Homework 3, due Thursday October 13 COMS 4771 Fall 2016

Problem 1 (Features; 10 points)**.** It is common to pre-preprocess the feature vectors in $\mathbb{R}^d$ before passing them to a learning algorithm. Two simple and generic ways to pre-process are as follows.

- **Centering**: Subtract the mean $\hat{\boldsymbol{\mu}} := \frac{1}{|S|}\sum_{(\boldsymbol{x},y)\in S}\boldsymbol{x}$ (of the training data) from every feature vector:
$$\boldsymbol{x} \;\mapsto\; \boldsymbol{x} - \hat{\boldsymbol{\mu}}\,.$$
- **Standardization**: Perform centering, and then divide every feature by the per-feature standard deviation $\hat{\sigma}_i = \sqrt{\frac{1}{|S|}\sum_{(\boldsymbol{x},y)\in S}(x_i - \hat{\mu}_i)^2}$:
$$(x_1, x_2, \ldots, x_d) \;\mapsto\; \left(\frac{x_1 - \hat{\mu}_1}{\hat{\sigma}_1}, \frac{x_2 - \hat{\mu}_2}{\hat{\sigma}_2}, \ldots, \frac{x_d - \hat{\mu}_d}{\hat{\sigma}_d}\right).$$

For each of the following learning algorithms, and each of the above pre-processing transformations, does the transformation affect the learning algorithm?

(a) The classifier based on the generative model where class conditional distributions are multivariate Gaussian distributions with a fixed covariance equal to the identity matrix $\boldsymbol{I}$. Assume MLE is used for parameter estimation.

(b) The 1-NN classifier using Euclidean distance.

(c) The greedy decision tree learning algorithm with axis-aligned splits. (For concreteness, assume Gini index is used as the uncertainty measure, and the algorithm stops after 20 leaf nodes.)

(d) Empirical Risk Minimization: the (intractable) algorithm that finds the linear classifier (both the weight vector and threshold) that has the smallest training error rate.

To make this more precise, consider any training data set S from $\mathcal{X} \times \mathcal{Y}$, and let $\hat{f}_S \colon \mathcal{X} \to \mathcal{Y}$ be the classifier obtained from the learning algorithm with training set S. Let $\boldsymbol{\phi} \colon \mathcal{X} \to \mathcal{X}'$ be the pre-processing transformation; and let $\tilde{f}_{S'} \colon \mathcal{X}' \to \mathcal{Y}$ be the classifier obtained from the learning algorithm with training set S', where S' is the data set from $\mathcal{X}' \times \mathcal{Y}$ containing $(\boldsymbol{\phi}(\boldsymbol{x}), y)$ for each $(\boldsymbol{x}, y)$ in S. We say $\boldsymbol{\phi}$ *affects the learning algorithm* if, for any training data S, the classifier $\hat{f}_S$ is not the same as $\boldsymbol{x} \mapsto \tilde{f}_{S'}(\boldsymbol{\phi}(\boldsymbol{x}))$.

You should assume the following: (i) the per-feature standard deviations are never zero; (ii) there are never any "ties" whenever you compute an arg max or an arg min; (iii) there are no issues with numerical precision or computational efficiency.

What to submit: "yes" or "no" for each pre-processing and learning algorithm pair, along with a brief statement of justification.

Problem 2 (More features; 30 points)**.** Download the review data set `reviews_tr.csv` (training data) and `reviews_te.csv` (test data) from Courseworks. This data set is comprised of reviews of restaurants in Pittsburgh; the label indicates whether or not the reviewer-assigned rating is at least four (on a five-point scale). The data are in CSV format (where the first line is the header); the first column is the label ("label"), and the second column is the review text ("text"). The text has been processed to remove non-alphanumeric symbols and to make all letters lowercase.

Write a script that, using only the training data, tries different methods of data representation and different methods for learning a linear classifier, and ultimately selects—*via five-fold cross validation*—and uses one combination of these methods to train a final classifier. (You can think of the choice of these methods as "hyperparameters".)

The data representations to try are the following.

1. Unigram representation.

   In this representation, there is a feature for every word w, and the feature value associated with a word w in a document d is

   $$\mathrm{tf}(w; d) \;:=\; \text{number of times word } w \text{ appears in document } d\,.$$

   (tf is short for *term frequency*.)

2. *Term frequency-inverse document frequency (tf-idf)* weighting.

   This is like the unigram representation, except the feature associated with a word w in a document d from a collection of documents D (e.g., training data) is

   $$\mathrm{tf}(w; d) \times \log_{10}(\mathrm{idf}(w; D))\,,$$

   where $\mathrm{tf}(w; d)$ is as defined above, and

   $$\mathrm{idf}(w; D) \;:=\; \frac{|D|}{\text{number of documents in } D \text{ that contain word } w}\,.$$

   This representation puts more emphasis on rare words and less emphasis on common words. (There are many variants of tf-idf that are unfortunately all referred to by the same name.)

