39 sequences in total that were correctly classified by the InterProScan model, but incorrectly classified by the AA composition model. 10 of these had positive labels in the original dataset and 29 had negative. This hints that there could be information in the InterProScan features not present in AA composition. In addition, while there is a positive correlation between the predictions made by the AA composition model and those made by the InterProScan model ($r = .51$, $p < .0001$), this is lower than one might expect from two feature sets containing the same information.

Another explanation for the performance drop when AA composition is combined with InterProScan features is that InterProScan features do not generalize well across different types of proteins. As an extreme example, consider the possibility that all the positive sequences in the training data contain a hypothetical protein domain A, and all the negative training sequences lack domain A. If all the positive *test* sequences also lack domain A, the model would be fitted to use the lack of domain A as an indication of a negative label, while in reality the test sequences could belong to a completely different protein family wherein the presence or absence of domain A is irrelevant to its producibility. Thus these features could compromise the model when training across different types of proteins, as might have happened in the case of the *A. niger* and *T. reesei* datasets. On the other hand, there is information in the InterProScan features relevant to the problem, and it might be that this information could be harnessed in some other form, e.g. by late fusion of models as opposed to fusion at the feature level prior to model training.

The higher overall performance of the model trained on the *A. niger* data is

probably due to the larger size of the dataset. Interestingly, the linear kernel model trained on the *T. reesei* with amino acid composition gives an AUC of .75 on the *A. niger* data (Table 5.3). This is similar the average AUC achieved by the *A. niger* data when a model is trained using only 50 examples (Figure 5.1), as described in the next section.

5.3 Number of training sequences

The results for experiments with fewer labeled training examples are visualized in Figure 5.1. From top to bottom the rows show average AUC, accuracy, precision, and recall as a function of the number of training examples. Each column shows the results using a different random cross-validation split of the data.

The average performance degraded slowly as the number of training examples decreased for both linear and RBF kernels. No major changes in performance occurred until around 100-60 training examples, when, for Run 1 and Run 2, there was a visible rise in recall and dip in precision for the RBF kernel. The reasons for why the RBF kernel model would start returning many false positives are unclear. Interestingly, the AUC did not suffer in the same way, indicating that while the thresholded classification of the test proteins gave false positives, the spatial relationship between the test points as returned by the classification function remained correct. In either case, it appears that the linear kernel is slightly more stable when using fewer training examples.

In all cross-validation splits the average performance was certainly lower for 10 training examples than it was for 310. However, with around .6-.7 accuracy and .7-.8 AUC, the models trained with 10 examples were still surprisingly useful for discriminating successfully produced proteins. This could in part be due to the small dimensionality of the feature vector, which limits the flexibility of the model and could help allow for the use of 10 training examples without overfitting or degeneracy. It could also be related to the near 50-50 ratio of positive to negative examples maintained at each iteration – using 5 negative and 5 positive examples would be expected to return a better model than 1 negative and 9 positive examples. Finally, the use of the same model parameters might also contribute to stability, as model parameters that generalize well can be difficult to identify with very few training examples.

5.4 TSVM and LapSVM experiments

Tables 5.4, 5.5, 5.6, and 5.7 show the average AUC across cross-validation folds and its standard deviation for combinations of sampling method, number of unlabeled

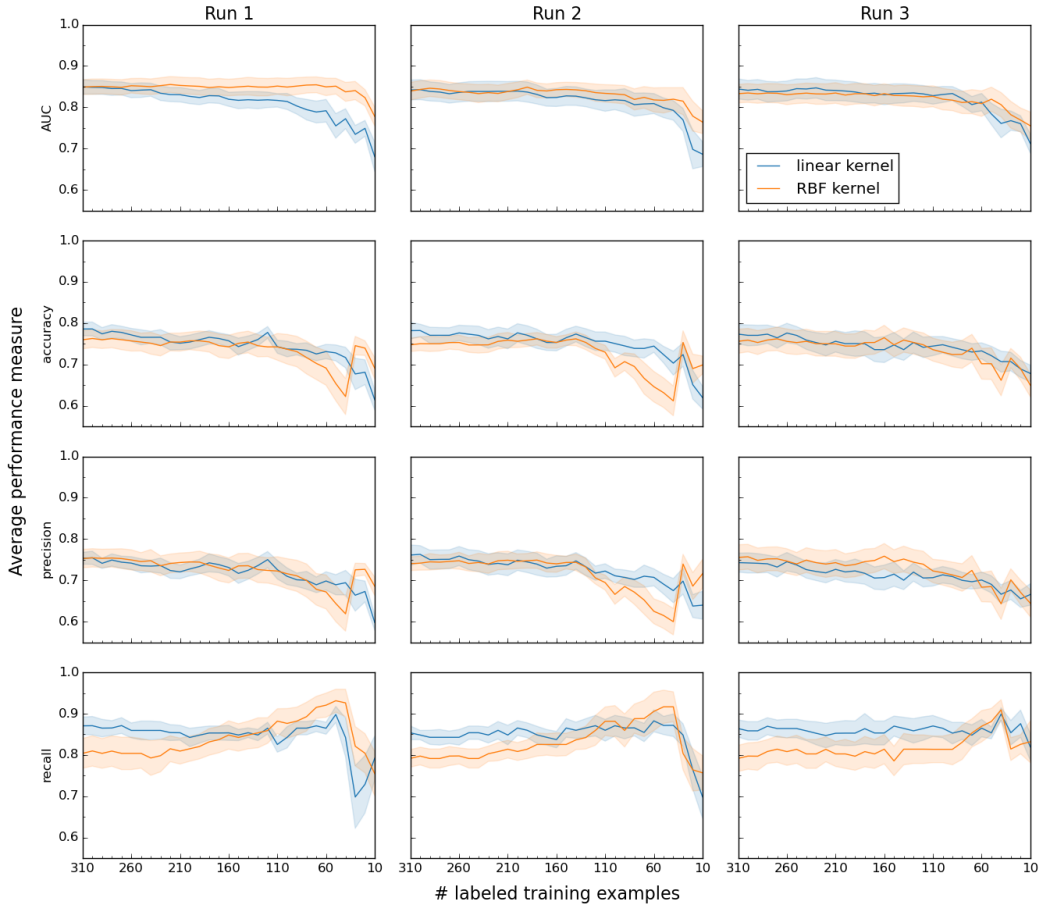


Figure 5.1: Performance measure as a function of the number of labeled training examples. Solid lines show the average across 10 cross-validation folds and the shaded area is the standard error. The blue and orange colors show results using a linear and RBF kernel, respectively. Each column is a different "run" a.k.a. randomized split of the labeled data into cross-validation folds. "AUC" stands for "Area under the curve." See Section 3.6 for a formal definition of performance metrics.

examples, and the labeled dataset used for training. Table 5.4 and Table 5.6 show results using the TSVM model with a linear and RBF kernel, respectively, and Table 5.5 and Table 5.7 show results for the LapSVM model with a linear and RBF kernel, respectively.

Table 5.8 shows the selected C^* parameters for the RBF kernel TSVM for all training conditions using 500 unlabeled sequences, and Table 5.9 shows the selected γ_I parameters for the RBF kernel LapSVM.

Interestingly, in most cases the use of semi-supervised models and inclusion of

Table 5.4: Avg. (std.) AUC performance of TSVM with linear kernel

	A. niger		*T. reesei* combined
	All ex	10 ex	All ex
SVM	.84 (.08)	.64 (.16)	.64 (.22)
random 500	.80 (.08)	.67 (.09)	.68 (.20)
random 2000	.80 (.08)	.69 (.09)	.69 (.19)
close 500	.81 (.08)	.69 (.09)	.81 (.17)
close 2000	.83 (.08)	.68 (.10)	.81 (.17)

Table 5.5: Avg. (std.) AUC performance of LapSVM with linear kernel

	A. niger		*T. reesei* combined
	All ex	10 ex	All ex
SVM (labeled only)	.84 (.08)	.64 (.16)	.64 (.22)
random 500	.83 (.07)	.67 (.18)	.76 (.21)
random 2000	.84 (.07)	.66 (.17)	.66 (.15)
close 500	.83 (.07)	.65 (.17)	.78 (.15)
close 2000	.83 (.07)	.67 (.18)	.72 (.12)

Table 5.6: Avg. (std.) AUC performance of TSVM with RBF kernel

	A. niger		*T. reesei* combined
	All ex	10 ex	All ex
SVM	.84 (.07)	.72 (.16)	.71 (.26)
random 500	.84 (.05)	.73 (.15)	.66 (.20)
random 2000	.83 (.05)	.73 (.13)	.51 (.34)
close 500	.82 (.07)	.73 (.16)	.70 (.16)
close 2000	.83 (.05)	.72 (.15)	.71 (.14)

Table 5.7: Avg. (std.) AUC performance of LapSVM with RBF kernel

	A. niger		*T. reesei* combined
	All ex	10 ex	All ex
SVM (labeled only)	.84 (.07)	.72 (.16)	.71 (.26)
random 500	.83 (.06)	.72 (.13)	.75 (.18)
random 2000	.83 (.07)	.68 (.16)	.76 (.16)
close 500	.83 (.06)	.71 (.14)	.78 (.15)
close 2000	.83 (.07)	.71 (.13)	.78 (.18)

