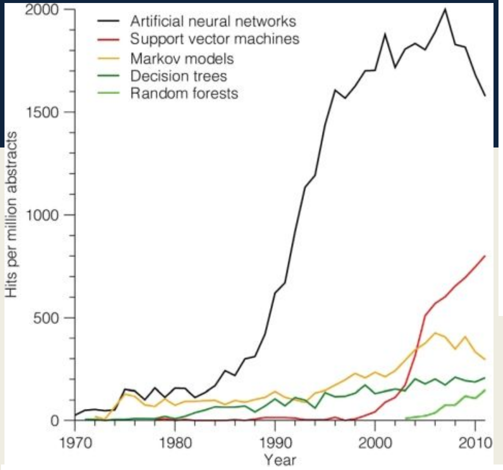
Predicting NRPS Substrates using a multi-Kernel SVM

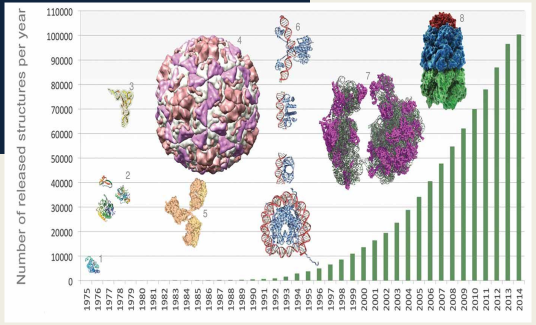
MS229s - Final Project

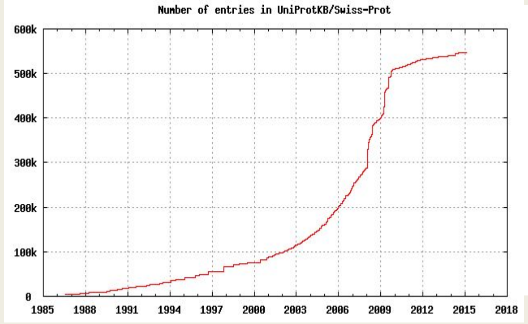
Brian Liu

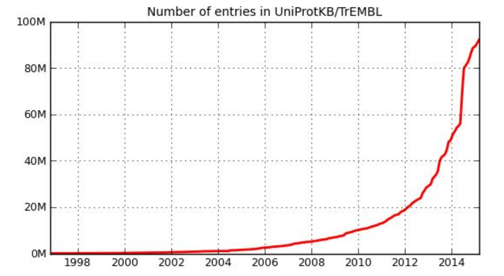
Motivation

- Kernel methods (SVM,PCA,etc.) are widely successful in other fields of applied statistics
- SVM is an underutilized tool in bioinformatics compared to other fields that (was) gaining popularity, so I hope to introduce another way to view problems besides the HMM / Bayesian models we have looked mostly at.
- Kernel methods have successfully been applied in characterizing Protein interactions
  - Also a domain with many open questions whose progress has slowed? recently
  - Also a very complex domain
- NRPS (non-ribosomal peptide synthesases) - resemble proteins in structure (chain of AAs)
  - We can apply same machine learning classifications techniques as we do for protein interactions, due to similarity in structure and function





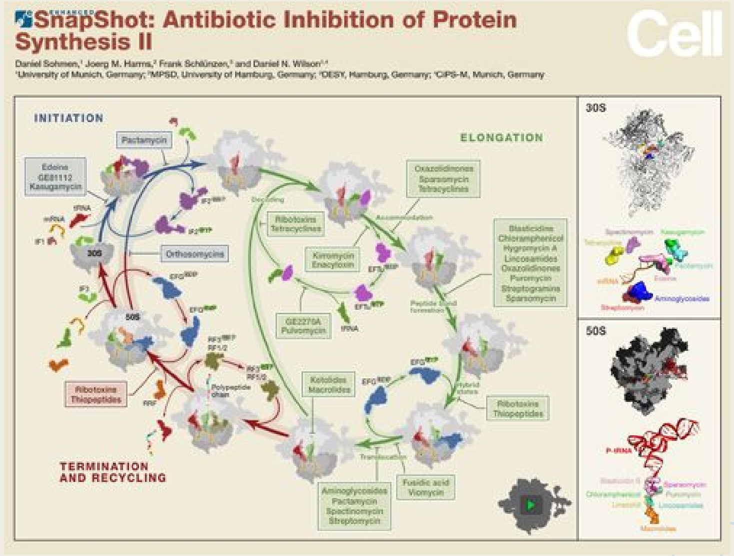




- Images from:Machine learning in bioinformatics: a brief survey and recommendations for practitioners. (Bhaskar)
- http://web.expasy.org/docs/relnotes/relstat.html
- http://www.ebi.ac.uk/uniprot/TrEMBLstats

