Homework 3

Aleck Zhao

April 13, 2018

1 Analytical (50 points)

1. Deep Neural Networks (12 points)

(a) Consider a 2-layer neural network, with M input nodes, Z nodes in the hidden layer and K nodes in the output layer. The network is fully connected, i.e. every node in the $n-1$th layer is connected to every node in the nth layer. However, for your application of interest, you suspect that only some of the nodes in the input are relevant. How would you modify the objective function to reflect this belief?

Answer. We could modify the objective function by adding a regularization term of the form $\lambda w^T w$. This will force some of the weights to be zero, which is equivalent to saying those features (input nodes) are not relevant.

(b) Consider a N layer neural network. We could (a) train the entire network at once using back-propagation or (b) pre-train each layer individually, and then tune the final network with back-propagation. Will (a) and (b) converge to the same solution? Why would we favor strategy (a) vs. strategy (b)?

Answer. (a) and (b) will not necessarily converge to the same solution. The purpose of pre-training is to try to initialize the weight vector at a better local optimum, whereas back-propagation of the entire network will likely find a worse local optimum.

We would prefer to use pre-training when the network is deep due to the vanishing gradient problem. This allows us to initialize weights that are more "tuned" to the problem, which allows us to avoid vanishing gradients. On the other hand, if we have a lot of labeled data, there may be no need to go through pre-training. Pre-training will also be more computationally difficult, but will probably perform better.

(c) Consider a $N \geq 2$ layer neural network with a single node in the output layer. We wish to train this network for binary classification. Rather than use a cross entropy objective, we want to take a max-margin approach and ensure a margin of $\gamma = 1$. Describe the structure of the last layer of the network, including the final activation function, and the training objective function that implements a max-margin neural network. What are the benefits of this network compared to one trained with cross entropy? Will a max-margin trained neural network learn the same decision boundary as an SVM?

Answer. We can use the binary hinge loss as the objective function, since this is used in the setting of max-margin, with a ReLU activation function. Using hinge loss is computationally more efficient than cross entropy. These two models might not learn the same decision boundary because while SVM has a convex objective function and therefore a distinct global minimum, the neural network objective functions are non-convex with the non-linear activation functions, so the neural network might learn the decision boundary given at a local minimum.

2. Adaboost (12 points) There is one good example at $x = 0$ and two negative examples at $x = \pm 1$. There are three weak classifiers are

$$\begin{aligned} h_1(x) &= 1 \cdot \mathbf{1}(x > 1/2) - 1 \cdot \mathbf{1}(x \leq 1/2), \\ h_2(x) &= 1 \cdot \mathbf{1}(x > -1/2) - 1 \cdot \mathbf{1}(x \leq -1/2) \\ h_3(x) &= 1. \end{aligned}$$

Show that this data can be classified correctly by a strong classifier which uses only three weak classifiers. Calculate the first two iterations of AdaBoost for this problem. Are they sufficient to classify the data correctly?

Solution. We have

$$\mathcal{D} = \{(0, 1), (-1, -1), (1, -1)\}$$

We wish to find constants $\alpha_1, \alpha_2, \alpha_3$ such that

$$h(x_i) = \alpha_1 h_1(x_i) + \alpha_2 h_2(x_i) + \alpha_3 h_3(x_i) = y_i$$

for our three training examples. Evaluating h_1, h_2, h_3 at each of x_1, x_2, x_3, we have the equations

$$\begin{aligned} -\alpha_1 + \alpha_2 + \alpha_3 &= 1 \\ -\alpha_1 - \alpha_2 + \alpha_3 &= -1 \\ \alpha_1 + \alpha_2 + \alpha_3 &= -1 \end{aligned}$$

which has solution $(\alpha_1, \alpha_2, \alpha_3) = (-1, 1, -1)$, so the data is correctly classified by the strong classifier

$$h(x) = -h_1(x) + h_2(x) - h_3(x)$$

Applying the Adaboost algorithm, we initialize $D_1 = \begin{bmatrix} 1/3 & 1/3 & 1/3 \end{bmatrix}$. Now, using h_1 as the first hypothesis, we have

$$\begin{aligned} h_1(x_1) &= -1 \neq 1 = y_1 \\ h_1(x_2) &= -1 = -1 = y_2 \\ h_1(x_3) &= 1 \neq -1 = y_3 \end{aligned}$$

so

$$\varepsilon_1 = \frac{2}{3} \implies \alpha_1 = \frac{1}{2} \log \frac{1/3}{2/3} = -\frac{1}{2} \log 2$$

