6.047 / 6.878 Computational Biology: Genomes, Networks, Evolution
Fall 2008

6.047/6.878 Fall 2008 Lecture #5

1. Overview

In the previous lecture we looked at examples of unsupervised learning techniques, such as clustering. In this lecture we focus on the use of supervised learning techniques, which use the inherent structure of the data to predict something about the state or "class" we should be in. There are two different approaches to classification which use either – (1) generative models that leverage probabilistic models that generate data, and (2) discriminative models that use an appropriate function to tease apart the data. Naïve Bayes' classifiers are an example of generative models and Support Vector Machines (SVMs) are example of discriminative models. We will discuss actual biological applications of each of these models, specifically in the use of Naïve Baye's classifiers to predict mitochondrial proteins across the genome and the use of SVMs for the classification of cancer based on gene expression monitoring by DNA microarrays. The salient features of both techniques and caveats of using each technique will also be discussed.

2. Classification – Bayesian Techniques

We will discuss classification in the context of the problem of identifying mitochondrial proteins. If we look across the genome, how do we determine which proteins are involved in mitochondrial processes or more generally which proteins are targeted to the mitochondria[1]. This is particularly useful because if we know the mitochondrial proteins, we can start asking interesting questions about how these proteins mediate disease processes and metabolic functions. The classification of these mitochondrial proteins involves the use of 7 features for all human proteins – (1) targeting signal, (2) protein domains, (3) co-expression, (4) mass spectrometry, (5) sequence homology, (6) induction, and (7) motifs.

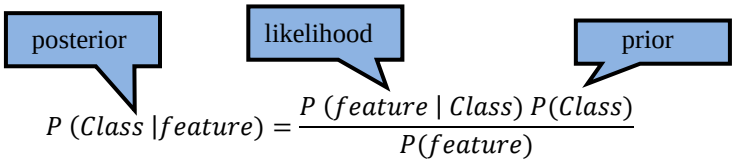
In general, our approach will be to determine how these features are distributed for objects of different classes. The classification decisions will then be made using probability calculus. For our case with 7 features for each protein (or more generally, each object), it is likely that each object is associated with more than one feature. To simplify things let us consider just one feature and introduce the two key concepts.

(1) *We want the model for the probability of a feature given each class.* So the features from each class are drawn from what are known as class-conditional probability distributions (CCPD).

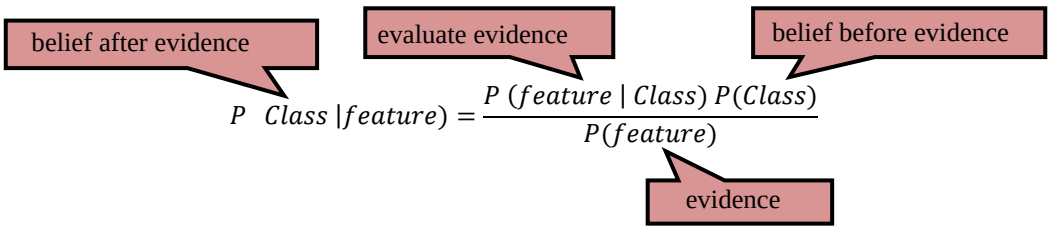
1. Mitochondria play an important role in metabolic processes and are known as the "powerhouse" of the cell. They represent an ancient bacterial invader that has been successfully subsumed by eukaryotic cells. Mitochondria have their own genetic material, commonly referred to as mtDNA. Interestingly, a large fraction of inherited mitochondrial disorders are due to nuclear genes encoding specific proteins targeted to the mitochondria and not due to mutations in mtDNA itself. Hence it is imperative to identify these mitochondrial proteins to shed light on the molecular basis for these disorders.

(2) *We model prior probabilities to quantify the expected a priori chance of seeing a class*. For the case of identifying mitochondrial proteins, we would like to have a low prior probability of seeing the "mitochondrial" class. In this way we can bias ourselves against classifying mitochondrial proteins. (But how do we actually determine these prior probabilities? We will cover this in section 2.2).

