## Homework 5, due Monday November 28 COMS 4771 Fall 2016

**Problem 1** (Non-linear classifiers; 20 points)**.** For this problem, you will need to learn to use software libraries for at least two of the following non-linear classifier types:

- neural networks;
- boosted decision trees (i.e., boosting, with the "weak learner" being a decision tree learner);
- random forests.

All of these are available in scikit-learn and the MATLAB Statistics and Machine Learning toolbox, although you may also use other external libraries (e.g., Tensorflow[1] for neural networks, XGBoost[2] for boosted decision trees). You are welcome to implement learning algorithms for these classifiers yourself, but this is neither required nor recommended.

Learn two different types of non-linear classifiers from above for the HW3 restaurant review data, with any feature representation you like. You can use the software libraries mentioned above, and as well as libraries & tools (or your old code from HW3) to perform the feature processing. You should try to match (and hopefully surpass) the results obtained by the "modified" Averaged-Perceptron with bigram features from HW3, which had five-fold cross-validation error rate around 8.8%. You may sub-sample the training data if you like, but note that using more training data usually improves performance (up to a point).

For many of these learning algorithms, you will need to set various hyperparameters (including classifier "architecture", like the number of layers and width of each layer in a neural network, and the number of trees in a random forest). Often there are defaults that make a good starting point, but you may need to adjust at least some of them to get good performance. Use hold-out validation or $K$-fold cross-validation to do this. **Do not make any hyperparameter choices (or any other similar choices) based on the test set!** You should only compute the test error rates after you have settled on hyperparameter settings and trained your two final classifiers.

What to submit:

1. Names of the two types of classifiers you opt to learn.
2. Proper citations for any external code you use. See `https://integrity.mit.edu/handbook/writing-code` for guidelines.
3. Description of your training methodology, with enough details so that another machine learning enthusiast can reproduce the your results.
4. The final hyperparameter settings you use.
5. Training error rates, hold-out or cross-validation error rates, and test error rates for your two final classifiers.

No need to submit any code.

[1] `https://www.tensorflow.org`

[2] `https://github.com/dmlc/xgboost/tree/master/python-package`

**Problem 2** (Conditional probability estimation; 15 points)**.**

(a) Suppose $(X, Y)$ is a random pair taking values in $\mathcal{X} \times \{-1, 1\}$. Let $\eta\colon \mathcal{X} \to [0, 1]$ be the conditional probability function, given by $\eta(x) := \mathbb{P}(Y = 1 \mid X = x)$. What function $f\colon \mathcal{X} \to \mathbb{R}$ minimizes $\mathbb{E}[\ell(Yf(X))]$ where $\ell(z) := e^{-z}$? Give your answer in terms of $\eta$ (e.g., $f(x) = \eta(x)$).

(b) Download the data set `hw5data.mat` from Courseworks (which has training feature vectors and labels, stored as `data` and `labels`, respectively). Compute the maximum likelihood estimates of the logistic regression model parameters $(\beta_0, \boldsymbol{\beta}) \in \mathbb{R} \times \mathbb{R}^d$. You may use your code from the previous homework assignment, or code from an existing software library for logistic regression. It should be possible to obtain estimates with objective value (from the previous homework) not larger than 0.50317.

The data set `hw5data.mat` also contains test feature vectors and labels, stored as `testdata` and `testlabels`, respectively. With your estimated parameters $(\hat{\beta}_0, \hat{\boldsymbol{\beta}}) \in \mathbb{R} \times \mathbb{R}^d$ in hand, you can compute the conditional probability

$$P_{(\hat{\beta}_0, \hat{\boldsymbol{\beta}})}(Y = 1 \mid \boldsymbol{X} = \boldsymbol{x}) \;=\; \frac{1}{1 + \exp(-\hat{\beta}_0 - \langle \hat{\boldsymbol{\beta}}, \boldsymbol{x} \rangle)}$$

for each row $\boldsymbol{x}^\top$ in `testdata`.

It turns out each row of `testdata` actually appears 128 times in `testdata`: for each $i = 1, 2, \ldots, 1024$, the `testdata` rows $\{\boldsymbol{x}_{1024k+i}^\top : k = 0, 1, \ldots, 127\}$ are identical. However, the corresponding entries in `testlabels` are not repeated in the same way: for each $i = 1, 2, \ldots, 1024$, the actual fraction of labels equal to one is $p_i := \frac{1}{128} \sum_{k=0}^{127} y_{1024k+i} \notin \{0, 1\}$.