Table 5.8: C^* selected per-fold for TSVM

Dataset \ Fold #	1	2	3	4	5	6	7	8	9	10
A. niger random 500	1000	.001	100	.001	100	1000	.1	10	1	1000
A. niger close 500	1000	.001	1	100	10	1	.01	100	100	1000
T. reesei random 500	.001	1	.001	1	.001					
T. reesei close 500	.001	.001	.001	.001	10					

Table 5.9: γ_I selected per-fold for LapSVM

Dataset \ Fold #	1	2	3	4	5	6	7	8	9	10
A. niger random 500	.005	.005	.5	5	.5	.005	.005	.5	.5	.5
A. niger close 500	.0005	.005	.5	.5	.5	.5	.5	.5	.5	.5
T. reesei random 500	.05	.5	500	500	500					
T. reesei close 500	.05	.5	500	500	500					

unlabeled sequences had a weak effect on the average AUC performance, especially for models trained with the *A. niger* data. The largest differences observed for the *A. niger* model occurred for the linear kernel TSVM (Table 5.4), where there was a decrease from .84 average AUC using all labeled examples to .81 using the 500 closest unlabeled sequences, and an increase from .64 average AUC using 10 labeled examples to .68 and .69 using the closest 500 or 2000 unlabeled sequences. In the TSVM case (Tables 5.4 and 5.6), the semi-supervised *A. niger* models performed 1-4 percentage points better than using only the 10 labeled examples, and in all cases the variance of the performance across cross-validation folds was lower. The decrease in variance is likely due to the use of a larger amount of data used during training.

The lack of a strong effect could be partially contributed to the selection of unlabeled sequences. It could be that the BLAST filtering described in Section 4.5 limited the unlabeled sequences to those that would have minimal effect on the learned decision function. This preprocessing step could also help explain the similarity between results using random unlabeled sequence selection or "close" selection, since both methods sampled from the pool of sequences pre-selected for low BLAST e-value with labeled sequences.

Another explanation would be that the models are selecting semi-supervised model parameters (C^* and γ_I) so that the unlabeled data is more or less ignored. If this were true, we would expect to observe the values closest to 0 selected, but as shown in Tables 5.8 and 5.9, this is not the case. We also note that the selected

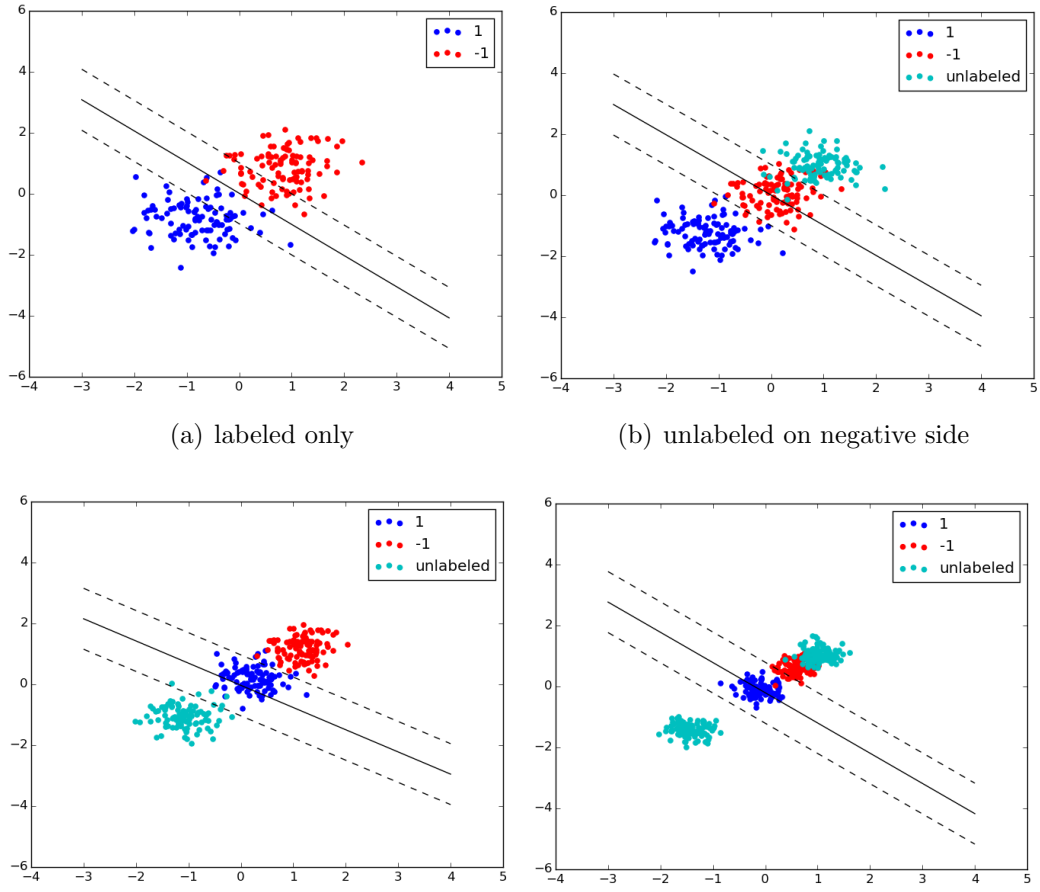
values for C^* and γ_I fluctuate dramatically between folds. This is partly linked to the fluctuations in the C parameter, which varies from .1 to 1000. This instability reflects heterogeneity within the dataset, in the sense that different divisions of the data favor different model parameterizations and vary in achievable classification performance. Similarly, there may be an effect of using semi-supervised methods for some folds and not others, which is masked by taking the performance average.

The decrease in performance in the case of all labeled examples and a linear kernel TSVM (Table 5.4) could be attributed to the linear kernel TSVM's high sensitivity to the distribution of the unlabeled sequences. To prevent the assignment of all unlabeled sequences to one class, the TSVM imposes a constraint on the class composition of the unlabeled sequences (Section 3.4). Through this constraint, the distance of the unlabeled examples to the separating hyperplane and the number of sequences on either side of the hyperplane both have a strong effect on the decision function. This effect is visually illustrated in Figure 5.2 using toy data generated from Gaussian distributions. It is less clear how this TSVM constraint effects models trained using an RBF kernel.

Whether using the TSVM or LapSVM formulation, linear or RBF kernel, or random or close unlabeled sequence selection, we were unable to use 10 labeled sequences combined with semi-supervised methods to recover the performance of the model trained using all labeled sequences (Tables 5.4, 5.5, 5.6, and 5.7). An improvement in performance using semi-supervised methods is not guaranteed, as both the TSVM and LapSVM are developed from assumptions about how the data is distributed. The TSVM assumes that there is a "gap" between classes in the feature space, and that the unlabeled sequences help determine the location of this gap. The LapSVM assumes that examples from the labeled classes are close together in their intrinsic geometries, and that the unlabeled data is useful in providing information about the class geometries. However, if the distribution of the unlabeled examples does not contain information relevant to the classification problem, then semi-supervised methods will not improve the model predictions.

A reasonable explanation for the lack of strong results is thus that the data does not conform to the assumptions of the semi-supervised model – in other words, the class labels of the proteins change in high density areas in amino acid composition space. Figure 5.3 compares the graph Laplacian for the *A. niger* dataset to the graph Laplacian for classifying the digits 2 and 3 from a handwritten digits dataset[1], for which the LapSVM performs well [6]. This square matrix represents the graph information input into the LapSVM. Diagonal elements of the graph Laplacian are the number of neighbors of each point, and the off-diagonal element x_{ij} is 0 if i and j are not neighbors, and -1 if they are neighbors. In the "Digits" dataset the first 206 points are 2, and the second 178 points are 3, and in

[1]Downloadable from http://people.cs.uchicago.edu/ vikass/manifoldregularization.html