Non-Ribosomal Peptide Synthetases

- “Large” multimodule catalysts (400-600 AA long) that assemble peptides
  - Mainly utilized by fungi and bacteria
  - Produce pharmacologically important compounds: antibiotics, immuno-suppressors, pigments, toxins
- Alternate pathway that allows production of polypeptides besides traditional DNA->mRNA,ribosome->polypeptide, Instead: NRPS+substrate ->NRP
- Each NRP has a A, PCP, and C domain
- Studies have shown that the A-domain content is highly correlated to substrate that gets bound and hence the product that gets formed
  - **Predictive, structure-based model of amino acid recognition by nonribosomal peptide synthetase adenylation domains.** (Challis et al. 2000)
  - **Substrate recognition by nonribosomal peptide synthetase multi-enzymes** (Lautra 2004)
  - **Protein Engineering by Evolutionary Methods** (Lutz 2002)


SnapShot: Antibiotic Inhibition of Protein Synthesis II
Cell
Daniel Sohmen, Joerg M. Harms, Frank Schlünzen, and Daniel N. Wilson
University of Munich, Germany; MPSD, University of Hamburg, Germany; DESY, Hamburg, Germany; CIPS-M, Munich, Germany
INITIATION
ELONGATION
TERMINATION AND RECYCLING
30S
50S
Pactamycin
Edeine GE81112 Kasugamycin
Orthosomycins
Oxazolidinones Sparsomycin Tetracyclines
Ribotoxins Tetracyclines
Kirromycin Enacyloxin
GE2270A Pulvomycin
Ketolides Macrolides
Ribotoxins Thiopeptides
Aminoglycosides Pactamycin Spectinomycin Streptomycin
Fusidic acid Viomycin
Blasticidin Chloramphenicol Hygromycin A Lincosamides Oxazolidinones Puromycin Streptogramins Sparsomycin
P-tRNA

- **Project: Attempt to predict binding substrates to NRPs using A-domain information and a Multiple Kernel SVM classification approach.**
  - e.g. Given a known A-domain represented as a sequence of AAs, e.g. **input: TYRELDEKSNQLARFLRKK….. (~550 AA long)**
  - Predict the binding substrate class, **output: P (P-glycoprotein) - 31 possible binding substrate classes**

Previous Work

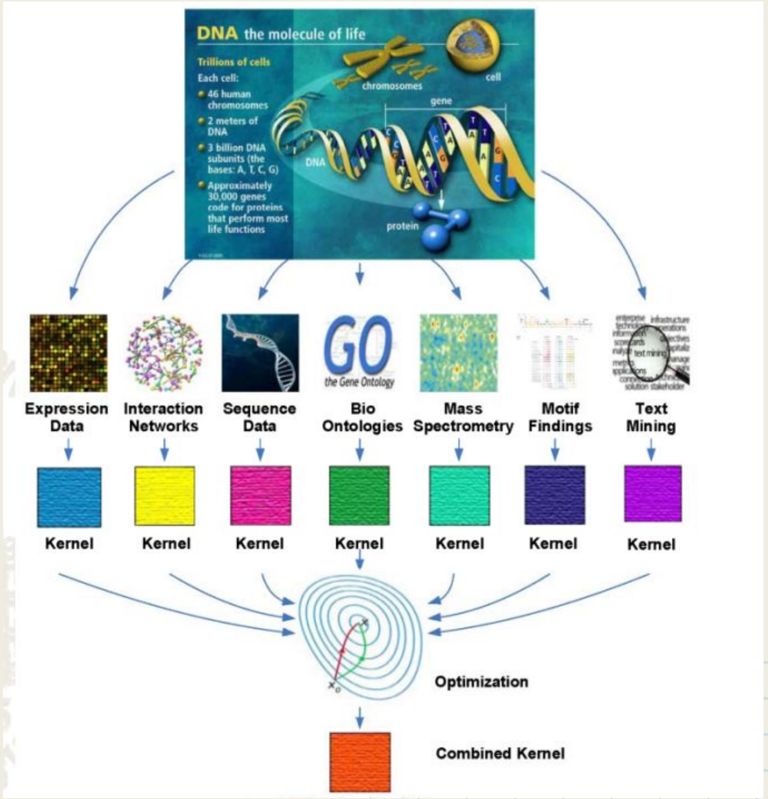
- **Specificity prediction of adenylation domains in nonribosomal peptide synthetases (NRPS) using transductive support vector machines (TSVMs) (2005 Rausch et. al**)
  - Uses TSVMs (Transductive Support Vector Machines) - SVM adapted for sparse data / data with many missing features
- **NRPSsp: Non-Ribosomal Peptide Synthase substrate predictor (2011 Prieto et. al)**
  - Builds an HMM profile per substrate based on AA sequence.
- **Classification of Non-Ribosomal Peptide Syntheses with FeedForward Neural Networks. (2013 Søndergaard, et. al)**
  - Neural Network Approach - builds a model based on k-mer domain content (20^k first layer input neuron!!)
- **Predicting substrate specificity of adenylation domains of nonribosomal peptide synthetases and other protein properties by latent semantic indexing**
  - Text-based LSI approach, similar to SVM, using sequence information, not combining information from other domains
- No one has taken a **multiple kernel SVM** approach to this problem.
- Approach: Build multiple kernels, similar to a method described in **Kernel methods for predicting protein–protein interactions (2004 Ben-hur et. al)**
  - Attempt to hand build multiple kernels / feature space representations to run SVM in to "learn" NRP substrate binding affinities

Project: Attempt to predict binding substrates to NRPs using A-domain information and a Multiple Kernel SVM classification approach.

