CMPS 242 Third Homework, Fall 2015

4 Problems, 14 pts, due start of class Wednesday, Oct. 28

This homework is to be done in groups of 3 (permission for other sizes can be given by the instructor or TA). Do not use the same exact group as a previous homework. Each group member should completely understand the group's solutions and the group *must* acknowledge all sources of inspiration, techniques, and/or helpful ideas (web, people, books, etc.) other than the instructor, TA, and class text. Each group should submit a single set of solutions containing the names and e-mail addresses of all group members. Although there are no points for "neatness", the TA may deduct points for illegible or poorly organized solutions.

1. (4 pts) Linear Regression

   The following arff file describes the data for this problem:

   ```
   —
   @relation 'hw3'


   @attribute 'x1' real
   @attribute 'x2' real
   @attribute 'x3' real
   @attribute 't' real
   @data
   3, 9, 2, 19
   6, 9, 1, 19
   7, 7, 7, 10
   8, 6, 4, 11
   1, 0, 8, -3
   —
   ```

   There are 3 features: x_1, x_2, x_3. The target is t. There are 5 instances. The first instance has features $x_1 = 3, x_2 = 9, x_3 = 2$ and target $t = 19$.

   Note: This is artificial & slightly cherry picked data. For each instance i, the features $\mathbf{x}_i$ were generated randomly and then noise was added. The targets were generated by the following function before the noise: $t = 0 * x_{1\text{ (clean)}} + 2 * x_{2\text{ (clean)}} - 1 * x_{3\text{ (clean)}} + 3$. Your computed weight vector should be somewhat similar to this, but with so few examples and so much relative noise you shouldn't expect to get really close.

   (a) Save the data to an arff file and run the linear regression algorithm in Weka on the full training set. Report the model and "Root mean squared error". Note that testing on the training set means we will have a relatively small error.

   (b) Suppose you had an unlabeled instance $\mathbf{x} = [3, 3, 5]$. What $\hat{t}$ prediction for t would the model from part (a) give?

(c) The "ridge parameter" is another name for the $L2$ regulariztion coeficient (often written λ). Re-run the linear regression in Weka with a larger ridge parameter, say 0.2. What are learned weights? Is the change qualitatively what you would expect?

(d) Compute regression weights using eq. 3.34 from Bishop. In our notation that is: $\mathbf{w} = (X^\mathsf{T} X)^{-1} X^\mathsf{T} \mathbf{t}$, here X is the (5-row, 4-column) matrix whose rows are the unlabeled instances and whose columns are each of the features for the instance augmented by the fixed feature $x_0 = 1$ which will lead to the bias. How does this $\mathbf{w}$ compare to the weights from part a? If you like, you may use your favorite math software for the matrix arithmetic but it is quite feasible to do directly. You should not compute the inverse by hand (Google "inverse matrix 4x4 calculator").

(e) If the examples are re-ordered (so the rows of X and elements of $\mathbf{t}$ are permuted), what happens to the learned $\mathbf{w}$ vector and why?

2. (4 pts) Weka Experiments. The purpose of this problem is to familiarize you with Weka, gain experience with three fundamental algorithms, and to consider how changes in the data or experimental protocol are likely to affect the results. *I strongly suggest that each student do this part on their own, and combine answers for the group solutions.* Open the diabetes.arff file (in the `data` directory distributed with Weka) and read the comments at the top (lines beginning with "`%`").

   For this problem you will be running Weka's Nearest Neighbor (`IB1` in the `lazy` folder), `NaiveBayes` (in the `bayes` folder), and logistic regression (`Logistic` in the `functions` folder, see also Section 4.3.2 in Bishop) classifiers.

   (a) Select the `Use training set` test option and run the three classifiers. Report their results (accuracies). Which algorithm is best and **why**?

   (b) For your logistic regression model from the previous step, give the decision boundary as a linear formula (of the form $\sum_i w_i * x_i + bias = 0$ where i is the index of an attribute). You can find the predictions made on the data by using the "more options" button in the test options window and selecting the "output predictions" box. You can then check your linear formula by finding data points whose probability distribution is close to 0.50/0.50 and verifying that they are close to the decision boundary. (Weka outputs one logistic regression model based on the entire data set).

   (c) Repeat part a) using 10-fold Cross-validation as the test option. What changes about the accuracies? Continue to use 10-fold Cross-validation for the rest of the problem.

   (d) Use preprocessing to "normalize" the features (use the preprocess tab and select unsupervised, attribute, normalize). Read the information on this method, and look at the new attribute values. What did it do?

Rerun the logistic regression with 10-fold cross validation and the attributes normalized. Did the accuracies change? Why?

Are there any dramatic changes in the logistic regression weight vector? Why?