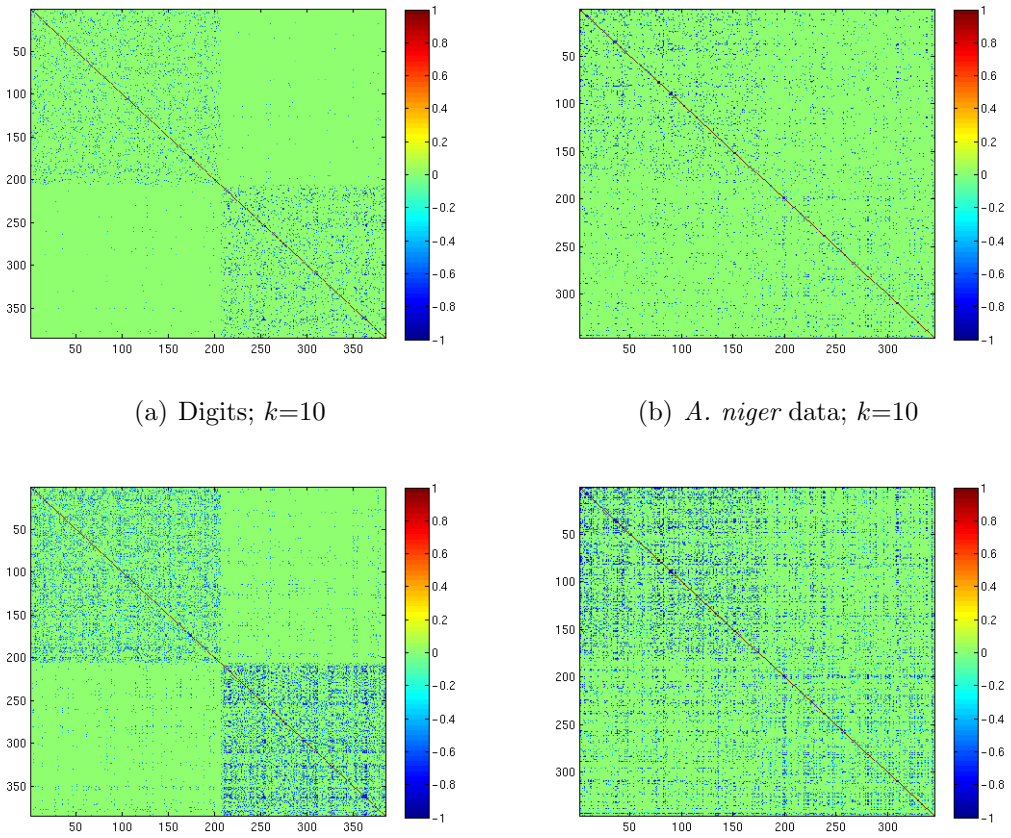
(a) labeled only

(b) unlabeled on negative side

(c) unlabeled on positive side

(d) different distances of unlabeled to hyperplane

Figure 5.2: Illustration of the effects of the distribution of unlabeled data on a linear kernel TSVM using toy data. Positive (red) and negative (blue) points were generated from Gaussian distributions with different means, and unlabeled data points are shown in light blue. The solid black line shows the position of the learned separating hyperplane, and the dotted lines show the location of the margin. Figures (b), (c), and (d) show different situations in which the uneven distribution of labeled examples hurts the model's ability to correctly classify the labeled examples.



(a) Digits; k=10

(b) *A. niger* data; k=10

(c) Digits; k=30

(d) *A. niger* data; k=30

Figure 5.3: Visualization of the graph Laplacian matrices constructed using k-nearest neighbors for $k = 10$ and $k = 30$. (a) and (c) show the matrices for images of the digits 2 and 3, for which the LapSVM performs well, and (b) and (d) show the matrices for the *A. niger* dataset in amino acid composition space, for which the LapSVM does not improve performance over a fully supervised SVM. Blue at position i, j indicates that points i and j are neighbors in the adjacency matrix.

the *A. niger* dataset the first 178 points are positive predictions, and the remaining are negative predictions. The digits dataset has much stronger connectivity within class than between class, which is in contrast to the *A. niger* dataset that has poorer separation between the two classes. Thus the protein labels in the *A. niger* dataset appear to change in high-density areas, which does not conform to the assumptions of the LapSVM.

The most encouraging results occurred for models trained using the combined *T. reesei* dataset. In the case of the linear kernel TSVM, the average AUC increased from .64 (.22 standard deviation [std]) using labeled data only to .81 (.17 std) when using the "close 500" unlabeled sequences (Table 5.4). For the RBF kernel LapSVM, the average AUC increased from .71 (.76 std) to .78 (.15 std), again with the "close 500" unlabeled sequences (Table 5.7). Because of the small size of the *T. reesei* dataset, it is difficult to draw conclusions from the cross-validation performance alone. This finding would be solidified by an analysis of the trained models' generalization performance to the *A. niger* dataset. As opposed to the *A. niger* models, the "close" unlabeled sequence selection generally performs better than the "random" unlabeled sequences selection.

5.5 Enrichment tests

A visual illustration of the most significantly enriched categories for three different models are given in Figures 5.4, 5.5, and 5.6. In all three figures, the rows and columns are composed of all categories that the model found enriched with positive predictions with a p-value of about 0[2]. The coloring in each cell reflects the number of positive predictions that were made for sequences with the EC number and organism taxonomy given by the row and column intersection. The 10-base logarithm was taken of the number of positive predictions to scale the colors for better visibility.

Figure 5.4 visualizes the enrichment test results for the model trained on the *A. niger* data with a polynomial kernel, Figure 5.5 visualizes the same but trained with a RBF kernel, and Figure 5.6 shows the results for the model trained on the *T. reesei* data with a polynomial kernel.

Both *T. reesei* and *A. niger* belong to the *pezizomycotina* sybphylum, so it is sensical that in Figures 5.4, 5.5, and 5.6, this organism taxonomy is highly significant, with many sequences predicted as successfully produced. Especially the *pezizomycotina* and 3.2.1 cell contains a large number of positive predictions, which contains e.g. the major *T. reesei* cellulases, known to be secreted in large quantities. The enrichment of *pezizomycotina* in conjunction with peptidases and

[2]A p-value of 0 was returned by the statistical programming language R, most likely with a very small non-zero rounding error

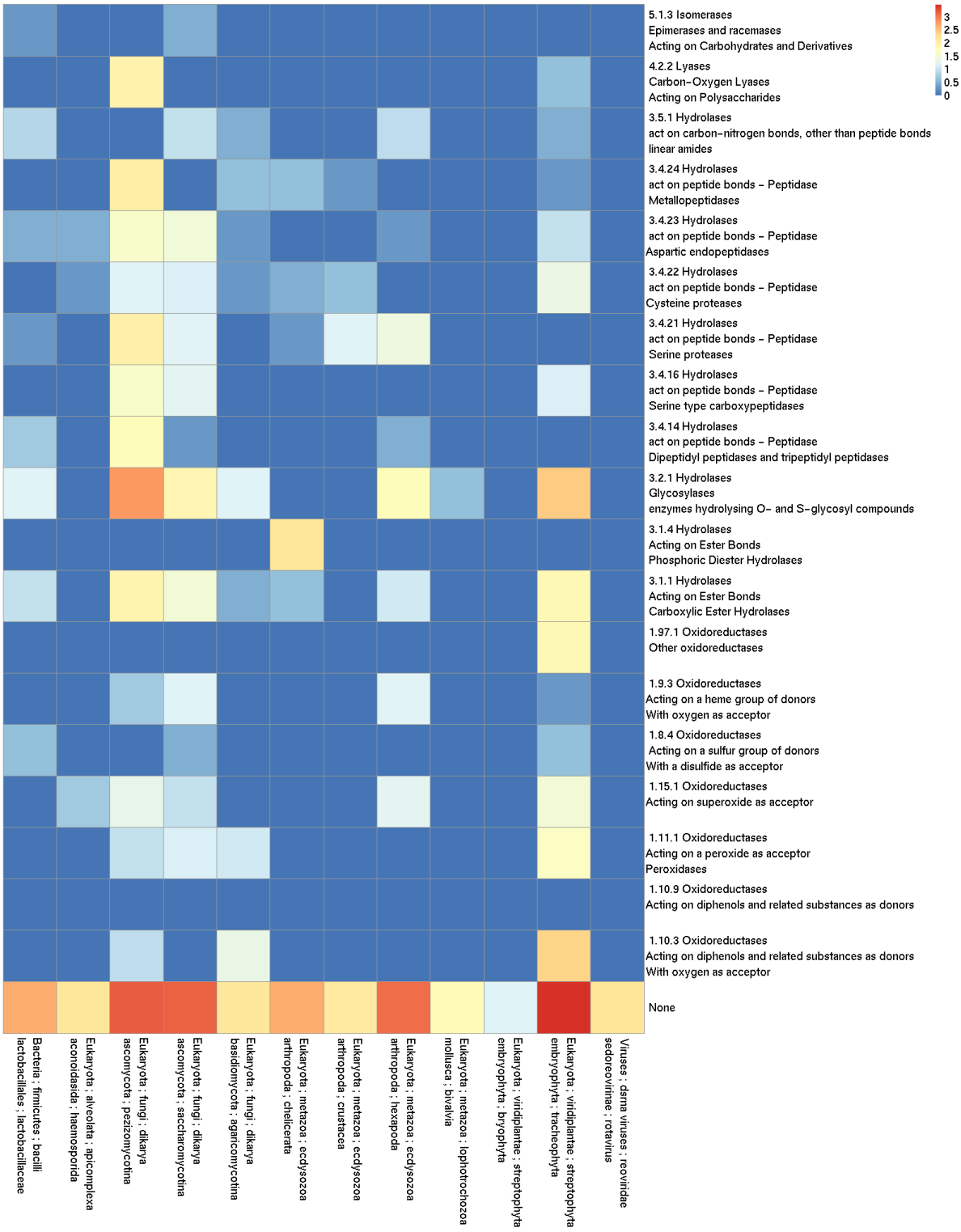


Figure 5.4: EC number/ organism taxonomy enrichment for the polynomial kernel model trained with *A. niger* data. Cell coloring reflects the $\log_{10}$ of the number of predicted successes with the donor organism taxonomy of the cell column and EC number of the cell row. Displayed EC numbers and organism taxonomies were enriched in the predictions of success with a p-value of 0.