A good example of how our second concept affects classification was discussed in lecture. We discussed the example of Eugene. Eugene likes to read books in philosophy and has a penchant for tweed jackets. The question posed was whether Eugene is more likely (in probabilistic terms) to be an ivy-league dean or a truck driver. The answer was that he was more likely to be a truck driver, due to the fact that there are many more truck drivers in the world than there are ivy-league deans. This overwhelming probability outweighs the fact that ivy-league deans are more likely to enjoy books on philosophy and tweed jackets than truck drivers. This prior knowledge influences our decision in classification and we can use this use prior information as the probability of belonging to a certain class *P(Class)* to make our classifications. Now we that we have the prior *P(Class)* and likelihood of a given feature, *P(feature | Class)*, we can simply apply Bayes' rule to determine the posterior *P(Class | feature)* since we are interested in classification based on features.



This is the application of Bayes' rule to our problem with the probabilities defined as shown above. More generally we can think of the probabilities as below,



We can now think about applying this recursively across classes and using the "belief after evidence" to form new "evidence" and then find new "beliefs after evidence". Once we've determined all our "beliefs after evidence" for each class, we can use a decision rule to choose a particular class.

Given two classes, Bayes' decision rule would be as follows,

$$P\ (Class\ 1\ |\ feature) > P(\ Class\ 2\ |\ feature)$$

$$P\ (feature\ |\ Class\ 1)P\ (Class\ 1) > P\ (feature\ |\ Class\ 2)\ P(Class\ 2)$$

We can take log probabilities to build a discriminant function as below,

$$G(feature) = \log \frac{P(feature \mid Class\ 1)\ P\ (Class\ 1)}{P(feature \mid Class\ 2)\ P\ (Class\ 2)} > 0$$

In this case the use of logarithms provide distinct advantages –

(1) numerical stability,

(2) easier math (it's easier to add the expanded terms than multiply them), and

(3) monotonically increasing discriminators.

This discriminant function does not capture the penalties associated with misclassification (in other words, is one classification more detrimental than other). In this case, we are essentially minimizing the number of misclassifications we make overall, but not assigning penalties to individual misclassifications. From examples we discussed in class and in the problem set - if we are trying to classify a patient as having cancer or not, it could be argued that it is far more harmful to misclassify a patient as being healthy if they have cancer than to misclassify a patient as having cancer if they are healthy. In the first case, the patient will not be treated and would be more likely to die, whereas the second mistake involves emotional grief but no greater chance of loss of life. To formalize the penalty of misclassification we define something called a loss function, L_{kj}, which assigns a loss to the misclassification of an object as class *j* when the true class is class *k* (a specific example of a loss function was seen in Problem Set 2).

2.1 Training Sets

Classifiers require a set of labeled data points. The number of data points within this data set will directly impact the quality of the classifier, however, the number of data points necessary to build a training set will be largely dependent on the problem to be solved. The importance of the training set is to correctly estimate the feature-given-class likelihoods. This can be done using a multinomial distribution.

2.2 Getting Priors

This involves getting *P(Class)* to complete the implementation of a Bayes' classifier. There are three general approaches –

(1) Estimate the priors by counting and dividing by the total to obtain the fraction of classes in a training set (again, this emphasizes the quality of the training set data)

(2) Estimate from "expert" knowledge, in that we might have information from actual experimental work, and have reasoning about the distribution from higher-level data

(3) Assume no prior knowledge, so all classes are equally likely

2.3 Multiple Features

Going back to our original example of mitochondrial proteins, we were looking at 7 features and not just a single feature. This makes things much more complicated and exemplifies the "curse of dimensionality". We would require huge amounts of data to be able to implement the classifier. Most likely we would end up with zero probability for a certain combination of features for a particular class, which might not be an accurate representation of the real probabilities. We can get around this by assuming the features are independent (this forms the crux of a Naïve Bayes' approach). Once we assume independence we can now take the product of all the conditional probabilities and we have to only estimate the individual distributions for each feature. So we now have,

$$P\ (f1,\ \ f2, \ldots . fN|\ Class) = P\ (f1\ |Class)\ P\ (f2\ |\ Class)\ \cdots\cdots\ P(fN\ |Class)$$

where, *f1* represents *feature* 1. Similarly, the discriminant function can be changed to the multiplication of the prior probabilities.

