EECS 545 – Machine Learning - Homework #6

Changhan (Aaron) Wang

Due: 11:00pm 04/18/2016

Homework Policy: Working in groups is fine, but each member must submit their own writeup. Please write the members of your group on your solutions. There is no strict limit to the size of the group but we may find it a bit suspicious if there are more than 4 to a team. **For coding problems, please include your code and report your results (values, plots, etc.)** in your PDF submission. You will lose points if your experimental results are only accessible through rerunning your code. Homework will be submitted via Gradescope (https://gradescope.com/).

1) **Principal Components Analysis (20 pts).**

In Principal Components Analysis (PCA), we project the data $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N | \boldsymbol{x}_i \in \mathbb{R}^D\}$ into K ($K < D$) orthogonal directions, which maximizes the following projection variance:

$$\max_{\substack{A \\ A^TA=I_k}} \sum_{k=1}^{K} \boldsymbol{a}_k^T S \boldsymbol{a}_k \tag{1}$$

where $S = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \bar{x})(\boldsymbol{x}_i - \bar{x})^T \in \mathbb{R}^{D\times D}$ is the data covariance matrix, transformation matrix $A = \begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_K \end{bmatrix} \in \mathbb{R}^{D\times K}$ and $\boldsymbol{a}_k^T \boldsymbol{x}_i$ is the projection of the i-th data point into the k-th direction. Suppose S has the eigenvalue decomposition $S = U\Lambda U^T$ where $U = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_D \end{bmatrix} \in \mathbb{R}^{D\times D}$ and $U^TU = I_D$; diagonal matrix $\Lambda = diag(\lambda_1, \lambda_2, \cdots, \lambda_D)$ and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$. Denote $\boldsymbol{w}_k = U^T\boldsymbol{a}_k$ and $W = \begin{bmatrix} \boldsymbol{w}_1 & \boldsymbol{w}_2 & \cdots & \boldsymbol{w}_K \end{bmatrix} \in \mathbb{R}^{D\times K}$, then we get the following optimization problem from (1):

$$\max_{\substack{W \\ W^TW=I_k}} \sum_{k=1}^{K} \boldsymbol{w}_k^T \Lambda \boldsymbol{w}_k \tag{2}$$

(a) We denote

$$h_j = \sum_{k=1}^{K} (\boldsymbol{w}_k^{(j)})^2$$

i.e., the square of L_2 norm of the j-th row vector in W. Prove that $0 \leq h_j \leq 1$ and $\sum_{j=1}^{D} h_j = K$.

(b) Prove that

$$\max_{\substack{W \\ W^TW=I_k}} \sum_{k=1}^{K} \boldsymbol{w}_k^T \Lambda \boldsymbol{w}_k = \max_{\substack{h_j \\ W^TW=I_k}} \sum_{j=1}^{D} h_j \lambda_j \tag{3}$$

(c) What are the optimal h_j in (3)? Show that $\boldsymbol{a}_k = \boldsymbol{u}_k$ ($k = 1, \cdots, K$) is a solution of (3).

(d) Is the solution of (3) unique? Give a necessary and sufficient condition for the subspace spanned by the solution $\{\boldsymbol{a}_1^*, \boldsymbol{a}_2^*, \cdots, \boldsymbol{a}_K^*\}$ to be unique.

(e) We can construct the solution $\boldsymbol{a}_k = \boldsymbol{u}_k$ $(k = 1, \cdots, K)$ in an iterative way. We notice that $\boldsymbol{a}_1 = \boldsymbol{u}_1$ is a solution for

$$\max_{\substack{\boldsymbol{a}_1 \\ \boldsymbol{a}_1^T \boldsymbol{a}_1 = 1}} \quad \boldsymbol{a}_1^T S \boldsymbol{a}_1$$

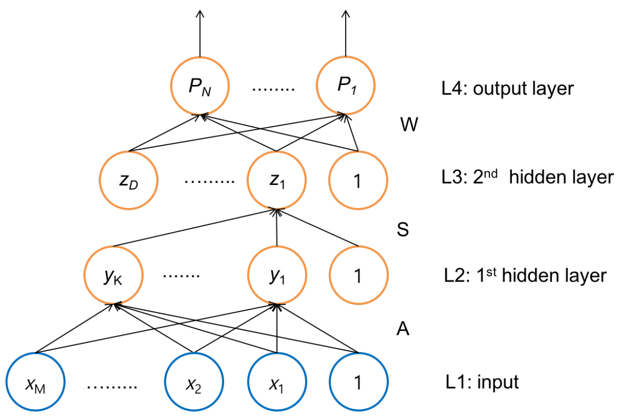
which is a special case (K=1) in (3).

Show that if $\boldsymbol{a}_k$ $(k = 2, \cdots, K)$ is orthogonal to $\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_{k-1}$ (i.e. $\boldsymbol{a}_k^T \boldsymbol{u}_i = 0$ where $i = 1, ..., k-1$), then $\boldsymbol{a}_k = \boldsymbol{u}_k$ is a solution of

$$\max_{\substack{\boldsymbol{a}_k \\ \boldsymbol{a}_k^T \boldsymbol{a}_k = 1 \\ \boldsymbol{a}^T \boldsymbol{u}_i = 0, i=1,\ldots,k-1}} \quad \boldsymbol{a}_k^T S \boldsymbol{a}_k$$

2) **Multi-layer Neural Network (20 pts).**

In this question, we will implement a 4-layer neural network with Softmax output.