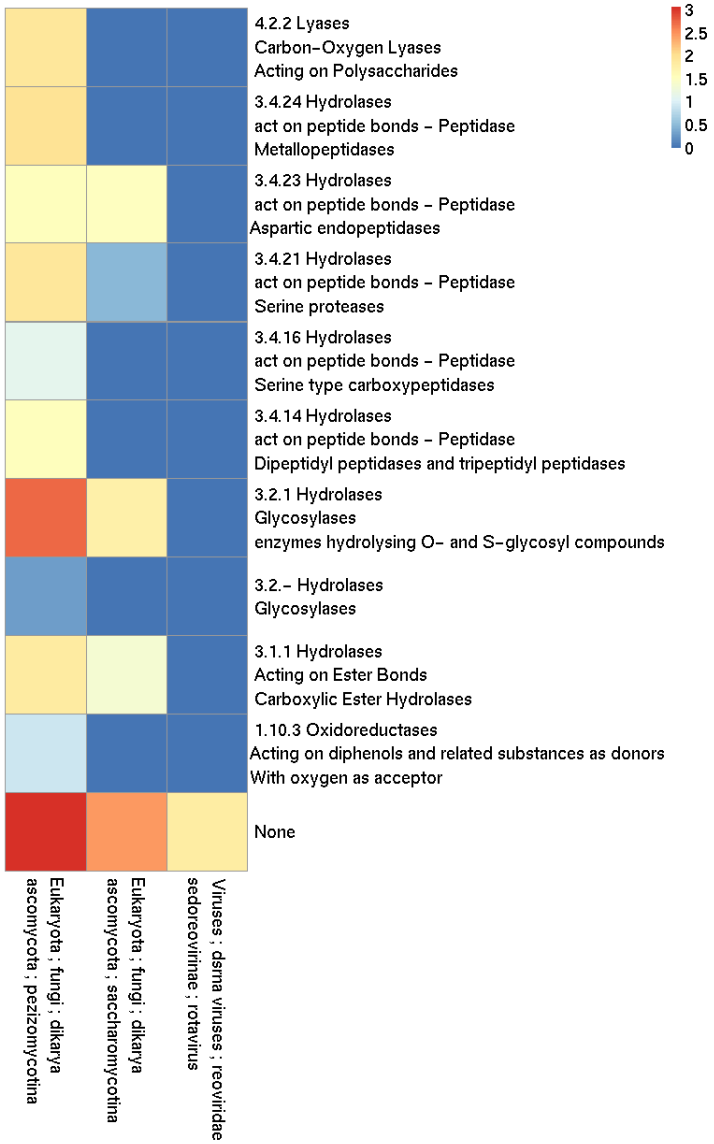


Figure 5.5: EC number/ organism taxonomy enrichment for the RBF kernel model trained with *A. niger* data. Cell coloring reflects the $\log_{10}$ of the number of predicted successes with the donor organism taxonomy of the cell column and EC number of the cell row. Displayed EC numbers and organism taxonomies were enriched in the predictions of success with a p-value of 0.

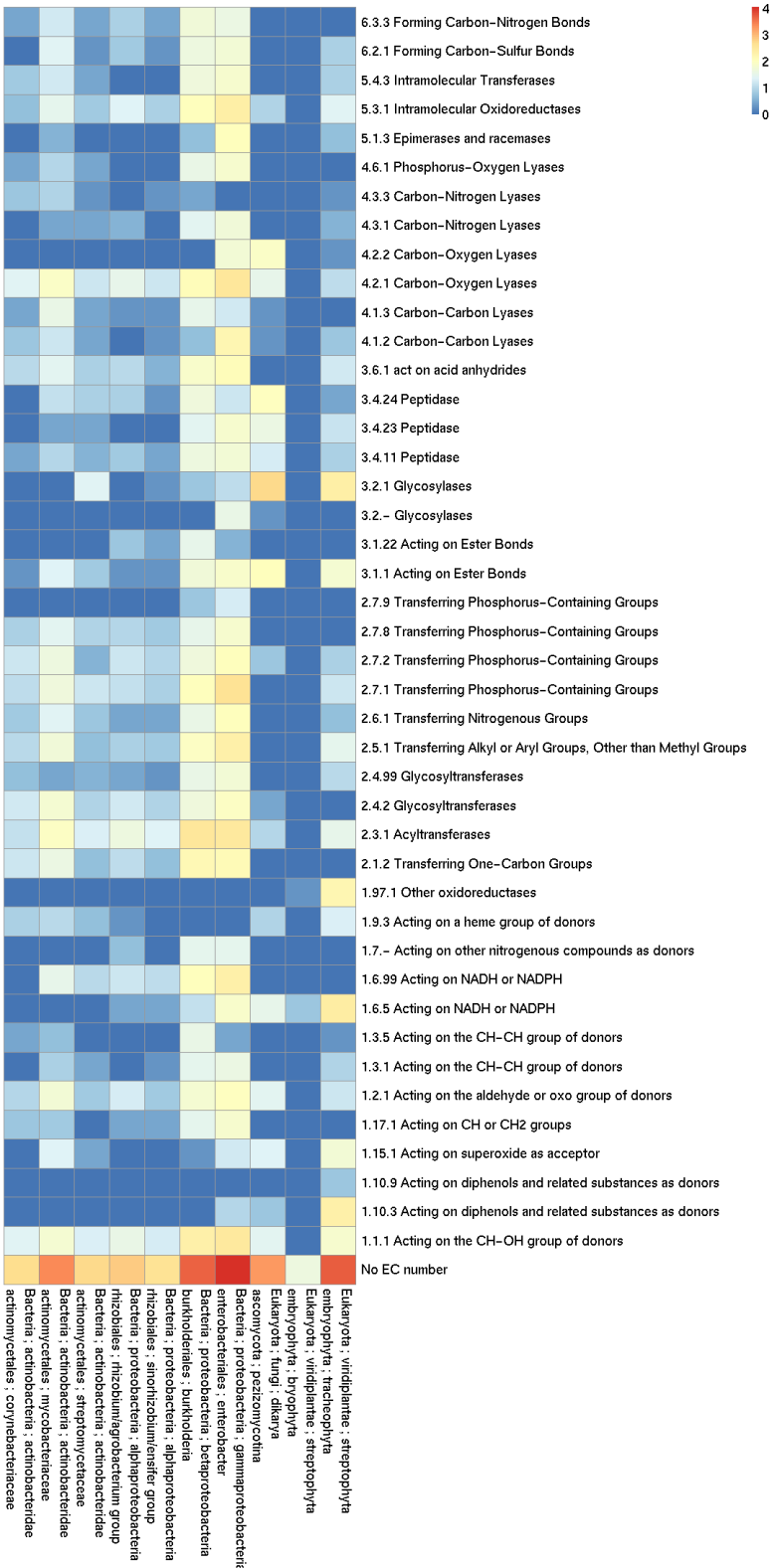


Figure 5.6: EC number/ organism taxonomy enrichment for the polynomial kernel model trained with *T. reesei* data. Cell coloring reflects the $\log_{10}$ of the number of predicted successes with the donor organism taxonomy of the cell column and EC number of the cell row. Displayed EC numbers and organism taxonomies were each enriched in the predictions of success with a p-value of 0.

Table 5.10: Number of positive predictions by dataset and kernel type

	A. niger			*T. reesei*		
	linear	RBF	poly	linear	RBF	poly
A. niger linear	26,956	4,776	20,065	14,941	10,530	12,897
A. niger RBF		6,076	5,584	4,491	3,629	3,956
A. niger poly			33,732	19,494	13,525	16,357
T. reesei linear				95,380	56,365	72,409
T. reesei RBF					59,969	58,762
T. reesei poly						79,476

hydrolases provides a small "sanity check" that the model returns some correct predictions.

It is impossible at this point to draw conclusions about the positive predictions – ultimately only laboratory results will provide an answer of whether the predictions are correct or not. Certainly the significant categories and highly populated cells reflect groups of sequences that have similar amino acid compositions to positively labeled sequences in the training datasets. However, it is unknown to what extent this is related to their producibility upon introduction into filamentous fungi hosts.

In many cases, it is feasible that the predicted groups contain producible sequences. For example, the polynomial *A. niger* model has significant enrichment of *chelicerata* sequences, and two sequences from organisms under the *chelicerata* taxonomy are positive labeled examples in the heterologous *T. reesei* dataset. Figure 5.6 shows that the polynomial kernel *T. reesei* model has positive prediction enrichment in several bacterial taxonomies. Both gram-negative bacteria (including *proteobacteria*) and gram-positive bacteria (including *actinobacteria*) have well-known secretory abilities [21], and it is possible that predictions are related to the protein sequences in bacterial secretion pathways.

The diagonal of Table 5.10 displays the number of positive predictions made by each of the different models, while the off-diagonal elements display the number of positive predictions that overlap between different models, separated by dataset-kernel combinations. Interestingly, Table 5.10 draws attention to the selectivity of different models, which was not considered in the previous analysis of kernel performance on the labeled data (Table 5.2). The RBF kernel was clearly the most selective. In the case of the *A. niger* dataset, the linear model predicted over 4 times as many successes as the RBF model, and the polynomial kernel model, over 5 times as many. The *T. reesei* RBF model predicted about 10 times as many successes as the *A. niger* RBF model. This could be caused by the difference

in the selected γ parameter between the two datasets; γ was set to 100 for the *A. niger* dataset and 10 for the *T. reesei* dataset. The narrow kernel function probably contributed to the high selectivity of the *A. niger* RBF model. The higher selectivity of the RBF kernel vs. other kernels is due to the enclosed shape of its decision function in the input space, as illustrated in Figure 3.3.

