CS 4780/5780 Machine Learning Due Nov 5, 2013

Assignment 4: Kernels & Generative Models

Instructor: Thorsten Joachims *Total Points: 100*

Course Policy:

Read all the instructions below carefully before you start working on the assignment, and before you make a submission

- Assignments are due at the beginning of class on the due-date in hard copy.
- Write your NetIDs with submission date and time on the first page of the hard copy.
- No assignment will be accepted after the solution is publicized (about 4 days after due).
- The submission time of whatever you submit last (hard copy or CMS) is counted as the official submission time and will determine the late penalty.
- Upload only the code you wrote to CMS. Do not upload any additional libraries. Provide a README file with your submission that includes a list of all libraries and instructions needed to run your code.
- Late assignments can be submitted in class or to Ian Lenz in Upson Hall 5151. Since the fifth floor of Upson is locked on the weekends, weekend submissions (all code and answers to all questions) should be made digitally via CMS, with a hard copy delivered to Ian as soon as possible afterwards.
- All sources of material must be cited. Assignment solutions will be made available along with the graded homework solutions. The University Academic Code of Conduct will be strictly enforced, including running cheating detection software.
- No more than one submission per group.

Problem 1: Kernels [35 points]

(a) [**10 points**] Consider the *unbiased* perceptron algorithm. Suppose we are given a kernel function $K(x_i, x_j = \phi(x_i)^T\phi(x_j)$ (for some unknown $\phi(\cdot)$). Modify the perceptron learning algorithm to directly use the kernel K. In particular provide both 1) the training updates and 2) the decision rule during prediction. You do not need to implement the modified algorithm in code; just work out the algorithm on paper.

(b) [**5 points**] What is the mistake bound for this kernel perceptron? Assume that $\forall x : \sqrt{x^T x} \leq R_1$ and $\sqrt{K(x,x)} \leq R_2$. You can also assume that the optimal hyperplane in this kernel space has margin δ. Clearly indicate what changes, if any, from the mistake bound of the regular perceptron are necessary.

(c) [**10 points**] A TA claimed during office hours that kernels allow us to fit non-linear classifiers in the instance space; lets examine if there is any benefit to be achieved in practice. The task we shall attempt is automatic recognition of handwritten digits. The dataset available at UCI Machine Learning Repository has been preprocessed to adhere to SVM-light format and made available on CMS as *digits.zip*. The input is image bitmap of gray-scale values.

The zipped folder contains 10 training sets and 10 validation sets (one pair for each digit) and one overall train, validation and test set. To start the experiment, we intend to learn 10 models (one for each digit) which predicts whether that digit was drawn or not. We will use the 10 pairs of training and validation sets and the overall test set for this experiment.

Use SVM^{light} to train 10 different biased classifiers with a linear kernel for each of the training sets. Use the validation set for each digit to arrive at good values for C (you may consider the range of $C = \{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1\}$).

Note, the best value of C for a classifier for a digit can be different from the best C for another digit. Tabulate the accuracies you get on the validation set for each digit as you vary C. Keep aside the best performing model for each digit for the next step (provide a brief explanation for why you picked a model to be the best performing model).

For each of your models (0-9), run `svm_classify` on the overall test set. After you have run all the classifications, find the highest of the 10 predictions for each test instance, and assign it the corresponding class. What is the accuracy on the overall test set?

Include representative commands (*i.e.,* the command-line arguments) you use for training for one of the digits.

(d) [**10 points**] We will now see the effect of non-linear kernels on performance. Train 10 different biased SVMs with polynomial kernel for each of the training sets, use the corresponding digit's validation set to tune the value of C and the degree of the polynomial, d (you may consider the range of $d = \{2, 3, 4, 5\}$). Note that, the d value will be same across the 10 classifiers (so, fix a value for d, tune C as you did in the first experiment; pick that value of d that produces least errors across all digits). Keep aside the best performing model for each digit for overall prediction. Run each model on the overall test set and use the highest of the 10 predictions to label each test instance. What is the accuracy on the overall test set now? Include representative commands (*i.e.,* the command-line arguments) you use for training for one of the digits.

Problem 2: Generative Models [35 points]

(a) [**10 points**] Linear Discriminant Analysis is named suggestively: perhaps it can be rewritten as a linear rule. Remember, the decision function for LDA to classify an input vector $x \in \mathbb{R}^N$ given class means $\vec{\mu_+}$ and $\vec{\mu_-}$ is:

$$h(\vec{x}) = argmax_{y\in\{+1,-1\}}\ Pr(Y = y)\ e^{-\frac{||(\vec{x}-\vec{\mu_y})||^2}{2}}$$

Assuming we have estimates for all the appropriate probabilities (none of them zero) and we have the class means, can we rewrite $h(x) = sign(\vec{v} \cdot \vec{x} + b)$? If so, provide $\vec{v}$ and b.