Compare $P_{(\hat{\beta}_0, \hat{\boldsymbol{\beta}})}(Y = 1 \mid \boldsymbol{X} = \boldsymbol{x}_i)$ to $p_i$ by computing the mean absolute error (MAE)

$$\text{MAE} \;:=\; \frac{1}{1024} \sum_{i=1}^{1024} \left| P_{(\hat{\beta}_0, \hat{\boldsymbol{\beta}})}(Y = 1 \mid \boldsymbol{X} = \boldsymbol{x}_i) - p_i \right|.$$

(This is one way to evaluate conditional probability predictions; there are many others.)

What to submit in your write-up:

1. MAE of conditional probability predictions.
2. Proper citations for any external code you use.

No need to submit any code.

(c) *Optional.* The data from part (b) was generated by a distribution for which the Bayes classifier is a linear classifier, but is not one from the logistic regression model. Try a different approach to get better conditional probability predictions on the data from part (b).

What to submit in your write-up:

1. MAE of conditional probability predictions.
2. Proper citations for any external code you use.
3. A description of the approach you tried.

No need to submit any code.

**Problem 3** (Linear regression; 15 points)**.**

(a) Let $S := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ be a data set from $\mathbb{R}^d \times \mathbb{R}$, and let $\mathcal{P} = \{P_{(\boldsymbol{w},\sigma^2)} : \boldsymbol{w} \in \mathbb{R}^d, \sigma^2 > 0\}$ be the "classical statistical model for linear regression" discussed in lecture.

Let $\boldsymbol{A}$ be the $n \times d$ matrix whose $i$-th row is $\boldsymbol{x}_i^\top$, let $\boldsymbol{y}$ be the vector whose $i$-th entry is $y_i$. Suppose $S$ really is an iid sample from a distribution $P_{(\boldsymbol{w}_\star,\sigma_\star^2)} \in \mathcal{P}$. Assume that $\boldsymbol{A}$ has rank $d$, so $\boldsymbol{A}^\top\boldsymbol{A}$ is invertible and $\hat{\boldsymbol{w}}_{\text{ols}}$ exists; define $\hat{\boldsymbol{y}} := \boldsymbol{A}\hat{\boldsymbol{w}}_{\text{ols}}$.

For each of the following assertions, state whether it is true or false, and briefly justify your answer.

1. $\text{var}(y_i \mid \boldsymbol{x}_i) = \sigma_\star^2$.
2. $\text{cov}(\boldsymbol{w}_\star \mid \boldsymbol{A}) = \sigma_\star^2(\boldsymbol{A}^\top\boldsymbol{A})^{-1}$.
3. $\text{cov}(\hat{\boldsymbol{w}}_{\text{ols}} \mid \boldsymbol{A}) = \sigma_\star^2(\boldsymbol{A}^\top\boldsymbol{A})^{-1}$.
4. $\mathbb{E}(\hat{\boldsymbol{y}} \mid \boldsymbol{A}) = \boldsymbol{A}\boldsymbol{w}_\star$.
5. $\boldsymbol{r} := \boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{w}}_{\text{ols}}$ is the orthogonal projection of $\boldsymbol{y}$ onto the null space of $\boldsymbol{A}^\top$.
6. If every entry in the first column of $\boldsymbol{A}$ is 1, then $\sum_{i=1}^n r_i = 0$ (where $r_i$ is the $i$-th entry of $\boldsymbol{r}$, defined above).

(b) Download the Boston housing data set `housing.mat` from Courseworks. The first feature is a constant feature, always equal to one: this is just here to simplify estimation of the "intercept" term. The next 13 features are `CRIM`, `ZN`, ..., `LSTAT`, as described in `https://archive.ics.uci.edu/ml/datasets/Housing`; note that standardization (based on the training data) has been applied (to both the training data and test data). The output (label) is `MEDV`, the median value of owner-occupied homes (in units of $1000).

Compute the ordinary least squares (OLS) estimator based on the training data (`data` and `labels`). You can use whatever software libraries you like to do this. Compute the average squared loss of the OLS estimator on the test data (`testdata` and `testlabels`).

Use some existing software package to compute a *sparse* weight vector with at most three non-zero entries (not including the "intercept") based on the training data. Compute the average squared loss of this sparse linear predictor on the test data (`testdata` and `testlabels`).

Some suggested methods to use: Lasso, LARS (which is actually an algorithm for solving the Lasso optimization problem with some additional convenient properties), stepwise regression, orthogonal matching pursuit.

What to submit in your write-up:

1. Test average squared loss of OLS estimator.
2. Test average squared loss of the sparse linear predictor.
3. Names of the variables with non-zero entries in the sparse linear prediction. Report the actual variable names[3] (e.g., `CRIM`, `ZN`, `INDUS`).
4. Proper citations for any external code you use.

No need to submit any code.

[3]The names of the variables are given in `https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.names`.