There is greater overlap between models trained on the same dataset with different kernels, than between models trained with the same kernel and different datasets. Even though the linear kernel model trained on the *T. reesei* dataset makes 95,380 positive predictions, and the linear kernel model trained on the *A. niger* dataset makes 26,956 positive predictions, only 14,941 of these overlap. It is possible that the sequences with positive predictions by both datasets are more reliable than those with positive predictions from only one model, but currently there is no evidence to this effect. An additional observation from Table 5.10 is that the smaller set of predictions made using the RBF kernel are largely, though not entirely, a subset of the positive predictions made using other kernels.

Lastly, it is important to remember that the selected taxonomy and EC number groups have been selected based on their containing more positive predictions than would be expected given the proportion of the database composed of these groups. There are positive predictions in other groups that occur at non-significant levels.

Chapter 6

Conclusion

We successfully applied an SVM trained with amino acid composition features to predict the successful production of proteins in *A. niger* and *T. reesei*. While this approach was previously applied to *A. niger* [55], we demonstrated its high success rate for proteins produced in a different filamentous fungi host species, *T. reesei*.

We explored the use of family domain information to improve the model through the inclusion of InterProScan features. While somewhat informative for the predictive task, these features do not generalize well across different kinds of proteins and should be excluded in future modeling efforts. Our exploration of different kernels gave the novel result that the RBF kernel predicts far fewer protein sequences to be successfully produced than linear and polynomial kernels. Since the correctness of the majority of predictions made by the model are as of yet unexplored, the decision of whether to trust the polynomial or RBF model is left to the experimentalist.

Experiments using a subset of labeled data for training indicate that one can train a decent predictor, achieving around 70% accuracy on the *A. niger* dataset, by using only 50 labeled training examples. In addition, this training data does not need to be balanced, as indicated by the success of the model trained with the unbalanced *T. reesei* data and applied to the *A. niger* data.

The semi-supervised TSVM and LapSVM methods gave borderline results. While in some cases they gave an inkling of improvement, their use can also harm performance, and thus their application is risky and unnecessarily complicated for training the predictive model. However, if future improvements in the feature representation of the proteins lead to clearer separation according to producibility, then it is likely that TSVM and LapSVM methods could be applied with success.

Based on observations from this thesis, improvements in the protein prediction task are unlikely to result from changes in the model, and efforts to improve in the future should focus on the protein representation. Protein 3D structure is one such unexplored avenue with the potential to improve predictions.

Bibliography

[1] Agostini, F., Cirillo, D., Livi, C. M., Delli Ponti, R., and Tartaglia, G. G. ccSOL omics: a webserver for large-scale prediction of endogenous and heterologous solubility in E. coli. *Bioinformatics 30*, 20 (2014), 2975–2977.

[2] Almquist, J., Cvijovic, M., Hatzimanikatis, V., Nielsen, J., and Jirstrand, M. Kinetic models in industrial biotechnology – Improving cell factory performance. *Metabolic Engineering 24* (2014), 38–60.

[3] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. Basic local alignment search tool. *Journal of molecular biology 215*, 3 (1990), 403–410.

[4] Arvas, M., Pakula, T., Smit, B., Rautio, J., Koivistoinen, H., Jouhten, P., Lindfors, E., Wiebe, M., Penttilä, M., and Saloheimo, M. Correlation of gene expression and protein production rate-a system wide study. *BMC genomics 12*, 1 (2011), 616.

[5] Barton, N. H., Briggs, D. E. G., Eisen, J. A., Goldstein, D. B., and Patel, N. H. General mechanism of eukaryotic protein synthesis, 2007. [Online; accessed December 1, 2015].

[6] Belkin, M., Niyogi, P., and Sindhwani, V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research 7* (2006), 2399–2434.

[7] Birch, J. R., and Racher, A. J. Antibody production. *Advanced Drug Delivery Reviews 58*, 5-6 (2006), 671 – 685. Engineered antibody therapeutics.

[8] Bishop, C. M. *Pattern recognition and machine learning*. springer, 2006.

[9] Boettner, M., Steffens, C., von Mering, C., Bork, P., Stahl, U., and Lang, C. Sequence-based factors influencing the expression of

heterologous genes in the yeast Pichia pastoris-A comparative view on 79 human genes. *Journal of Biotechnology 130*, 1 (2007), 1–10.

[10] Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., and Madden, T. L. Blast+: architecture and applications. *BMC bioinformatics 10*, 1 (2009), 421.

[11] Clark, D. P. *Molecular Biology: Understanding the Genetic Revolution*. Academic Press, Burlington, MA, USA, 2005.

[12] Collobert, R., Sinz, F., Weston, J., and Bottou, L. Large Scale Transductive SVMs. *Journal of Machine Learning Research 7*, 7 (2006), 1687–1712.

[13] Consortium, T. U. Uniprot: a hub for protein information. *Nucleic Acids Research 43*, D1 (2015), D204–D212.

[14] Cortes, C., and Vapnik, V. Support-vector networks. *Machine learning 20*, 3 (1995), 273–297.

[15] Demain, A. L., and Vaishnav, P. Production of recombinant proteins by microbes and higher organisms. *Biotechnology advances 27*, 3 (2009), 297–306.

[16] Driouch, H., Melzer, G., and Wittmann, C. Integration of in vivo and in silico metabolic fluxes for improvement of recombinant protein production. *Metabolic Engineering 14*, 1 (2012), 47–58.

[17] Engelman, D., Steitz, T., and Goldman, A. Identifying nonpolar transbilayer helices in amino acid sequences of membrane proteins. *Annual review of biophysics and biophysical chemistry 15*, 1 (1986), 321–353.

[18] Fisher, H. F. A limiting law relating the size and shape of protein molecules to their composition. *Proceedings of the National Academy of Sciences of the United States of America 51*, 6 (1964), 1285.

[19] Fu, L., Niu, B., Zhu, Z., Wu, S., and Li, W. CD-HIT: Accelerated for clustering the next-generation sequencing data. *Bioinformatics 28*, 23 (2012), 3150–3152.

[20] Genentech, I. First successful laboratory production of human insulin announced. Press Release, September 1978. `http://www.gene.com/media/press-releases/4160/1978-09-06/first-successful-laboratory-production-o`.

[21] Glick, B. R., Pasternak, J. J., and Patten, C. L. *Molecular Biotechnology: Principles and Applications of Recombinant DNA*. ASM Press, Washington, DC, USA, 2010.

[22] Goh, C. S., Lan, N., Douglas, S. M., Wu, B., Echols, N., Smith, A., Milburn, D., Montelione, G. T., Zhao, H., and Gerstein, M. Mining the Structural Genomics Pipeline: Identification of Protein Properties that Affect High-throughput Experimental Analysis. *Journal of Molecular Biology 336*, 1 (2004), 115–130.

[23] Gombert, A. K., and Nielsen, J. Mathematical modelling of metabolism. *Current Opinion in Biotechnology 11*, 2 (2000), 180–186.

[24] Habibi, N., Hashim, S. Z. M., Norouzi, A., and Samian, M. R. A review of machine learning methods to predict the solubility of overexpressed recombinant proteins in escherichia coli. *BMC bioinformatics 15*, 1 (2014), 134.

[25] Harden, V. A., and Lyons, M. Diagnosing and treating genetic diseases - a revolution in progress: Human genetics and medical research, 2010. [Online; accessed December 3, 2015].

[26] Hirose, S., Kawamura, Y., Yokota, K., Kuroita, T., Natsume, T., Komiya, K., Tsutsumi, T., Suwa, Y., Isogai, T., Goshima, N., and Noguchi, T. Statistical analysis of features associated with protein expression/solubility in an in vivo Escherichia coli expression system and a wheat germ cell-free expression system. *Journal of Biochemistry 150*, 1 (2011), 73–81.

[27] Hirose, S., and Noguchi, T. Espresso: A system for estimating protein expression and solubility in protein expression systems. *Proteomics 13*, 9 (2013), 1444–1456.

[28] Idicula-Thomas, S., and Balaji, P. V. Understanding the relationship between the primary structure of proteins and its propensity to be soluble on overexpression in Escherichia coli. *Protein Science*, 5 (2005), 582–592.

[29] Idicula-Thomas, S., Kulkarni, A. J., Kulkarni, B. D., Jayaraman, V. K., and Balaji, P. V. A support vector machine-based method for predicting the propensity of a protein to be soluble or to form inclusion body on overexpression in Escherichia coli. *Bioinformatics 22*, 3 (2006), 278–284.

[30] Ilmén, M., Onnela, M.-L., Klemsdal, S., Keränen, S., and Penttilä, M. Functional analysis of the cellobiohydrolase i promoter of the filamentous fungus trichoderma reesei. *Molecular and General Genetics MGG 253*, 3 (1996), 303–314.