- **To my knowledge, a multiple kernel approach has not yet been attempted towards solving this problem.**
- **Use same dataset as previous studies / cross validation methods to compare classifier**

Multiple Kernel / Feature Space Learning Approach

- Quick Aside: In my experience I have used **single kernels**
- Basic idea: **kernel matrix** is a matrix that describes similarity between two vectors in a feature space.
- Can be created by ALMOST ANYTHING that describes similarity between two inputs
- We can then apply a multitude of machine learning algorithms (SVM, PCA, etc.), by replacing **XtX(normalized variance covariance matrix)** matrix with **Kernel Matrix**
- Transforms the data into a more informative feature space
- The key idea is that multiple kernels can represent different perspectives of the data, and combined we get a holistic "more accurate" view **(reliant on training data)**
    - A kernel for sequence data
    - A kernel representing bio / chemical properties of an amino acid / peptide.
    - A kernel incorporating outside database protein interaction knowledge
    - **Problem: How to combine kernels?**



Methodology

- Attempt to create a multiple kernel SVM predictor for domain A substrates
  - Find meaningful feature space mapping - replacement of AA’s with some sort of “class”
    - Capturing relevant physio / chemical properties of amino acid.
  - Kernels based on AA sequence matching
    - Bag of words Model, K-mer Model, Alignment Kernel
  - UnitProt Kernel (self constructed using Uniprot DB)
    - Modified data scraper in python to build feature vector in 54 dimensional space of protein annotations
- After building new kernels / feature space representations, run SVM and use k=5 cross fold validation to estimate classification error (same as original paper) on individual kernels, then combine them and evaluate methodology
- Tools used: svm-suite, svm-lite, sci-py, BeautifulSoup(scraping), ggplot2, caret
  - used for sequence alignment, kernel creation, data visualization, SVM training and testing

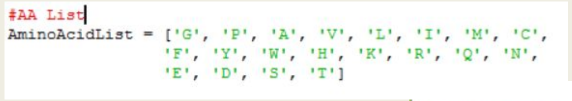
Methodology

Project: Predict binding substrates to NRPs using A-domain content and a multiple Kernel SVM.

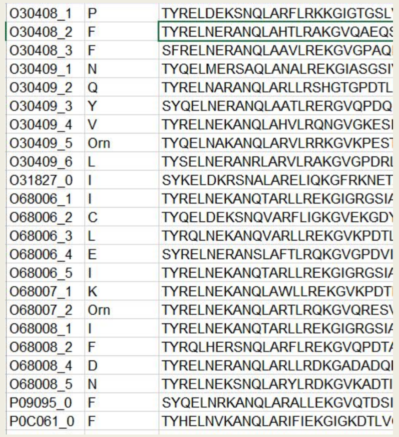
- **Can bind to 31 known substrate classes**
- **Represented as a chain of amino acids (20 total)**
- **Use same dataset as previous studies / cross validation methods to compare classifier**

- Original Data representation:

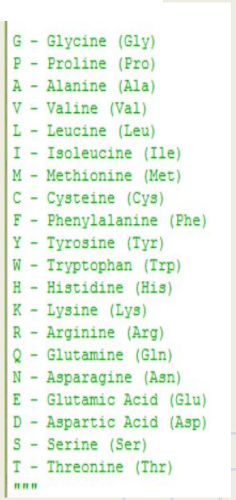
  "TYRELDEKSNQLARFLRKKGIGTGSLVGTLLDRSLDMIVGILGVLKAGGAFVPIDPELPAERI AYMLTHSRVPLVVTQNHLRAKVTTPTETIDINTAVIGEESRAPIESLNQPHDLFYIIYTSGTTG QPKGVMLEHRNMANLMHFTFDQTNIAFHEKVLQYTTCSFDVCYQEIFSTLLSGGQLYLITNE LRRHVEKLFAFIQEKQISILSLPVSFLKFIFNEQDYAQSFPRCVKHIITAGEQLVVTHELQKYLR QHRVFLHNHYGPSETHVVTTCTMDPGQAIPELPPIGKPISNTGIYILDEGLQLKPEGIVGELYI SGANVGRGYLHQPELTAEKFLDNPYQPGERMYRTGDLALWLPDGQLEFLGRIDHQVKIRG HRIELGEIESRLLNHPAIKEAVV" ~400-500 AA in length
- Example: TYREL - Threonine >Tyrosine>Arginine>...->Leucine
- Output one of 30 known substrate classes