   Important: When you apply this representation to a new document (e.g., a document in the test set), you should still use the idf defined with respect to D. This, however, becomes problematic if a word w appears in a new document but did not appear in any document in D: in this case, $\mathrm{idf}(w; D) = |D|/0 = \infty$. It is ambiguous what should be done in these cases, but a safe recourse, which you should use in this assignment, is to simply ignore words w that do not appear in any document in D.

3. Bigram representation.

   In addition to the unigram features, there is a feature for every *pair* of words (w_1, w_2) (called a *bigram*), and the feature value associated with a bigram (w_1, w_2) in a given document d is

   $$\mathrm{tf}((w_1, w_2); d) \;:=\; \text{number of times bigram } (w_1, w_2) \text{ appears consecutively in document } d\,.$$

   In the sequence of words "a rose is a rose", the bigrams that appear are: $(\mathrm{a}, \mathrm{rose})$, which appears twice; $(\mathrm{rose}, \mathrm{is})$; and $(\mathrm{is}, \mathrm{a})$.

4. Another data representation of your own choosing or design.

   Some examples:

- Extend the bigram representation to n-gram representations (for any positive integer n) to consider n-tuples of words appearing consecutively in documents.
- Extend to *k-skip-n-grams*: instead of just counting the number of times the n-tuples $(w_1, w_2, \ldots, w_n)$ appears *consecutively* in the document, also count occurrences where w_i and w_{i+1} are at most k words apart.
- Use variants of tf-idf weighting, such as one that replace $\text{tf}(w; d)$ with $1 + \log_{10}(\text{tf}(w; d))$. This is better for documents where words often appear in *bursts*.
- Select specific words or n-grams you think are informative for the classification task.

The learning methods to try are the following.

1. *Averaged-Perceptron* (with some modifications described below).

   Averaged-Perceptron is like Voted-Perceptron (which uses Online Perceptron), except instead of forming the final classifier as a weighed-majority vote over the various linear classifiers, you simply form a single linear classifier by taking a weighted average the weight vectors and thresholds of all the linear classifiers used by Online Perceptron.

   Use the following two modifications:

   - Run Online Perceptron to make *two* passes through the data. Before each pass, randomly shuffle the order of the training examples. Note that a total of $2n + 1$ linear classifiers are considered during the run of the algorithm (where n is the number of training data).
   - Instead of averaging the weights and thresholds for all $2n + 1$ linear classifiers, just average these things for the *final* $n + 1$ *linear classifiers*.

   You should use Averaged-Perceptron with all four data representations.

2. Naïve Bayes classifier with parameter estimation as in the previous homework assignment.

   Note that with this learning method, a word that appears more than once in a document is treated the same as appearing exactly once.

   You only need to use Naïve Bayes with the unigram representation.

3. (Optional.) Any other learning method you like.

You must write your own code to implement Averaged-Perceptron and to carry out the experiments. **The code should be easy to understand (e.g., by using sensible variable names and comments).** For other things (e.g., Naïve Bayes, cross validation), you can use your own or existing library implementations, but you are responsible for the correctness of these implementations as per the specifications from the course lectures. Provide references for any such third-party implementations you use (e.g., specific scikit-learn or MATLAB functions that perform cross validation).

What to submit:

1. A concise and unambiguous description of the fourth data representation you try (and also of any additional learning methods you try).
2. Cross-validation error rates for all data representations / learning methods combinations.
3. Name of the method ultimately selected using the cross validation procedure.
4. Training and test error rates of the classifier learned by the selected method.
5. Source code and scripts that you write yourself (in separate files).

Problem 3 (MLE; 10 points)**.**

(a) Consider the statistical model $\mathcal{P} = \{P_{\boldsymbol{\mu},\sigma^2} : \boldsymbol{\mu} \in \mathbb{R}^d, \sigma^2 > 0\}$, where $P_{\boldsymbol{\mu},\sigma^2}$ is the multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\sigma^2 \boldsymbol{I}$. Give a formula for the MLE of σ^2 given the data $\{\boldsymbol{x}_i\}_{i=1}^n$ from $\mathbb{R}^d$, which is regarded as an iid sample. Also give a clear derivation of the formula in which you briefly justify each step.

(b) Consider the statistical model $\mathcal{P} = \{P_\theta : \theta \in \mathbb{N}\}$, where P_θ is the distribution of the random pair (X, Y) in $\mathbb{N} \times \{0, 1\}$ such that:

- X is uniformly distributed on $\{1, 2, \ldots, \theta\}$;
- for any $x \in \{1, 2, \ldots, \theta\}$, the conditional distribution of Y given $X = x$ is specified by $P_\theta(Y = 1 \mid X = x) = x/\theta$.

Give a formula for the MLE of θ given a *single* observation $(x, y) \in \mathbb{N} \times \{0, 1\}$. Also give a clear derivation of the formula in which you briefly justify each step.