[31] Jones, P., Binns, D., Chang, H.-Y., Fraser, M., Li, W., McAnulla, C., McWilliam, H., Maslen, J., Mitchell, A., Nuka, G., et al. Interproscan 5: genome-scale protein function classification. *Bioinformatics 30*, 9 (2014), 1236–1240.

[32] Klee, E. W., and Sosa, C. P. Computational classification of classically secreted proteins. *Drug Discovery Today 12*, 5-6 (2007), 234–240.

[33] Kyte, J., and Doolittle, R. F. A simple method for displaying the hydropathic character of a protein. *Journal of molecular biology 157*, 1 (1982), 105–132.

[34] Landowski, C. P., Huuskonen, A., Wahl, R., Westerholm-Parvinen, A., Kanerva, A., Hänninen, A.-L., Salovuori, N., Penttilä, M., Natunen, J., Ostermeier, C., et al. Enabling low cost biopharmaceuticals: A systematic approach to delete proteases from a well-known protein production host trichoderma reesei. *PloS one 10*, 8 (2015), e0134723.

[35] Lehninger, A., Nelson, D., and Cox, M. Principles of biochemistry, (1993). *Worth, New York*.

[36] Leslie, C. S., Eskin, E., and Noble, W. S. The spectrum kernel: A string kernel for svm protein classification. In *Pacific symposium on biocomputing* (2002), vol. 7, World Scientific, pp. 566–575.

[37] Li, W., and Godzik, A. Cd-hit: A fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics 22*, 13 (2006), 1658–1659.

[38] Magnan, C. N., Randall, A., and Baldi, P. SOLpro: Accurate sequence-based prediction of protein solubility. *Bioinformatics 25*, 17 (2009), 2200–2207.

[39] Mehlin, C., Boni, E., Buckner, F. S., Engel, L., Feist, T., Gelb, M. H., Haji, L., Kim, D., Liu, C., Mueller, N., Myler, P. J., Reddy, J. T., Sampson, J. N., Subramanian, E., Van Voorhis, W. C., Worthey, E., Zucker, F., and Hol, W. G. J. Heterologous expression

of proteins from Plasmodium falciparum: Results from 1000 genes. *Molecular and Biochemical Parasitology 148*, 2 (2006), 144–160.

[40] Merino, S., and Cherry, J. Progress and challenges in enzyme development for biomass utilization. In *Biofuels*, L. Olsson, Ed., vol. 108 of *Advances in Biochemical Engineering/Biotechnology*. Springer Berlin Heidelberg, 2007, pp. 95–120.

[41] Middelberg, a. P. J. Preparative protein refolding. *Trends in Biotechnology 20*, 10 (2002), 437–443.

[42] Nevalainen, K. M. H., Te'o, V. S. J., and Bergquist, P. L. Heterologous protein expression in filamentous fungi. *Trends in biotechnology 23*, 9 (2005), 468–474.

[43] Niwa, T., Ying, B.-W., Saito, K., Jin, W., Takada, S., Ueda, T., and Taguchi, H. Bimodal protein solubility distribution revealed by an aggregation analysis of the entire ensemble of Escherichia coli proteins. *Proceedings of the National Academy of Sciences of the United States of America 106*, 11 (2009), 4201–4206.

[44] Orth, J. D., Thiele, I., and Palsson, B. Ø. What is flux balance analysis? *Nature biotechnology 28*, 3 (2010), 245–248.

[45] Piippo, M., Lietzén, N., Nevalainen, O. S., Salmi, J., and Nyman, T. A. Pripper: prediction of caspase cleavage sites from whole proteomes. *BMC bioinformatics 11*, 1 (2010), 320.

[46] Rosano, G. L., and Ceccarelli, E. a. Recombinant protein expression in Escherichia coli: Advances and challenges. *Frontiers in Microbiology 5*, APR (2014), 1–17.

[47] Saloheimo, M., and Pakula, T. M. The cargo and the transport system: Secreted proteins and protein secretion in Trichoderma reesei (Hypocrea jecorina). *Microbiology 158*, 1 (2012), 46–57.

[48] Sharp, P. M., and Li, W. H. The codon adaptation index-a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic Acids Research 15*, 3 (1987), 1281–1295.

[49] Shawe-Taylor, J., and Cristianini, N. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[50] Smialowski, P., Doose, G., Torkler, P., Kaufmann, S., and Frishman, D. PROSO II–a new method for protein solubility prediction. *The FEBS journal 279*, 12 (2012), 2192–200.

[51] Song, J., Tan, H., Boyd, S. E., Shen, H., Mahmood, K., Webb, G. I., Akutsu, T., Whisstock, J. C., and Pike, R. N. Bioinformatic Approaches for Predicting Substrates of Proteases. *Journal of Bioinformatics and Computational Biology 09*, 01 (2011), 149–178.

[52] Stefano, M., and Mikhail, B. Laplacian Support Vector Machines Trained in the Primal.pdf. *Journal of Machine Learning Research 12* (2011), 1149–1184.

[53] Sun, A., Peterson, R., Te'o, J., and Nevalainen, H. Expression of the mammalian peptide hormone obestatin in trichoderma reesei. *New biotechnology 33*, 1 (2016), 99–106.

[54] Van Den Berg, B. a., Nijkamp, J. F., Reinders, M. J. T., Wu, L., Pel, H. J., Roubos, J. a., and De Ridder, D. Sequence-based prediction of protein secretion success in Aspergillus niger. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 6282 LNBI* (2010), 3–14.

[55] van den Berg, B. a., Reinders, M. J. T., Hulsman, M., Wu, L., Pel, H. J., Roubos, J. a., and de Ridder, D. Exploring Sequence Characteristics Related to High-Level Production of Secreted Proteins in Aspergillus niger. *PLoS ONE 7*, 10 (2012), 1–11.

[56] Ventura, S., and Villaverde, A. Protein quality in bacterial inclusion bodies. *Trends in Biotechnology 24*, 4 (2006), 179–185.

[57] Ward, O. P. Production of recombinant proteins by filamentous fungi. *Biotechnology Advances 30*, 5 (2012), 1119–1139.

[58] Wee, L. J., Tan, T. W., and Ranganathan, S. Svm-based prediction of caspase substrate cleavage sites. *BMC bioinformatics 7*, Suppl 5 (2006), S14.

[59] Weisstein, E. W. Hyperplane. http://mathworld.wolfram.com/Hyperplane.html. From MathWorld–A Wolfram Web Resource.

[60] Wiebe, M. G. Stable production of recombinant proteins in filamentous fungi - problems and improvements. *Mycologist 17*, 3 (2003), 140–144.

[61] Wilcox, R. R. *Applying Contemporary Statistical Techniques*. Elsevier Inc., 2003.

[62] Yuille, A. L., and Rangarajan, A. The concave-convex procedure. *Neural computation 15*, 4 (2003), 915–936.

Chapter 7

Appendix: InterProScan Features

For each InterProScan feature included in experiments we list the software tool used for extraction, the feature name, and the description from the InterProScan output when available.

Tool name	Feature name	Description
Coils	Coil	
Gene3D	G3DSA:1.10.287.410	
Gene3D	G3DSA:1.10.530.10	
Gene3D	G3DSA:2.120.10.60	
Gene3D	G3DSA:2.130.10.140	
Gene3D	G3DSA:2.160.10.10	
Gene3D	G3DSA:2.40.30.10	
Gene3D	G3DSA:2.80.10.50	
Gene3D	G3DSA:3.30.410.10	
Gene3D	G3DSA:3.30.560.10	
Gene3D	G3DSA:3.30.9.10	
Gene3D	G3DSA:3.40.1090.10	
Gene3D	G3DSA:3.40.630.10	
Gene3D	G3DSA:3.50.30.30	
Gene3D	G3DSA:3.50.50.60	
Gene3D	IPR001128	
Gene3D	IPR001623	
Gene3D	IPR001764	
Gene3D	IPR002772	
Gene3D	IPR008972	
Gene3D	IPR008979	
Gene3D	IPR010259	
Gene3D	IPR011042	
Gene3D	IPR011650	
Gene3D	IPR011990	
Gene3D	IPR012334	
Gene3D	IPR012338	
Gene3D	IPR012340	
Gene3D	IPR012341	
Gene3D	IPR013189	
Gene3D	IPR013319	
Gene3D	IPR013780	
Gene3D	IPR013781	
Gene3D	IPR013783	
Gene3D	IPR013785	
Gene3D	IPR013812	
Gene3D	IPR013830	
Gene3D	IPR014710	
Gene3D	IPR014718	
Gene3D	IPR014766	
Gene3D	IPR015914	
Gene3D	IPR015943	
Gene3D	IPR016040	
Gene3D	IPR016167	
Gene3D	IPR016169	
Gene3D	IPR016288	
Gene3D	IPR017849	
Gene3D	IPR021109	