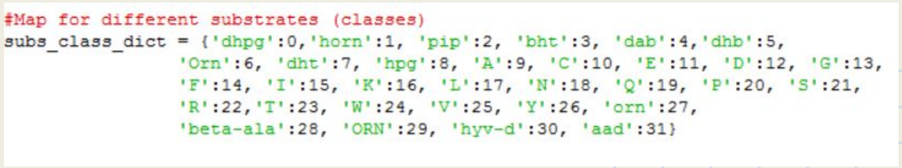
```
#AA List
AminoAcidList = ['G', 'P', 'A', 'V', 'L', 'I', 'M', 'C',
                 'F', 'Y', 'W', 'H', 'K', 'R', 'Q', 'N',
                 'E', 'D', 'S', 'T']
```



O30408_1	P	TYRELDEKSNQLARFLRKKGIGTGSL
O30408_2	F	TYRELNERANQLAHTLRAKGVQAEQ
O30408_3	F	SFRELNERANQLAAVLREKGVGPAQ
O30409_1	N	TYQELMERSAQLANALREKGIASGSI
O30409_2	Q	TYRELNARANQLARLLRSHGTGPDTL
O30409_3	Y	SYQELNERANQLAATLRERGVQPDQ
O30409_4	V	TYRELNEKANQLAHVLRQNGVGKES
O30409_5	Orn	TYQELNAKANQLARVLRRKGVKPES
O30409_6	L	TYSELNERANRLARVLRAKGVGPDR
O31827_0	I	SYKELDKRSNALARELIQKGFRKNET
O68006_1	I	TYRELNEKANQTARLLREKGIGRGSI
O68006_2	C	TYQELDEKSNQVARFLIGKGVEKGD
O68006_3	L	TYRQLNEKANQVARLLREKGVKPDT
O68006_4	E	SYRELNERANSLAFTLRQKGVGPDV
O68006_5	I	TYRELNEKANQTARLLREKGIGRGSI
O68007_1	K	TYRELNEKANQLAWLLREKGVKPDT
O68007_2	Orn	TYRELNEKANQLARTLRQKGVQRES
O68008_1	I	TYRELNEKANQTARLLREKGIGRGSI
O68008_2	F	TYRQLHERSNQLARFLREKGVQPDT
O68008_4	D	TYRELNERANQLARLLRDKGADADQ
O68008_5	N	TYRELNEKSNQLARYLRDKGVKADT
P09095_0	F	SYQELNRKANQLARALLEKGVQTDS
P0C061_0	F	TYHELNVKANQLARIFIEKGIGKDTLV



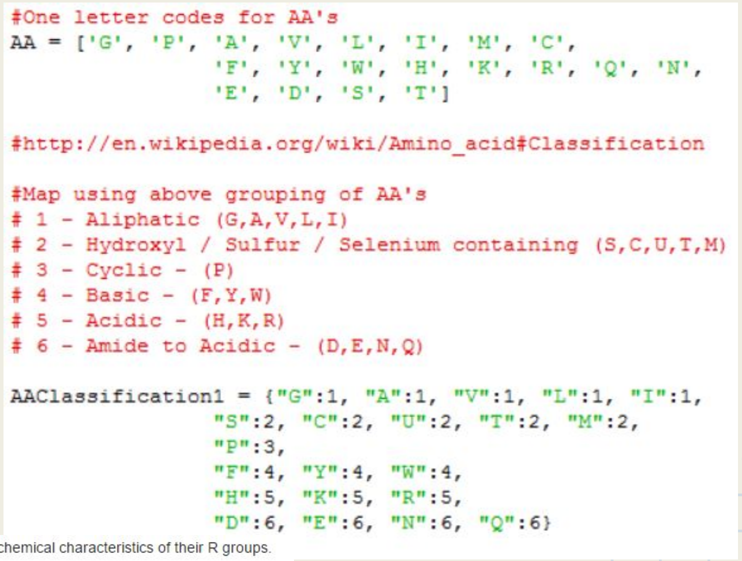
```
G - Glycine (Gly)
P - Proline (Pro)
A - Alanine (Ala)
V - Valine (Val)
L - Leucine (Leu)
I - Isoleucine (Ile)
M - Methionine (Met)
C - Cysteine (Cys)
F - Phenylalanine (Phe)
Y - Tyrosine (Tyr)
W - Tryptophan (Trp)
H - Histidine (His)
K - Lysine (Lys)
R - Arginine (Arg)
Q - Glutamine (Gln)
N - Asparagine (Asn)
E - Glutamic Acid (Glu)
D - Aspartic Acid (Asp)
S - Serine (Ser)
T - Threonine (Thr)
"""
```