Now, we update the distribution as

$$\begin{aligned} Z_2 &= \sum_{i=1}^{3} D_1(i) \exp(-\alpha_1 y_i h_1(\mathbf{x_i})) = \frac{1}{3} \exp\left(-\frac{1}{2} \log 2\right) + \frac{1}{3} \exp\left(\frac{1}{2} \log 2\right) + \frac{1}{3} \left(-\frac{1}{2} \log 2\right) \\ &= \frac{1}{3} \left(\frac{1}{\sqrt{2}} + \sqrt{2} + \frac{1}{\sqrt{2}}\right) = \frac{2\sqrt{2}}{3} \\ D_2(1) &= \frac{D_1(1)}{Z_2} \exp\left(-\alpha_1 y_1 h_1(\mathbf{x_1})\right) = \frac{1/3}{2\sqrt{2}/3} \exp\left(-\frac{1}{2} \log 2\right) = \frac{1}{2\sqrt{2}} \cdot \frac{1}{\sqrt{2}} = \frac{1}{4} \\ D_2(2) &= \frac{D_1(2)}{Z_2} \exp\left(-\alpha_1 y_2 h_1(\mathbf{x_2})\right) = \frac{1/3}{2\sqrt{2/3}} \exp\left(\frac{1}{2} \log 2\right) = \frac{1}{2\sqrt{2}} \cdot \sqrt{2} = \frac{1}{2} \\ D_2(3) &= \frac{1}{4} \end{aligned}$$

Now, using h_2 as the second hypothesis, we have

$$\begin{aligned} h_2(x_1) &= 1 = 1 = y_1 \\ h_2(x_2) &= -1 = -1 = y_2 \\ h_2(x_3) &= 1 \neq y_3 \end{aligned}$$

so

$$\varepsilon_2 = \frac{1}{4} \implies \alpha_2 = \frac{1}{2}\log\frac{3/4}{1/4} = \frac{1}{2}\log 3$$

Now, we update the distribution as

$$\begin{aligned} Z_3 &= \sum_{i=1}^{3} D_2(i)\exp\left(-\alpha_2 y_i h_2(x_i)\right) = \frac{1}{4}\exp\left(-\frac{1}{2}\log 3\right) + \frac{1}{2}\exp\left(-\frac{1}{2}\log 3\right) + \frac{1}{4}\exp\left(\frac{1}{2}\log 3\right) \\ &= \frac{1}{4}\frac{1}{\sqrt{3}} + \frac{1}{2}\frac{1}{\sqrt{3}} + \frac{1}{4}\sqrt{3} = \frac{\sqrt{3}}{2} \\ D_3(1) &= \frac{D_2(1)}{Z_3}\exp\left(-\alpha_2 y_1 h_2(x_1)\right) = \frac{1/4}{\sqrt{3}/2}\exp\left(-\frac{1}{2}\log 3\right) = \frac{1}{2\sqrt{3}}\cdot\frac{1}{\sqrt{3}} = \frac{1}{6} \\ D_3(2) &= \frac{D_2(2)}{Z_3}\exp\left(-\alpha_2 y_2 h_2(x_2)\right) = \frac{1/2}{\sqrt{3}/2}\exp\left(-\frac{1}{2}\log 3\right) = \frac{1}{\sqrt{3}}\cdot\frac{1}{\sqrt{3}} = \frac{1}{3} \\ D_3(3) &= \frac{1}{2} \end{aligned}$$

so our final model is

$$H(x) = \operatorname{sign}\{\alpha_1 h_1(x) + \alpha_2 h_2(x)\} = \operatorname{sign}\left\{\left(-\frac{1}{2}\log 2\right)h_1(x) + \left(\frac{1}{2}\log 3\right)h_2(x)\right\}$$

This model is insufficient to classify our data. Consider the point $(1, -1)$. We have

$$\begin{aligned} & h_1(1) = 1, \quad h_2(1) = 1 \\ \implies & H(1) = \operatorname{sign}\left\{-\frac{1}{2}\log 2 + \frac{1}{2}\log 3\right\} = 1 \neq -1 \end{aligned}$$