(Note: attribute normalization is so important to nearest neighbor that IB1 normalizes the attributes automatically.)

(e) In logistic regression, the ridge parameter penalizes large weights. What happens to the cross validation accuracy and hypothesis weights when it is set to 0? How about when it is increased (to say 0.3)?

(f) Would you expect 3NN or 5NN do better than Nearest Neighbor? Why? Test your hypothesis by using `IBk` in the `lazy` folder and report the resulting accuracies.

(g) Create a modified version of the diabetes dataset by picking one attribute at random and adding 10 additional copies of that attribute to the data set (or .arff file). There should be the same number of examples, but each example will now have 19 rather than 9 attributes (including the class label). How would you expect the 10-fold cross validation accuracies of the classifiers to change? Run the classifiers on the modified .arff file and report the changes.

(h) Create a second modified version of the (original) diabetes.arff file, this time adding 20 random (0,1)-valued attributes to the file. Re-run the algorithms on this version of the data (with pre-processing to normalize the features) and report how the accuracies changed.

3. (4 pts) Perceptron algorithm with noise experiment. Implement the Perceptron algorithm presented in class, where $\mathbf{w}_{\text{new}} := \mathbf{w}_{\text{old}} + t\mathbf{x}$ when the algorithm makes a mistake for examples with 15 features. You may use any appropriate programming language/system you are familiar with. Create a set of 500 instances where the instances are selected uniformly from the boolean hyper-cube $\{\pm 1\}^{15}$ (i.e. each feature is $+1$ with probability 1/2 and -1 with probability 1/2).

(a) Label your 500 instances with the first component of the instances, so $t_i = x_{i,1}$ to create a labeled sample. Report how many epochs (iterations through the training set) and prediction errors occur before producing a hypothesis that correctly labels the training set.

(b) Re-label the 500 instances training set so that $t_i = +1$ if at least half the features in $\mathbf{x}_i$ are $+1$ (and $t_1 = -1$ if at least half the features are -1. Run the perceptron algorithm on the modified training set and report how many epochs and prediction errors occur before producing a hypothesis that correctly labels the training set. (Would you expect this to take more or fewer epochs than the first part? Why?)

(c) Create a third training set with label noise from the 500 instances. Set each label t_i to $\text{sign}(r + \sum_{j=1}^{11} x_{i,j})$ where r is a random number uniformly distributed between -4 and 4 (note that the sum ranges over only the first 11 features). Run two epochs through your training set, keeping track of the 1000 $\mathbf{w}$ vectors produced by the algorithm, call these vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{1000}$ (note that if $\mathbf{w}_j$ will be the same as $\mathbf{w}_{j+1}$ if $\mathbf{w}_j$ correctly predicts the jth instance's label). Consider creating several hypotheses created from the two-epoch run.

i. The *last hypothesis* predicts with $\text{sign}(\mathbf{w}_{1000} \cdot \mathbf{x})$, using the last weight vector from the run.
ii. The *average hypothesis* computes $\bar{\mathbf{w}} = \sum_{j=1}^{1000} \mathbf{w}_j / 1000$ and predicts with $\text{sign}(\bar{\mathbf{w}} \cdot \mathbf{x})$.
iii. The *voted hypothesis* votes all 1000 weight vectors, predicting with (if the vote is tied, treat it as an incorrect prediction).
iv. The *last epoch average* computes $\bar{\mathbf{w}}_\ell = \sum_{j=501}^{1000} \mathbf{w}_j / 500$ and predicts with $\text{sign}(\bar{\mathbf{w}}_\ell \cdot \mathbf{x})$.
v. The *last epoch vote* predicts with the vote sign $\left(\sum_{i=501}^{1000} \text{sign}(\mathbf{w}_i \cdot \mathbf{x})\right)$

Which of these do you think will be more accurate on new unseen data? Create a separate test set of 500 examples (with the random noise) and evaluate the accuracy of these five hypotheses on it. Ambitious students may run this experiment multiple times (say 10) and report the average accuracies.

4. (2 pts) Logistic Regression. Recall that in the book's notation (without the ϕ's): $y_n = p(\text{class1} \mid \mathbf{x}_n) = \sigma(\mathbf{w}^\mathsf{T}\mathbf{x}_n)$ where $\sigma()$ is the logistic sigmoid, $\sigma(a) = \dfrac{1}{1+\exp(-a)}$, and the cross-entropy error function is:

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \left(t_n \ln y_n + (1-t_n)\ln(1-y_n)\right).$$

Show that the gradient of the cross-entropy error is

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\mathbf{x}_n.$$

(i.e. Show that the gradient of (4.90) in the text is given by (4.91).) You may use (4.88) in your solution.