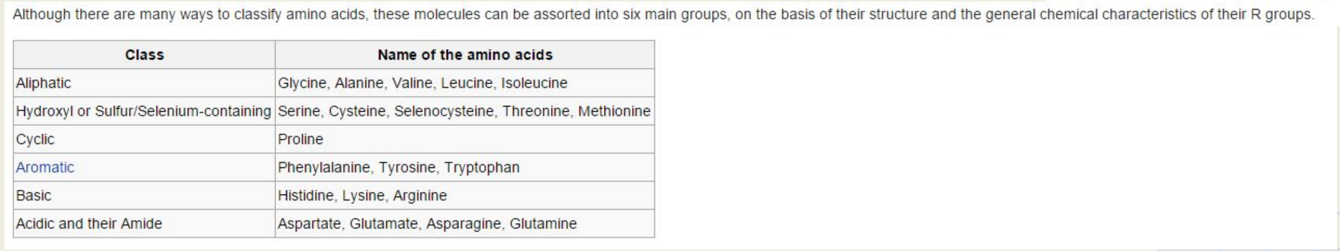
```
#Map for different substrates (classes)
subs_class_dict = {'dhpg':0,'horn':1, 'pip':2, 'bht':3, 'dab':4,'dhb':5,
                   'Orn':6, 'dht':7, 'hpg':8, 'A':9, 'C':10, 'E':11, 'D':12, 'G':13,
                   'F':14, 'I':15, 'K':16, 'L':17, 'N':18, 'Q':19, 'P':20, 'S':21,
                   'R':22,'T':23, 'W':24, 'V':25, 'Y':26, 'orn':27,
                   'beta-ala':28, 'ORN':29, 'hyv-d':30, 'aad':31}
```

Amino Acid Class Feature Space Mapping

- AA Class Representation:
    - Replace each AA with a class representing some chemical / physical properties of the AA
        - Build classes based on 5 datasets (see paper for references)
            - http://www.ann.com.au/MedSci/amino.htm
            - http://en.wikipedia.org/wiki/Amino_acid
            - Reflect some "feature" of amino acid, e.g. structure, chemical properties, physical properties (polarity, acidity, more scientific stuff)
        - New representation: e.g. "**IEDPRA" -> "166351"**
        - Reduces our feature space, e.g. "**IEDPRA" = 166351 = "ANQPHL", gives NEW notions of similarity**
            - May provide more accurate representation of underlying data



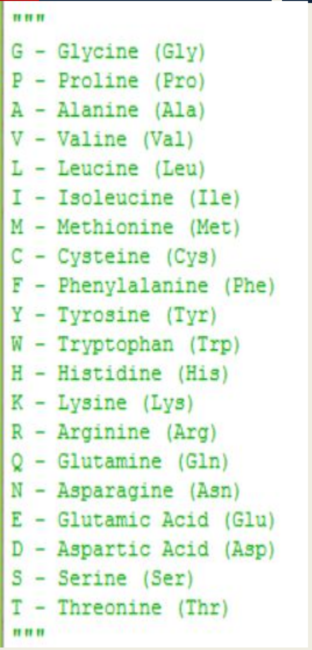
```
#One letter codes for AA's
AA = ['G', 'P', 'A', 'V', 'L', 'I', 'M', 'C',
                'F', 'Y', 'W', 'H', 'K', 'R', 'Q', 'N',
                'E', 'D', 'S', 'T']

#http://en.wikipedia.org/wiki/Amino_acid#Classification

#Map using above grouping of AA's
# 1 - Aliphatic (G,A,V,L,I)
# 2 - Hydroxyl / Sulfur / Selenium containing (S,C,U,T,M)
# 3 - Cyclic - (P)
# 4 - Basic - (F,Y,W)
# 5 - Acidic - (H,K,R)
# 6 - Amide to Acidic - (D,E,N,Q)

AAClassification1 = {"G":1, "A":1, "V":1, "L":1, "I":1,
                "S":2, "C":2, "U":2, "T":2, "M":2,
                "P":3,
                "F":4, "Y":4, "W":4,
                "H":5, "K":5, "R":5,
                "D":6, "E":6, "N":6, "Q":6}
```



Although there are many ways to classify amino acids, these molecules can be assorted into six main groups, on the basis of their structure and the general chemical characteristics of their R groups.