(b) [**15 points**] We will attempt to write the multinomial Naive Bayes Classifier Decision rule as a linear rule. Remember, a document is viewed as a sequence $d = (w_1, w_2, ..., w_l)$ of l words and is classified as:

$$h(d) = argmax_{y\in\{+1,-1\}} Pr(Y = y) \prod_{i=1}^{l} Pr(W = w_i|Y = y)$$

Assuming we have estimates for all the appropriate probabilities and none of them are zero, can we rewrite $h(d) = sign(\vec{v} \cdot \vec{x} + b)$? If so, provide $\vec{v}, \vec{x}$ and b.

HINT: For $\vec{x}$: You might find the construction of $\vec{x}$ such that each component corresponds to a word in the vocabulary particularly helpful.

(c) [**10 points**] We learned in class that the Naive Bayes Classifier makes an independence assumption which, given the suggestive name, is probably very naive. Consider the multivariate Naive Bayes Classifier for a binary classification problem, where the input features are all binary. Construct a learning task $Pr(X, Y)$, the Naive Bayes distributions $Pr(X = x|Y = y) = \prod Pr(X_i = x_i|Y = y)$ and $Pr(Y = y)$ that you would learn for $Pr(X, Y)$, and a test point x such that the naive Bayes labeling and the Bayes-optimal labeling of x differ.

Provide calculations to prove your construction answers the question.

HINT: You may wish to consider the case when two of the input attributes are dependent (say, equal).

Problem 3: Naive Bayes Implementation [30 points]

(a) [**15 points**] We will attempt automatic classification of documents as pertaining to Astrophysics or not; a subset of the arXiv corpus of scientific papers has been preprocessed to adhere to SVM-light format and made available on CMS as *arxiv.zip*. The zipped folder contains four files: Each line in the arxiv.train/test file is one instance, the first field indicating if the instance was in the Astrophysics category (+1) or not (-1). The remaining fields in a line indicate:

`<Index of word in dictionary>:<Number of occurrences of word in doc>.`

You can use the maximum index you see in the training file as the size of the vocabulary.

You need to implement a Multinomial Naive Bayes Classifier for this classification problem. Remember, the conditional probability of a word w being in a document d is estimated as:

$$Pr(W = w|Y = y) = \frac{1 + \text{Number of occurrences of w in documents in class } y}{\text{Size of vocabulary} + \text{Number of all words in documents in class } y}$$

$$Pr(Y = y) = \frac{\text{Number of documents in class } y}{\text{Number of all documents}}$$

The decision rule is given in an earlier question. In case of ties, predict the class that occurs more often in the training set.

To avoid underflow, you should not compare products of probabilities directly but rather compare their logarithms instead (remember, the logarithm of a product is a sum of logarithms).

What accuracy do you get on the test set? Explicitly list out the number of false positives and false negatives. (A false positive is when our algorithm predicts the class to be *True* but the actual label is *False*, and a false negative is when we predict *False* when the actual value is *True*).

A linear SVM achieves accuracy of around 96.43% on the test set. How does that compare with the NB classifier?

(b) [**15 points**] We will now examine cost sensitive classification; in several real-world classification tasks, the cost of a false positive is not equal to the cost of a false negative; let mislabeling a true instance as false have cost c_{10} , and mislabeling a false instance as true have cost c_{01} . There are also real-life costs for attaining positive examples (c_{11}) and negative examples (c_{00}). The setting discussed in class is the case when $c_{00} = c_{11} = 0$; $c_{01} = c_{10} = 1$.

In our current setting, we note that a user looking for articles on astrophysics will tend to have non-symmetric costs for false positives and false negatives: for instance, missing one document talking about astrophysics could be just as bad as having to look at a bunch of documents about things other than astrophysics. So, rather than optimizing for accuracy on the test set, we must optimize for this user's cost. For this question, $c_{00} = c_{11} = 0$; $c_{01} = 1; c_{10} = 10$. The decision rule for the multinomial Naive Bayes classifier now becomes

Label instance as +1 if: $(c_{10} - c_{11}) Pr(Y = +1 | X = x) \geq (c_{01} - c_{00}) Pr(Y = -1 | X = x)$

Implement the cost sensitive classifier for the given c values and report your accuracy on the test set. Explicitly list out the number of false positives and false negatives. Can you provide an intuitive connect between the number of false positives, false negatives and c_{01}, c_{10}?