$$G(f_1, f_2, \ldots . , f_N) = \log \frac{\prod P\ (f_i\ |\ Class\ 1)\ P\ (Class\ 1)}{\prod P\ (f_i\ |\ Class\ 2)\ P\ (Class\ 2)} > 0$$

2.4 Testing a classifier

The classifier must be tested on a different set (termed the labeled test set) than the training set. Apparently, a large number of publications continue to test their classifiers solely with their training data sets (resulting in skewed performance metrics). To perform a robust classification we classify each object in the test set and count the number of each type of error. In the case of a binary classification (i.e. an object either belongs to Class 1 or Class 2), we have the following four types of errors,

(1) True positive (TP)
(2) True negative (TN)
(3) False positive (FP)
(4) False negative (FN)

The frequency of these errors can be encapsulated in performance metrics of a classifier which are defined as,

(1) Sensitivity – determines how good we are at classifying. In the case of our binary example, what fraction of *actual* Class 1 classified as Class1?

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

(2) Specificity – determines how tolerant we are to misclassifications. What fraction of *not* Class 2 (in the binary case, simply Class 1) classified as Class 2?

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

The higher the sensitivity (think of it as minimizing FN) and higher the specificity (think of it as minimizing FP) the better the classifier. However, achieving a high score in one of these areas (and not the other) doesn't amount to much. Consider the classification of mitochondrial proteins, we can classify all data points as mitochondrial proteins and get 100% sensitivity but 0% specificity. The MAESTRO algorithm achieves a 99% specificity and a 71% sensitivity for the classification of mitochondrial proteins.

In all, we have defined Bayesian classifiers and shown how to determine all the probabilities required to implement a Bayesian classifier and discussed a particular implementation that assumes independence of features (Naïve Bayesian classifiers). The final section describes the specifics of the application of classification to determine mitochondrial proteins.

2.6 MAESTRO – Mitochondrial Protein Classification

Calvo et al. (2006) sought to construct high-quality predictions of human proteins localized to the mitochondrion by generating and integrating data sets that provide complementary clues about mitochondrial localization. Specifically, for each human gene product *p*, they assign a score $s_i(p)$, using each of the following seven genome-scale data sets – targeting signal score, protein domain score, *cis*-motif score, yeast homology score, ancestry score, coexpression score, and induction score (details of each of the meaning and content of each of these data sets can be found in the manuscript). Each of these scores (s_1–s_7) can be used individually as a weak genome-wide predictor of mitochondrial localization. Each method's performance was assessed using large 'gold standard' curated training sets - 654 mitochondrial proteins (T_{mito}) maintained by the MitoP2 database1 and 2,847 nonmitochondrial proteins ($T_{\sim mito}$) annotated to localize to other cellular compartments. To improve prediction accuracy, the authors integrated these eight approaches using a naïve Bayes classifier that was implemented as a program called MAESTRO.

When MAESTRO was applied across the human proteome, 1451 proteins were predicted as mitochondrial proteins and 450 novel proteins predictions were made. As mentioned in the previous section The MAESTRO algorithm achieves a 99% specificity and a 71% sensitivity for the classification of mitochondrial proteins, suggesting that even with the assumption of feature independence, Naïve Bayes classification techniques can prove extremely powerful for large-scale (i.e. genome-wide) scale classification.

3. Classification – Support Vector Machines

The previous section looked at using probabilistic (or generative) models for classification, this section looks at using discriminative techniques – in essence, can we run our data through a function to determine

its structure? Such discriminative techniques avoid the inherent cost involved in generative models which might require more information than is actually necessary.

Support vector machine techniques essentially involve drawing a vector that's perpendicular to the line optimally separating the data. This vector is determined from a set of support vectors from the individual data points. Interestingly, the vector can be determined from the scalar dot-product of these support vectors, so even if we get to higher dimensions then we go from points in multi-dimensional space to just a simple dot-product (allowing for simpler implementation for higher order data).