Gene3D	IPR023413	
Gene3D	IPR023696	
Gene3D	IPR024078	
Gene3D	IPR027414	
Gene3D	IPR027424	
Gene3D	IPR027443	
Gene3D	IPR027477	
Gene3D	IPR029018	
Gene3D	IPR029033	
Gene3D	IPR029039	
Gene3D	IPR029044	
Gene3D	IPR029070	
PANTHER	IPR001139	
PANTHER	IPR001360	
PANTHER	IPR003737	
PANTHER	IPR005880	
PANTHER	IPR007266	
PANTHER	IPR023209	
PANTHER	IPR026892	
PANTHER	IPR029514	
PANTHER	IPR030056	
PANTHER	PTHR10066	
PANTHER	PTHR10209	
PANTHER	PTHR10353:SF36	
PANTHER	PTHR10566	
PANTHER	PTHR10566:SF8	
PANTHER	PTHR10587	
PANTHER	PTHR10587:SF66	
PANTHER	PTHR10728	
PANTHER	PTHR10728:SF29	
PANTHER	PTHR10961	
PANTHER	PTHR10963	
PANTHER	PTHR10963:SF27	
PANTHER	PTHR11010	
PANTHER	PTHR11014	
PANTHER	PTHR11069:SF9	
PANTHER	PTHR11177	
PANTHER	PTHR11177:SF21	
PANTHER	PTHR11407	
PANTHER	PTHR11452	
PANTHER	PTHR11452:SF12	
PANTHER	PTHR11474	
PANTHER	PTHR11474:SF14	
PANTHER	PTHR11552	
PANTHER	PTHR11552:SF62	
PANTHER	PTHR11552:SF67	
PANTHER	PTHR11552:SF70	
PANTHER	PTHR11552:SF75	
PANTHER	PTHR11559	
PANTHER	PTHR11559:SF124	
PANTHER	PTHR11559:SF151	
PANTHER	PTHR11559:SF8	
PANTHER	PTHR11709	
PANTHER	PTHR11709:SF59	
PANTHER	PTHR11731	
PANTHER	PTHR11748	
PANTHER	PTHR11748:SF33	
PANTHER	PTHR11748:SF46	
PANTHER	PTHR11748:SF47	
PANTHER	PTHR11748:SF48	
PANTHER	PTHR11748:SF57	
PANTHER	PTHR11802:SF40	
PANTHER	PTHR11802:SF5	
PANTHER	PTHR12147	
PANTHER	PTHR12147:SF16	
PANTHER	PTHR12993:SF11	
PANTHER	PTHR13593	
PANTHER	PTHR13683:SF255	
PANTHER	PTHR13683:SF75	

PANTHER	PTHR13878	
PANTHER	PTHR14218	
PANTHER	PTHR14218:SF13	
PANTHER	PTHR14218:SF17	
PANTHER	PTHR16631	
PANTHER	PTHR16631:SF4	
PANTHER	PTHR20963	
PANTHER	PTHR21493	
PANTHER	PTHR22600	
PANTHER	PTHR22762	
PANTHER	PTHR22912	
PANTHER	PTHR22912:SF137	
PANTHER	PTHR22912:SF48	
PANTHER	PTHR22925:SF7	
PANTHER	PTHR22953	
PANTHER	PTHR23267	
PANTHER	PTHR23267:SF135	
PANTHER	PTHR24286	
PANTHER	PTHR24287	
PANTHER	PTHR24305	
PANTHER	PTHR30175	
PANTHER	PTHR30175:SF0	
PANTHER	PTHR30620:SF12	
PANTHER	PTHR30620:SF8	
PANTHER	PTHR31018	
PANTHER	PTHR31297	
PANTHER	PTHR31297:SF12	
PANTHER	PTHR31490	
PANTHER	PTHR31490:SF7	
PANTHER	PTHR31683	
PANTHER	PTHR31736	
PANTHER	PTHR31736:SF2	
PANTHER	PTHR31736:SF4	
PANTHER	PTHR31884	
PANTHER	PTHR31884:SF2	
PANTHER	PTHR31939	
PANTHER	PTHR31956	
PANTHER	PTHR31987	
PANTHER	PTHR31987:SF1	
PANTHER	PTHR32438	
PANTHER	PTHR32438:SF8	
PIRSF	IPR002056	
PIRSF	IPR005152	
PIRSF	IPR008384	
PIRSF	IPR011395	
PIRSF	IPR012132	
PIRSF	IPR016274	
PIRSF	IPR016840	
PIRSF	IPR017168	
PRINTS	IPR000103	Pyridine nucleotide disulphide reductase class-II signature
PRINTS	IPR000165	Glycosyl hydrolase family 15 signature
PRINTS	IPR000447	FAD-dependent glycerol-3-phosphate dehydrogenase family signature
PRINTS	IPR000786	Green fluorescent protein signature
PRINTS	IPR000805	Glycosyl hydrolase family 26 signature
PRINTS	IPR000974	Lysozyme signature
PRINTS	IPR001000	Glycosyl hydrolase family 10 signature
PRINTS	IPR001094	Flavodoxin signature
PRINTS	IPR001137	Glycosyl hydrolase family 11 signature
PRINTS	IPR001382	Glycosyl hydrolase family 47 signature
PRINTS	IPR001709	Flavoprotein pyridine nucleotide cytochrome reductase signature
PRINTS	IPR001722	Glycosyl hydrolase family 7 signature
PRINTS	IPR001944	Glycosyl hydrolase family 35 signature
PRINTS	IPR002241	Glycosyl hydrolase family 27 signature
PRINTS	IPR002347	Glucose/ribitol dehydrogenase family signature
PRINTS	IPR002401	E-class P450 group I signature
PRINTS	IPR002402	Group II E-class P450 signature
PRINTS	IPR002403	E-class P450 group IV signature

PRINTS	IPR002974	CYP52 P450 protein signature
PRINTS	IPR011150	Cutinase signature
PRINTS	IPR015500	Subtilisin serine protease family (S8) signature
PRINTS	IPR025705	Glycosyl hydrolase family 20 signature
PRINTS	PR00368	FAD-dependent pyridine nucleotide reductase signature
PRINTS	PR00420	Aromatic-ring hydroxylase (flavoprotein monooxygenase) signature
Pfam	IPR000073	Alpha/beta hydrolase family
Pfam	IPR000101	Gamma-glutamyltranspeptidase
Pfam	IPR000120	Amidase
Pfam	IPR000250	Peptidase A4 family
Pfam	IPR000322	Glycosyl hydrolases family 31
Pfam	IPR000675	Cutinase
Pfam	IPR000719	Protein kinase domain
Pfam	IPR000757	Glycosyl hydrolases family 16
Pfam	IPR001117	Multicopper oxidase
Pfam	IPR001199	Cytochrome b5-like Heme/Steroid binding domain
Pfam	IPR001223	Glycosyl hydrolases family 18
Pfam	IPR001338	Fungal hydrophobin
Pfam	IPR001461	Eukaryotic aspartyl protease
Pfam	IPR001547	Cellulase (glycosyl hydrolase family 5)
Pfam	IPR001563	Serine carboxypeptidase
Pfam	IPR001568	Ribonuclease T2 family
Pfam	IPR001842	Fungalysin metallopeptidase (M36)
Pfam	IPR002018	Carboxylesterase family
Pfam	IPR002123	Acyltransferase
Pfam	IPR002198	short chain dehydrogenase
Pfam	IPR002227	Common central domain of tyrosinase
Pfam	IPR002472	Palmitoyl protein thioesterase
Pfam	IPR002594	Glycosyl hydrolase family 12
Pfam	IPR002820	MoaC family
Pfam	IPR002921	Lipase (class 3)
Pfam	IPR002933	Peptidase family M20/M25/M40
Pfam	IPR003137	PA domain
Pfam	IPR003961	Fibronectin type III domain
Pfam	IPR004843	Calcineurin-like phosphoesterase
Pfam	IPR005103	Glycosyl hydrolase family 61
Pfam	IPR005123	2OG-Fe(II) oxygenase superfamily
Pfam	IPR005154	Glycosyl hydrolase family 67 N-terminus
Pfam	IPR005193	Glycosyl hydrolase family 62
Pfam	IPR005197	Glycosyl hydrolase family 71
Pfam	IPR006034	Asparaginase
Pfam	IPR006076	FAD dependent oxidoreductase
Pfam	IPR006094	FAD binding domain
Pfam	IPR006102	Glycosyl hydrolases family 2
Pfam	IPR006357	Haloacid dehalogenase-like hydrolase
Pfam	IPR006710	Glycosyl hydrolases family 43
Pfam	IPR007117	Pollen allergen
Pfam	IPR007312	Phosphoesterase family
Pfam	IPR007484	Peptidase family M28
Pfam	IPR007867	GMC oxidoreductase
Pfam	IPR007934	Alpha-L-arabinofuranosidase B (ABFB) domain
Pfam	IPR008758	Serine carboxypeptidase S28
Pfam	IPR008902	Bacterial alpha-L-rhamnosidase
Pfam	IPR009939	Fungal chitosanase of glycosyl hydrolase group 75
Pfam	IPR010435	Fn3-like domain
Pfam	IPR010720	Alpha-L-arabinofuranosidase C-terminus
Pfam	IPR011099	Glycosyl hydrolase family 67 C-terminus
Pfam	IPR011100	Glycosyl hydrolase family 67 middle domain
Pfam	IPR011118	Tannase and feruloyl esterase
Pfam	IPR011584	Green fluorescent protein
Pfam	IPR011659	WD40-like Beta Propeller Repeat
Pfam	IPR011706	Multicopper oxidase
Pfam	IPR011707	Multicopper oxidase
Pfam	IPR012946	X8 domain
Pfam	IPR012951	Berberine and berberine like
Pfam	IPR013106	Immunoglobulin V-set domain
Pfam	IPR013148	Glycosyl hydrolases family 32 N-terminal domain
Pfam	IPR014870	Domain of unknown function (DUF1793)