The input layer (L1) contains M+1 units including a bias term where M is the dimensionality of the data. The 1st hidden layer (L2) has K+1 hidden units including a bias term. The connection between L1 and L2 is represented by matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,M+1} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,M+1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{K,1} & a_{K,2} & \cdots & a_{K,M+1} \end{pmatrix}$$

L2 uses Sigmoid activation function

$$h(t) = \frac{1}{1 + e^{-t}}$$

and the output of y_i is therefore

$$y_i = h(a_{i,M+1} + \sum_{j=1}^{M} a_{i,j} x_j)$$

Similarly, the 2nd hidden layer (L3) contains D+1 hidden units including a bias term. The connection is denoted as matrix **S**:

$$\mathbf{S} = \begin{pmatrix} s_{1,1} & s_{1,2} & \cdots & a_{1,K+1} \\ s_{2,1} & s_{2,2} & \cdots & a_{2,K+1} \\ \vdots & \vdots & \ddots & \vdots \\ s_{D,1} & s_{D,2} & \cdots & a_{D,K+1} \end{pmatrix}$$

L3 also uses Sigmoid activation function and the output of z_i is therefore

$$z_i = h(s_{i,K+1} + \sum_{j=1}^{K} s_{i,j} y_j)$$

The output layer (L4) contains N Softmax units where N is the number of classes. The connection between L3 and L4 is denoted by the matrix **W**:

$$\mathbf{W} = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,D+1} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,D+1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N,1} & w_{N,2} & \cdots & w_{N,D+1} \end{pmatrix}$$

The Softmax output P_i is

$$P_i = \frac{\exp(\sum_{j=1}^{D} w_{i,j} z_j + w_{i,D+1})}{\sum_{k=1}^{N} \exp(\sum_{j=1}^{D} w_{k,j} z_j + w_{k,D+1})}$$

For a data point $\mathbf{x} \in \mathbb{R}^M$ and its label $t \in \{0, 1, \cdots, N-1\}$, the loss function is defined as

$$E(\mathbf{W}, \mathbf{S}, \mathbf{A}) = -\sum_{i=1}^{N} 1(t+1=i) \log(P_i)$$

where $1(\cdot)$ is the indicator function.

(a) Derive the gradients: $\nabla_{\mathbf{W}} E$, $\nabla_{\mathbf{S}} E$ and $\nabla_{\mathbf{A}} E$.

(b) Implement back-propagation algorithm and perform gradient checking (`http://deeplearning.stanford.edu/wiki/index.php/Gradient_checking_and_advanced_optimization`) to verify your gradient computation. Use the following setting in your experiment (please refer to `q2_starter.ipynb`):

- $M = 100$, $K = 50$, $D = 30$, $N = 10$
- We will use fake data: sample $\boldsymbol{x}$ from $[-0.05, 0.05]^M$ and t from $\{0, 1, ..., N-1\}$
- Sample **A**, **S**, **W** from $[-0.05, 0.05]^{K \times (M+1)}$, $[-0.05, 0.05]^{D \times (K+1)}$ and $[-0.05, 0.05]^{N \times (D+1)}$, respectively
- Gradient checking: fix **A**, **S**, **W**, $\boldsymbol{x}$ and t. Sample row index r and column index c 1000 times (e.g. for 50x101 matrix **A**, sample r from $\{0, 1, ..., 49\}$ and c from $\{0, 1, ..., 100\}$). For each sample, alter the matrix entry at (r, c) by $\pm\epsilon$ ($\epsilon = 10^{-4}$) to compute the gradient (its entry at (r, c)) numerically.

For each of the gradients $\nabla_{\mathbf{W}} E$, $\nabla_{\mathbf{S}} E$ and $\nabla_{\mathbf{A}} E$, report the mean absolute error (MAE) in gradient checking using the 1000 samples (comparing your implemented gradients to the numerical versions).

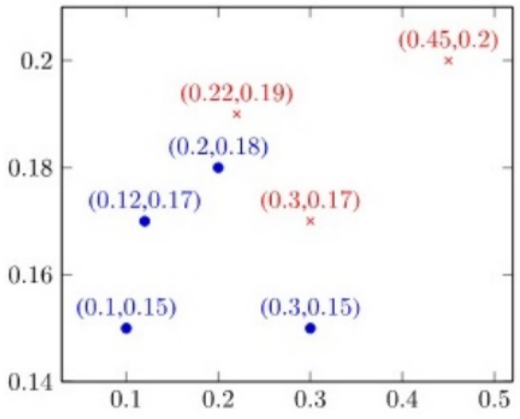
3) **Open Kaggle Challenge (20 pts).** You can use any features and any models to perform classification on the tiny image dataset. Please refer to `https://inclass.kaggle.com/c/tiny-image-classification` for details (join the competition at `https://kaggle.com/join/eecs545hw6`). This problem will be graded separately based on your performance on the leaderboard.