In this case when the data is not linearly separable we can use a kernel function (essentially a transform) to map to a higher dimensional space where the data is now linearly separable. The beauty of this technique is that since the SVM technique only requires the dot-product of the support vectors we can work directly with the kernel mapping matrix and not with the actual data points. There are a number of transformations (such as polynomial and sigmoidal) that serve as good kernel mapping functions and are furthered explored in the next section.

3.1 Non-Linear Classification

What if the data are not linearly separable? One way to deal with this is to just to try to find the best linear classifier for the data, and accept mistakes. Another, completely different, way to deal with this comes from the fact that the more dimensions there are, the more likely it is that the data set is linearly separable. In a certain sense, for any finite data set, as the number of dimensions goes to infinity, the probability that the data is separable goes to 1.

How can we take advantage of this? Consider the two classes of one-dimensional data:
$\{-5,-4,-3,3,4,5\}$ and $\{-2,-1,0,1,2,\}$. This data is clearly not linearly separable, and the best separation boundary we can find might be $x > -2.5$.
Now, consider applying the transformation $x \to (x, y = x^2)$. The data now can be written as new pairs,
$\{-5,-4,-3,3,4,5\} \to (-5,25), (-4,16), (-3,9), (3,9), (4,16), (5,25)$, and
$\{-2,-1,0,1,2,\} \to (-2,4), (-1,1), (0,0), (1,1), (2,4)$
This data is separable by the rule $y > 6.5$, and in general the more dimensions we transform to, the more separable it becomes.

An alternate way of thinking of this problem is to transform the classifier back in to the original low-dimensional space. In this particular example, we would get the rule $x^2 < 6.5$, which would bisect the number line at two points. In general, the higher dimensionality of the space that we transform to, the more complicated a classifier we get when we transform back to the original space.

3.2 Kernels

We see that SVMs are dependent only on the dot products of the vectors. So, if we call our transformation $\phi(v)$, for two vectors we only care about the value of $\phi(v_1) \cdot \phi(v_2)$. The trick with use of kernels consists of realizing that for certain transformations ϕ, there exists a function $K(v_1, v_2)$, such that,

$$K(v_1, v_2) = \phi(v_1) \cdot \phi(v_2)$$

In the above relation, the right-hand side is the dot product of vectors with very high dimension, but the left-hand side is the function of two vectors with lower dimension. In our previous example of mapping $x \to (x, y = x^2)$, we get

$$K(x_1, x_2) = (x_1, {x_1}^2) \cdot (x_2, {x_2}^2) = x_1 x_2 + (x_1 x_2)^2$$

Now we need not actually apply the transformation ϕ, we can do all our calculations in the lower dimensional space, but get all the power of using a higher dimension.

Example kernels are the following,

(1) Linear kernel: $K(v_1, v_2) = v_1 \cdot v_2$
which represents the trivial mapping of $\phi(x) = x$

(2) Polynomial kernel: $K(v_1, v_2) = (1 + v_1 \cdot v_2)^n$
which was used in the previous example, if we had instead used the transformation $\phi(x) = (1, \sqrt{2}x, x^2)$, we would get a more polynomial-looking kernel of $\phi(v_1) \cdot \phi(v_2) = (1 + v_1 \cdot v_2)^2$

(3) Radial Basis kernel: $K(v_1, v_2) = \exp(-\gamma |v_1 - v_2|^2)$
This transformation is actually from a point $v_1$to a function (which can be thought of being a point in Hilbert space) in a infinite-dimensional space. So what we're actually doing is transforming out training set into functions, and combining them to get a decision boundary. The functions are Gaussians centered at the input points.

(4) Sigmoid kernel: $K(v_1, v_2) = \tanh[\, \gamma({v_1}^T v_2 + \mathrm{r})]$
Sigmoid kernels have been popular for use in SVMs due to their origins in neural networks (e.g. sigmoid kernel functions are equivalent to two-level, perceptron neural networks). It has been pointed out in previous work (Vapnik 1995) that the kernel matrix may not be positive semi-definite for certain values of the parameters γ and r. The sigmoid kernel has nevertheless been used in practical applications (see Scholkopf 1997).