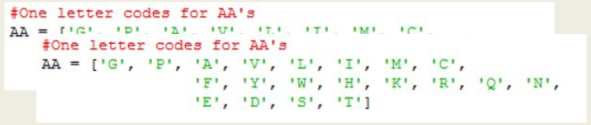
Class	Name of the amino acids
Aliphatic	Glycine, Alanine, Valine, Leucine, Isoleucine
Hydroxyl or Sulfur/Selenium-containing	Serine, Cysteine, Selenocysteine, Threonine, Methionine
Cyclic	Proline
Aromatic	Phenylalanine, Tyrosine, Tryptophan
Basic	Histidine, Lysine, Arginine
Acidic and their Amide	Aspartate, Glutamate, Asparagine, Glutamine

String Kernels / Representations



```
"""
G - Glycine (Gly)
P - Proline (Pro)
A - Alanine (Ala)
V - Valine (Val)
L - Leucine (Leu)
I - Isoleucine (Ile)
M - Methionine (Met)
C - Cysteine (Cys)
F - Phenylalanine (Phe)
Y - Tyrosine (Tyr)
W - Tryptophan (Trp)
H - Histidine (His)
K - Lysine (Lys)
R - Arginine (Arg)
Q - Glutamine (Gln)
N - Asparagine (Asn)
E - Glutamic Acid (Glu)
D - Aspartic Acid (Asp)
S - Serine (Ser)
T - Threonine (Thr)
"""
```

- Bag of Words AA Representation:
  - IEDPRAAAA: [0,1,4,0,0,1,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0]
  - AIEPRAAA: [0,1,4,0,0,1,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0]
    - This representation ignores sequence, so the kernel entry for the two amino acids would be 1 (max similarity).
- AA k-mer Representation:
  - Feature vector containing 20^k combinations of AA's., e.g. a 2-mer representation of a peptide would be a 20^2 long feature vector with every combination of AA's, e.g. AR, AN., AD...YV.
  - IEDPRA - 2 mers woudl be IE, ED, DP, PR, RA.
  - Instead I used AA-Class k-mer content
- AA-Class k-mer Representation
  - New representation: e.g. IEDPRA -> "166351"
    - Instead of a 20^2 space, we have a 6^2 space, much more tractable
    - 16, 66, 63, 35, 51, would be 2-mers of "166351" using this definition
      - 166, 663, 351 - 3 mers (what I used in actual training data)

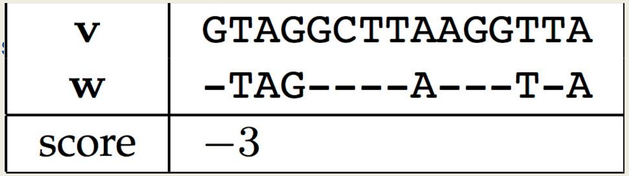


```
#One letter codes for AA's
AA = ['G', 'P', 'A', 'V', 'L', 'I', 'M', 'C',
```

```
#One letter codes for AA's
AA = ['G', 'P', 'A', 'V', 'L', 'I', 'M', 'C',
      'F', 'Y', 'W', 'H', 'K', 'R', 'Q', 'N',
      'E', 'D', 'S', 'T']
```

String Kernels / Representations

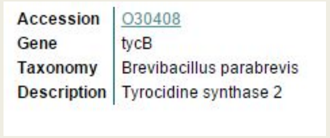
- Alignment Kernel (Naive approach-Needleman Wunsch):
  - Attempts to find “optimal alignment” given penalties for gaps, mismatche and values for matches. (-1, -1, 1)
    - Compare each possible alignment with every other possible one using DP and find highest scoring match
    - Computationally expensive
  - Alternative: de brujin graphs (much faster, more complicated)
  - Use alignment score with every other string as Kernel / similarity measure, after normalizing scores to 0-1 range. (Since kernel matrix must be semi positive definite)



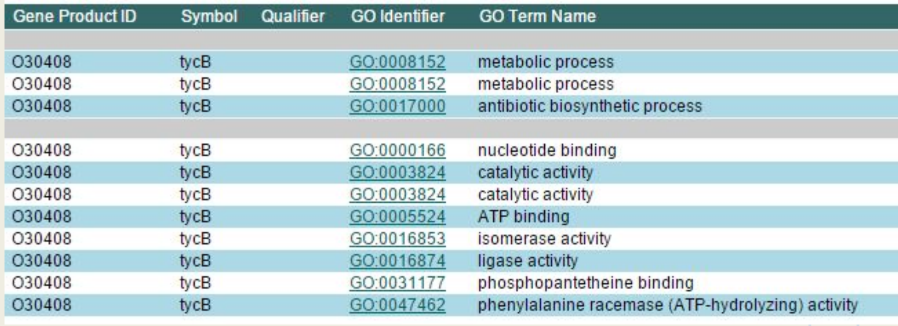
v	GTAGGCTTAAGGTTA
w	-TAG----A---T-A
score	-3

Domain Knowledge Kernel (Uniprot)

- Scraped uniprot to build feature set based on GO protein annotations
- Result was a 54-dimensional vector describing various properties / functions of the protein
  - 1 if it exhibited a property e.g ATP binding
  - 0 if not
- Taking dot product then normalizing the data to limit the data between the ranges of 0-1 gives us a **domain knowledge kernel**
- Adapted a script I found on github to scrape Uniprot (Protein annotations database) and store information in a python dictionary
- Currently **PFAM kernel** is constructed in a very similar manner. I wanted to see if I could create my own using similar procedures