Pfam	IPR015289	Alpha-L-arabinofuranosidase B, catalytic
Pfam	IPR015366	Pro-kumamolisin, activation domain
Pfam	IPR015883	Glycosyl hydrolase family 20, catalytic domain
Pfam	IPR018466	Ser-Thr-rich glycosyl-phosphatidyl-inositol-anchored membrane family
Pfam	IPR018803	Putative stress-responsive nuclear envelope protein
Pfam	IPR018954	Beta-galactosidase, domain 2
Pfam	IPR019623	Chaperone for protein-folding within the ER, fungal
Pfam	IPR020683	Ankyrin repeats (3 copies)
Pfam	IPR021054	Hydrophobic surface binding protein A
Pfam	IPR021986	Spherulation-specific family 4
Pfam	IPR023753	Pyridine nucleotide-disulphide oxidoreductase
Pfam	IPR025300	Beta-galactosidase jelly roll domain
Pfam	IPR025733	Iron/zinc purple acid phosphatase-like protein C
Pfam	IPR025972	Beta-galactosidase, domain 3
Pfam	IPR026891	Fibronectin type III-like domain
Pfam	IPR026992	non-haem dioxygenase in morphine synthesis N-terminal
Pfam	IPR029411	Polysaccharide lyase family 4, domain III
Pfam	IPR029413	Polysaccharide lyase family 4, domain II
Pfam	IPR031330	Glycosyl hydrolases family 35
Pfam	PF12708	Pectate lyase superfamily protein
Pfam	PF13472	GDSL-like Lipase/Acylhydrolase family
Pfam	PF16335	Domain of unknown function (DUF4965)
Pfam	PF16499	Alpha galactosidase A
Pfam	PF16656	Purple acid Phosphatase, N-terminal domain
Pfam	PF16862	Glycosyl hydrolase family 79 C-terminal beta domain
Pfam	PF17111	Fungal N-terminal domain of STAND proteins
Phobius	CYTOPLASMIC_DOMAIN	Region of a membrane-bound protein predicted to be outside the membrane, in the cytoplasm.
Phobius	NON_CYTOPLASMIC_DOMAIN	Region of a membrane-bound protein predicted to be outside the membrane, in the extracellular region.
Phobius	SIGNAL_PEPTIDE	Signal peptide region
Phobius	SIGNAL_PEPTIDE_C_REGION	C-terminal region of a signal peptide.
Phobius	SIGNAL_PEPTIDE_H_REGION	Hydrophobic region of a signal peptide.
Phobius	SIGNAL_PEPTIDE_N_REGION	N-terminal region of a signal peptide.
Phobius	TRANSMEMBRANE	Region of a membrane-bound protein predicted to be embedded in the membrane.
ProSitePatterns	IPR000111	Alpha-galactosidase signature.
ProSitePatterns	IPR000172	GMC oxidoreductases signature 2.
ProSitePatterns	IPR000254	CBM1 (carbohydrate binding type-1) domain signature.
ProSitePatterns	IPR000334	Glycosyl hydrolases family 45 active site.
ProSitePatterns	IPR000560	Histidine acid phosphatases active site signature.
ProSitePatterns	IPR000743	Polygalacturonase active site.
ProSitePatterns	IPR001261	ArgE / dapE / ACY1 / CPG2 / yscS family signature 1.
ProSitePatterns	IPR001524	Glycosyl hydrolases family 6 signature 1.
ProSitePatterns	IPR001579	Chitinases family 18 active site.
ProSitePatterns	IPR001969	Eukaryotic and viral aspartyl proteases active site.
ProSitePatterns	IPR002355	Multicopper oxidases signature 1.
ProSitePatterns	IPR017972	Cytochrome P450 cysteine heme-iron ligand signature.
ProSitePatterns	IPR018040	Pectinesterase signature 2.
ProSitePatterns	IPR018087	Glycosyl hydrolases family 5 signature.
ProSitePatterns	IPR018120	Glycosyl hydrolases family 1 active site.
ProSitePatterns	IPR018188	Ribonuclease T2 family histidine active site 2.
ProSitePatterns	IPR018202	Serine carboxypeptidases, histidine active site.
ProSitePatterns	IPR018208	Glycosyl hydrolases family 11 active site signature 1.
ProSitePatterns	IPR018357	Hexapeptide-repeat containing-transferases signature.
ProSitePatterns	IPR018371	Chitin recognition or binding domain signature.
ProSitePatterns	IPR019799	Alpha-lactalbumin / lysozyme C signature.
ProSitePatterns	IPR019819	Carboxylesterases type-B signature 2.
ProSitePatterns	IPR019826	Carboxylesterases type-B serine active site.
ProSitePatterns	IPR020855	Arginase family signature.
ProSitePatterns	IPR020904	Short-chain dehydrogenases/reductases family signature.
ProSitePatterns	IPR022398	Serine proteases, subtilase family, histidine active site.
ProSitePatterns	IPR023827	Serine proteases, subtilase family, aspartic acid active site.
ProSitePatterns	IPR023828	Serine proteases, subtilase family, serine active site.
ProSitePatterns	IPR031158	Glycosyl hydrolases family 10 active site.
ProSiteProfiles	IPR002509	NodB homology domain profile.
ProSiteProfiles	IPR006035	Arginase family profile.
ProSiteProfiles	IPR007112	Expansin, family-45 endoglucanase-like domain profile.

ProSiteProfiles	IPR008254	Flavodoxin-like domain profile.
ProSiteProfiles	IPR010513	KEN domain profile.
ProSiteProfiles	IPR019791	Animal heme peroxidase superfamily profile.
ProSiteProfiles	IPR030400	Sedolisin domain profile.
ProSiteProfiles	PS51257	Prokaryotic membrane lipoprotein lipid attachment site profile.
SMART	IPR000772	Ricin-type beta-trefoil
SMART	IPR000834	
SMART	IPR001362	Glycosyl hydrolases family 32
SMART	IPR001916	Alpha-lactalbumin / lysozyme C
SMART	IPR002022	
SMART	IPR002044	Starch binding domain
SMART	IPR002110	ankyrin repeats
SMART	IPR002642	Cytoplasmic phospholipase A2, catalytic subunit
SMART	IPR003172	Domain involved in innate immunity and lipid metabolism.
SMART	IPR003598	Immunoglobulin C-2 Type
SMART	IPR003599	Immunoglobulin
SMART	IPR006045	Cupin
SMART	IPR006626	Parallel beta-helix repeats
SMART	IPR011583	
SMART	IPR018391	beta-propeller repeat
SUPERFAMILY	IPR000028	
SUPERFAMILY	IPR000209	
SUPERFAMILY	IPR001002	
SUPERFAMILY	IPR003010	
SUPERFAMILY	IPR007110	
SUPERFAMILY	IPR008922	
SUPERFAMILY	IPR008928	
SUPERFAMILY	IPR008963	
SUPERFAMILY	IPR009009	
SUPERFAMILY	IPR009011	
SUPERFAMILY	IPR009017	
SUPERFAMILY	IPR009020	
SUPERFAMILY	IPR010255	
SUPERFAMILY	IPR010636	
SUPERFAMILY	IPR011009	
SUPERFAMILY	IPR011013	
SUPERFAMILY	IPR011045	
SUPERFAMILY	IPR011050	
SUPERFAMILY	IPR011330	
SUPERFAMILY	IPR012349	
SUPERFAMILY	IPR013320	
SUPERFAMILY	IPR013784	
SUPERFAMILY	IPR016035	
SUPERFAMILY	IPR016087	
SUPERFAMILY	IPR016164	
SUPERFAMILY	IPR016166	
SUPERFAMILY	IPR016191	
SUPERFAMILY	IPR017850	
SUPERFAMILY	IPR017853	
SUPERFAMILY	IPR017946	
SUPERFAMILY	IPR018392	
SUPERFAMILY	IPR023214	
SUPERFAMILY	IPR023296	
SUPERFAMILY	IPR023346	
SUPERFAMILY	IPR023392	
SUPERFAMILY	IPR023631	
SUPERFAMILY	IPR029052	
SUPERFAMILY	IPR029058	
SUPERFAMILY	IPR029069	
SUPERFAMILY	SSF110019	
SUPERFAMILY	SSF110296	
SUPERFAMILY	SSF51011	
SUPERFAMILY	SSF51197	
SUPERFAMILY	SSF51735	
SUPERFAMILY	SSF51905	
SUPERFAMILY	SSF51971	
SUPERFAMILY	SSF52025	
SUPERFAMILY	SSF52058	

SUPERFAMILY	SSF52768	
SUPERFAMILY	SSF53187	
SUPERFAMILY	SSF54373	
SUPERFAMILY	SSF55486	
SUPERFAMILY	SSF63829	
SUPERFAMILY	SSF82171	