One of the caveats of transforming the input data using a kernel is the risk of over-fitting (or over-classifying) the data. More generally, the SVM may generate so many feature vector dimensions that it does not generalize well to other data. To avoid over-fitting, cross-validation is typically used to evaluate

the fitting provided by each parameter set tried during the grid or pattern search process. In the radial-basis kernel, you can essentially increase the value of γ until each point is within its own classification region (thereby defeating the classification process altogether). SVMs generally avoid this problem of over-fitting due to the fact that they maximize margins between data points.

When using difficult-to-separate training sets, SVMs can incorporate a cost parameter *C*, to allow some flexibility in separating the categories. This parameter controls the trade-off between allowing training errors and forcing rigid margins. It can thereby create a *soft margin* that permits some misclassifications. Increasing the value of *C* increases the cost of misclassifying points and forces the creation of a more accurate model that may not generalize well.

Can we use just any function as our kernel? The answer to this is provided by Mercer's Condition which provides us an analytical criterion for choosing an acceptable kernel. Mercer's Condition states that a kernel *K(x,y)* is a valid kernel if and only if the following holds (Burgess 1998),

For any *g(x)*, such that,

$$\int g(x)^2 dx \quad \text{is finite}$$

then,

$$\iint K(x,y)g(x)g(y)dxdy \geq 0$$

In all, we have defined SVM discriminators and shown how to perform classification with appropriate kernel mapping functions that allow performing computations on lower dimension while being to capture all the information available at higher dimensions. The next section describes the application of SVMs to the classification of tumors for cancer diagnostics.

3.3 Tumor Classifications with SVMs

A generic approach for classifying two types of acute leukemias – acute myeloid leukemia (AML) and acute lymphoid leukemia (ALL) was presented by Golub *et al.* (1999). This approach centered on effectively addressing three main issues,

(1) whether there were genes whose expression pattern to be predicted was strongly correlated with the class distinction (i.e. can ALL and AML be distinguished)
(2) how to use a collection of known samples to create a "class predictor" capable of assigning a new sample to one of two classes
(3) how to test the validity of their class predictors

They addressed (1) by using a "neighbourhood analysis" technique to establish whether the observed correlations were stronger than would be expected by chance. This analysis showed that roughly 1100 genes were more highly correlated with the AML-ALL class distinction than would be expected by chance. To address (2) they developed a procedure that uses a fixed subset of "informative genes" (chosen based on their correlation with the class distinction of AML and ALL) and makes a prediction based on the expression level of these genes in a new sample. Each informative gene casts a "weighted vote" for one of the classes, with the weight of each vote dependent on the expression level in the new sample and the degree of that gene's correlation with the class distinction. The votes are summed to determine the winning class. To address (3) and effectively test their predictor by first testing by cross-validation on the initial data set and then assessing its accuracy on an independent set of samples. Based on their tests, they were able to identify 36 of the 38 samples (which were part of their training set!) and all 36 predictions were clinically correct. On the independent test set 29 of 34 samples were strongly predicted with 100% accuracy and 5 were not predicted.

A SVM approach to this same classification problem was implemented by Mukherjee *et al.* (2001). The output of classical SVM is a binary class designation. In this particular application it is particularly important to be able to reject points for which the classifier is not confident enough. Therefore, the authors introduced a confidence interval on the output of the SVM that allows for rejection of points with low confidence values. As in the case of Golub *et al.* it was important for the authors to infer which genes are important for the classification. The SVM was trained on the 38 samples in the training set and tested on the 34 samples in the independent test set (exactly in the case of Golub *et al.*). The authors' results are summarized in the following table (where $|d|$ corresponds to the cutoff for rejection).

Genes	Rejects	Errors	Confidence Level	$\|d\|$
7129	3	0	~93%	0.1
40	0	0	~93%	0.1
5	3	0	~92%	0.1

These results a significant improvement over previously reported techniques, suggesting that SVMs play an important role in classification of large data sets (as those generated by DNA microarray experiments).

4. Semi-supervised learning

In some scenarios we have a data set with only a few labeled data points, a large number of unlabeled data points and inherent structure in the data. This type of scenario both clustering and classification do not perform well and a hybrid approach is required. This semi-supervised approach could involve the clustering of data first followed by the classification of the generated clusters.