This homework is individual work. All assignments are due at the beginning of class on the due date. Assignments turned in late will drop 10 points for each period of 24 hours for which the assignment is late. In addition, no assignments will be accepted after the solutions have been made available. Please include your cornell net id on your homework.

A. Kernels

1. Equivalence Relations as Kernels (20 POINTS)

One way of showing that a function $K : X \times X \rightarrow \Re$ is a valid kernel is by constructing a vector space so that $\forall x_1, x_2 \in X : K(x_1, x_2) = \vec{x}_1 \cdot \vec{x}_2$. You can think about this as constructing the mapping $\Phi(x) = \vec{x}$ that produces the feature vectors $\vec{x}$. For example, in class we used this method to show that the quadratic kernel is a valid kernel and constructed the mapping from the low dimensional input vectors x to the high dimensional vectors $\vec{x}$ (all the polynomial terms of degree two).

Using this method, show that

$$K(x_i, x_j) = \begin{cases} 1 & x_i \text{ and } x_j \text{ are equivalent} \\ 0 & otherwise \end{cases}$$

is a valid kernel for any equivalence relation over X. Describe how you constructed the feature vectors $\vec{x}$.

To remind you, an equivalence relation partitions X into some number k of disjoint subsets $X_1, ..., X_k$, so that $X = X_1 \cup ... \cup X_k$ and $\forall x \in X$, x belongs to exactly one X_i. We denote the subset that x belongs to by $[x]$ (i.e. if $x \in X_i$, then $[x] = i$).

2. The Ideal Kernel (10 POINTS)

You can think about a kernel as a measure of similarity. Imagine you could pick any Kernel defined on an equivalence relation as above.

- What would be the ideal kernel? Explain what function this Kernel should compute.
- Show that this Kernel has margin $\delta = 1$ for any training set.

Hint: Note that you can probably not realisticly compute this Kernel in practice, since it would require that you already solved the learning problem.

3. Kernel Perceptron (20 POINTS)

Note that every time the Perceptron algorithm modifies $\vec{w}$, it adds $\eta y_i \vec{x}_i$ to $\vec{w}$. If we replace the line $\vec{w}_{k+1} = \vec{w}_k + \eta y_i \vec{x}_i$ in the Perceptron algorithm with $\alpha_i = \alpha_i + \eta$, we can see that the Perceptron hyperplane can be expressed in the form $\vec{w} = \sum_{i=1}^{n} \alpha_i y_i \vec{x}_i$.

Using the above formulation, extend the Perceptron algorithm so that it can use Kernels. Assume unbiased hyperplanes (i.e. $b = 0$). Write down the new Perceptron algorithm and explain why your changes work.

B. Generative Models

Naive Bayes for Text Classification (30 POINTS)

Implement a binary naive Bayes classifier for text classification as outlined in Mitchell's book on page 183. Note that it is slightly different from the naive Bayes algorithm we derived in class, since it is particularly designed for text classification.

To test the classifier, we put a dataset on the WWW page. The task is to classify Reuters newswire articles according to whether they are about a corporate acquisition or not. The dataset is in SVM-light format (as described on the WWW page). Each line describes one example. The first value is the class. The following groups are "feature number:value". Each feature number stands for a different word (f.y.i. feature number i stands for the word in the i-th line of the file "words"). If the feature number occurs in the line of the example, then this word occurs in the article at least once. Otherwise, it does not occur. The feature value indicates how often the word occurs in the document.

Train your naive classifier in the following two ways. First, train on $train.dat$ using the estimator
$\hat{P}(X_i = w|Y = y) = \frac{1+\text{number of times word } w \text{ occurs in class } y}{|Vocabulary|+\text{total number of words in class } y}$
given in the book. Second, train your naive Bayes on $train.dat$ using the estimator
$\hat{P}(X_i = w|Y = y) = \frac{\text{number of times word } w \text{ occurs in class } y}{\text{total number of words in class } y}$.

- Attach a printout of your implementation (including comments).
- Report the number of errors on the test set that your two classifiers make. Explain, why the second estimator performs poorly. Hint: Consider the case where a word in a test document has never occured in a training document of the correct class before.

Hint: Implement your naive Bayes classifier, so that it does not multiply probabilities p, but adds $-log(p)$. Otherwise, you will likely exceed the range of floating point numbers.

Single Feature Model (20 POINTS)

Imagine the following learning problem.

You have a binary classification problem where the input space is $X = \{0,\ 1\}^{100}$ (each example has 100 binary features), but each example vector $\vec{x}$ has only exactly one feature that has the value 1. All other features have value 0. All feature vectors in both classes have this particular form, but the probability which feature is 1 depends on the class.

Derive a generative classifier for this problem. Also, propose estimators for the parameters of the model. Explain your solution.