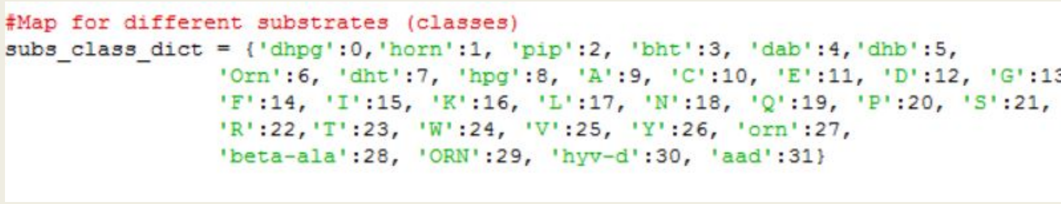
Accession	O30408
Gene	tycB
Taxonomy	Brevibacillus parabrevis
Description	Tyrocidine synthase 2



Gene Product ID	Symbol	Qualifier	GO Identifier	GO Term Name
O30408	tycB		GO:0008152	metabolic process
O30408	tycB		GO:0008152	metabolic process
O30408	tycB		GO:0017000	antibiotic biosynthetic process
O30408	tycB		GO:0000166	nucleotide binding
O30408	tycB		GO:0003824	catalytic activity
O30408	tycB		GO:0003824	catalytic activity
O30408	tycB		GO:0005524	ATP binding
O30408	tycB		GO:0016853	isomerase activity
O30408	tycB		GO:0016874	ligase activity
O30408	tycB		GO:0031177	phosphopantetheine binding
O30408	tycB		GO:0047462	phenylalanine racemase (ATP-hydrolyzing) activity

Recap of approach

- Goal: create multiple kernel SVM predictor of substrates that bind to a specific A domain
  - e.g. **input->APTRLS**…..
  - **output: predicted substate class-> dhpg**



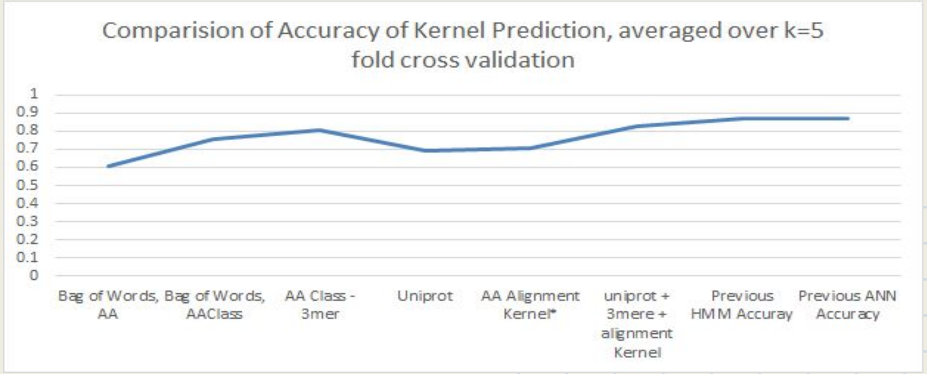
```
#Map for different substrates (classes)
subs_class_dict = {'dhpg':0,'horn':1, 'pip':2, 'bht':3, 'dab':4,'dhb':5,
                   'Orn':6, 'dht':7, 'hpg':8, 'A':9, 'C':10, 'E':11, 'D':12, 'G':13
                   'F':14, 'I':15, 'K':16, 'L':17, 'N':18, 'Q':19, 'P':20, 'S':21,
                   'R':22,'T':23, 'W':24, 'V':25, 'Y':26, 'orn':27,
                   'beta-ala':28, 'ORN':29, 'hyv-d':30, 'aad':31}
```

- In total, 5 feature transformations / kernels were implemented to better capture salient features of the data. These were trimmed down from a much larger set of kernels, which did not show much predictive power / were too computationally expensive to calculate.
  - Bag of Words AA representation
  - Bag of Words AA-Class4 representation
    - Originally had 5 different class representations, after trying each separately and linear combinations of separate class representations, choosing just the 4th class as a representation of our AA gave us the higher accuracy
  - 3-mer AA-Class4 representation
  - AA - Alignment Kernel
  - Uniprot Kernel
  - My final multiple kernel was a combination of the 3-mer AA class4 kernel, AA-Alignment kernel, and Uniprot kernel, capturing what I believe to be relevant:
    - chemical/physio properties, sequential alignment properties, and functional properties of a protein.