4) **AdaBoost (20 pts).**

Boosting combines a set of base classifiers into a stronger ensemble classifier $h_m(\bar{x}) = \sum_{j=1}^{m} \alpha_j h(\bar{x}; \bar{\theta}_j)$, where $\alpha_j \geq 0$ are non-negative votes allocated to each base classifier (decision stump). $h(\bar{x}; \bar{\theta}_j) = sgn(\theta_1^{(j)} x_k + \theta_0^{(j)})$ described a parameter vector $\bar{\theta}_j = \{k, \theta_1^{(j)}, \theta_0^{(j)}\}$ that encodes the co-ordinate, direction and location information. Finding a jointly optimum solution of $\bar{\theta}_j$ and α_j for all j is a hard problem and therefore, we take a sequential learning approach exemplified by the adaptive boosting (AdaBoost) algorithm:

```
Set $W_0(i) = \frac{1}{n}$ for $i = 1, \cdots, n$
for $m = 1$ to $M$ do:
  find $h(\bar{x}; \bar{\theta}_m)$ that minimizes the weighted training error $\epsilon_m$:
    $\epsilon_m = \sum_{i=1}^{n} W_{m-1}(i) 1(y^{(i)} \neq h(\bar{x}; \bar{\theta}_m))$
  given $\bar{\theta}_m$, compute $\alpha_m$ that minimizes weighted training loss:
    $\alpha_m = \frac{1}{2} \log(\frac{1-\epsilon_m}{\epsilon_m})$
  update weights on all training examples:
  for $i = 1$ to $n$ do:
    $W_m(i) = c_m W_{m-1}(i) \exp\{-y^{(i)} \alpha_m h(\bar{x}; \bar{\theta}_m)\}$   where $c_m$ is the normalizer of $W_m(\cdot)$
  end for
end for
```

Recall that we define the training error of a classifier h as $E(h) = \sum_{t=1}^{n} 1(y^{(t)} h(\bar{x}^{(t)}) < 0)$, while the exponential loss function often used in training a boosted classifier is given by $L(h) = \sum_{t=1}^{n} \exp(-y^{(t)} h(\bar{x}^{(t)}))$. Consider the following points: blue circles are positive examples, and red crosses negative.



(a) Determine α_1 and $\tilde{W}_1(i)$ $(i = 1, 2, ..., 7)$ generated by one iteration of the Adaboost algorithm applied to the above points, using the stump classifier $h(\bar{x}) = sign(-x_2 + 0.17)$. Assume uniform initial weights.

(b) Determine a boosted combination of decision stumps that correctly classifies the above points. What is the corresponding exponential loss?

(c) Suppose you have many boosted classifiers that correctly classify your training set. Is there an advantage to picking the classifier that minimizes exponential loss, and if so, why?

5) **Decision trees and random forest (20 pts).**

In this question, we will implement the following functions in `q5_starter.ipynb`: `plot_error`, `random_forest` and `bagging_ensemble`.

(a) First, we will study decision trees on the Iris flower dataset, which consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Each sample is described by 4 features: the length and the width of the sepals and the length and the width of the petals. We will use only 2 features sepal length and sepal width.

Implement decision tree classifiers using `sklearn`. Let `criterion='entropy'`, so that our selection of which node to split on depends on the information gain. In addition, we will use `max_depth`, the maximum allowed depth of the decision tree, to avoid over-complex trees that overfit the data.

Implement `plot_error(X, y)` and generate a plot of 5-fold cross-validation training and test error vs. depth of the trees by varying `max_depth` from 1 to 20. (Split the data and labels in 5-folds using `sklearn.cross_validation.StratifiedKFold` and train decision trees on 4 folds.) For which value of `max_depth`, does the classifier perform the best?

(b) Now, we will study ensemble approaches, bagging and random forest on a handwritten digit dataset. We will use a subset of 720 examples from 4 classes.

Implement `bagging_ensemble(X_train, y_train, X_test, y_test, n_clf = 10)`. A bagging ensemble classifier consists of `n_clf` decision trees where each decision tree is trained independently on a bootstrap sample of the training data. Here, the final prediction of the bagging classifier is determined by a majority vote of these `n_clf` decision trees.

Implement `random_forest(X_train, y_train, X_test, y_test, n_clf = 10)`. Like bagging, random forest also consists of `n_clf` decision trees where each decision tree is trained independently on a bootstrap sample of the training data. However, for each node we randomly select m features as candidates for splitting on (see parameter `max_features` of `sklearn.tree.DecisionTressClassifier`). Again, here the final output is determined by majority vote.

Now, compare the performance of these ensemble classifiers using 100 random splits of the digits dataset into training and test sets, where the test set contains roughly 20% of the data. Run both algorithms on these data and obtain 100 accuracy values for each algorithm.

How does the average test performance of the two methods compare as we vary m? Choose a setting for m based on your observations and plot the result as two histograms (we've provided you with a function for plotting the histograms).