Initial Conclusions and Results

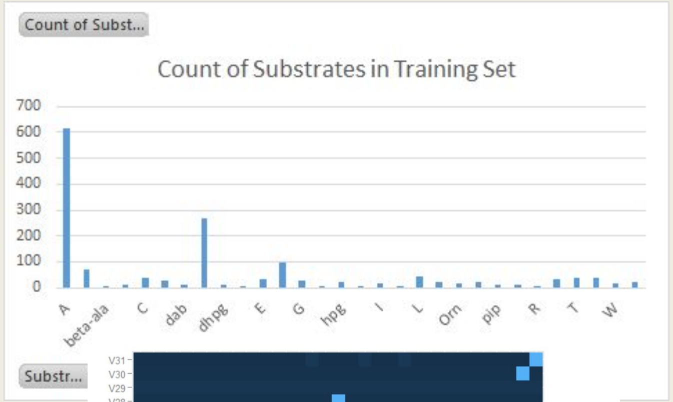
Kernel Used	Accuracy
Bag of Words, AA	0.603073329
Bag of Words, AAClass4	0.753536236
AAClass4 - 3mer	0.807205687
Uniprot	0.691433707
AA Alignment Kernel	0.703073329
uniprot + AA4Class 3mer + alignment Kernel	0.825974682
Previous HMM Accuray	0.87
Previous ANN Accuracy	0.87

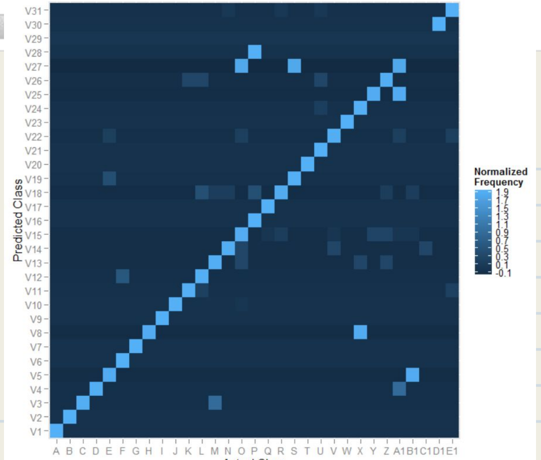
- Bag of words model (AA) gave us the lowest accuracy (still surprisingly high considering # of classes).
- Uniprot kernel performed poorer than expected.
  - Possibly due to overfitting due to too many features.
- AAClass 3-mer model performed suprisingly well.
- The **best results came from a combined kernel**, in which I incorporated the AA class 3mer, uniprot, and alignment features, weighting them equally(naive approach).
  - The sum of the weights equalled 1.
- Combining information from different sources using a naive equal weighting approach led to higher accuracy than individual predictors
  - However the improvement was only ~2.5% not sure if its "statistically significant"



Open Questions / Concerns / Future Investigation

- Dataset itself was very imbalanced/somewhat outdated…
  - **618/1546 NRPS bound to A substrate,** a naive predictor would achieve **0.39974126778 accuracy**
    - Stratified k-fold sampling? More folds? Multiplier knowing prior class probabilities?
  - Try to recreate using more current dataset (however lose baseline for comparison)
  - Now over 400 known NRPS substrates vs 30
  - Trouble predicting certain substrate from confusion matrix: F in particular (out of classes with >30 members)
    - Also reflected in original 2011 study
- More rigorous analysis of data (besides just accuracy comparisons)
  - ROC curves, kernel PCA for visualization, etc.
- Methodical generation of kernel weights (there are a bajillion different methods)
  - Probabilistic construction of kernel weights using Bayesian approach
- Attempt to implement string kernels, instead of projecting it into feature space and then computing kernel
  - There are many optimizations, e.g. de brujin graphs, which I did not have time to implement due to the complexity of the algorithm / wanting to hand code the algorithm myself. Given time it would be nice to try some of the faster more complex string matching / alignment algorithms.





Takeaways / Insights Gained

- Feature engineering / selection is crucial for classification/clustering.
    - Domain knowledge is crucial. Spending a couple minutes thinking about what are the relevant features of the data are **way way way way** more important than the algorithms themselves.
- So is a “good” dataset, if we want accurate results. (or a bad/biased dataset if we want to claim good results)
- See a trend heading towards multiple learning methods similar to multi-kernel learning
    - Deep Learning
    - Random Forest / other ensemble methods (kind of)
- Reading through papers this quarter, it seemed that most predictive models were based on a Bayesian framework or an HMM framework. It seems that most authors have a conclusion in mind already, and then work backwards from that conclusion creating their models to suit the hypothesis.
    - Some rely on very strong (often oversimplifying) assumptions
    - Dangerous, especially in biology where it seems every 2-3 years our models are being drastically revised
- I think that in fields where relatively little is known, e.g. bioinformatics, more exploratory data analysis is required, rather than predictive; and kernels / feature space transformations can be a tool for “exploring” the data.
    - Seeing what kernels give us better accuracy lets us know what features are more relevant for extracting meaning from data.
    - Can be constructed relatively easily without much domain knowledge (scary and powerful).