□

3. Ensemble Methods (12 points) Consider the following binary classification Boosting algorithm.

1. Given $\{\mathbf{x}_i, y_i\}_{i=1}^N$, number of iterations T, weak learner f.
2. Initialize $\mathcal{D}_0$ to be a uniform distribution over examples.
3. For each iteration $t = 1 \dots T$:
   (a) Train a weak learner f on the data given $\mathcal{D}_t$ to produce hypothesis h_t.
   (b) Compute the error of h_t as $\epsilon_t = P_{\mathcal{D}_t}[h_t(\mathbf{x}_i) \neq y_i]$
   (c) Compute $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
   (d) Update $\mathcal{D}$ as:
   $$\mathcal{D}_{t+1}(i) = \frac{\mathcal{D}_t(i)}{Z_t} \times \begin{cases} \exp(-\alpha_t + (T-t)/T) & \text{if } h_t(\mathbf{x}_i) = y_i \\ \exp(\alpha_t + (T-t)/T) & \text{otherwise} \end{cases}$$
4. Output final hypothesis $H(\mathbf{x}) = \text{sign}\left\{\sum_{t=1}^T \alpha_t h_t(\mathbf{x})\right\}$

Z_t is a normalization constant so that $\mathcal{D}$ is a valid probability distribution.

Describe the difference between this algorithm and the AdaBoost algorithm we learned about in class. What problem of AdaBoost is this change designed to fix? How does changing the algorithm's user provided parameter affect this behavior?

Solution. As written, this algorithm behaves identically to the original AdaBoost algorithm. Here, we have

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \exp\{-\alpha_t y_i h_t(x_i)\} \exp\{(T-t)/T\}$$

where Z_t is the normalizing constant

$$\begin{aligned} Z_t &= \sum_{i=1}^N D_t(i) \exp\{-\alpha_t y_i h_t(x_i)\} \exp\{(T-t)/T\} = \exp\{(T-t)/T\} \sum_{i=1}^N D_t(i) \exp\{-\alpha_t y_i h_t(x_i)\} \\ \implies D_{t+1}(i) &= \frac{D_t(i)}{\exp\{(T-t)/T\} \sum_{i=1}^N D_t(i) \exp\{-\alpha_t y_i h_t(x_i)\}} \exp\{-\alpha_t y_i h_t(x_i)\} \exp\{(T-t)/T\} \\ &= \frac{D_t(i)}{\sum_{i=1}^N \exp\{-\alpha_t y_i h_t(x_i)\}} \exp\{-\alpha_t y_i h_t(x_i)\} \end{aligned}$$

which is exactly the update performed in the original AdaBoost algorithm. Larger values of T will increase the number of hypotheses used, which is expected to improve the test accuracy of AdaBoost since it is resistant to overfitting. □

4. Overfitting in Clustering (14 points) Given the data set $x_1, ..., x_n$, we want cluster the data using the K-means algorithm. The K-means algorithm aims to partition the n observations into k sets ($k < n$) $S = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares

$$\underset{S=\{S_1,\ldots,S_k\}}{\operatorname{argmin}} \sum_{j=1}^{k} \sum_{x_i \in S_j} \|x_j - \mu_j\|_2^2 \tag{1}$$

where μ_j is the mean of points in S_j.

(a) Let γ_k denote the optimal value of the objective function, prove γ_k is non-increasing in k.

Proof. Define

$$f(S) = \sum_{S_j \in S} \sum_{x_i \in S_j} \|x_j - \mu_j\|_2^2$$

to be the objective function evaluated on a partition S. Suppose $S = \{S_1, \cdots, S_k\}$ is the partition that gives the optimal objective function value for k. Suppose we created a $k+1$th cluster centered at any of the data points, WLOG $x_1 \in S_1$. Then for the partition $T = \{S_1 \setminus \{x_1\}, S_2, \cdots, S_k, \{x_1\}\}$, we have

$$\begin{aligned} f(T) &= \sum_{T_j \in T} \sum_{x_i \in T_j} \|x_j - \mu_j\|_2^2 = \sum_{S_j \in S} \sum_{x_i \in S_j} \|x_j - \mu_j\|_2^2 - \|x_1 - \mu_1\|_2^2 + \|x_1 - x_1\|_2^2 \\ &= \gamma_k - \|x_1 - \mu_1\|_2^2 \\ &\leq \gamma_k \end{aligned}$$

Since T is a partition with $k+1$ sets, and γ_{k+1} is the minimum objective function value over all partition with $k+1$ sets, it follows that

$$\gamma_{k+1} \leq f(T) \leq f(S) = \gamma_k$$

and thus γ_k is non-increasing on k. □

(b) Suppose we modified the objective function as follows:

$$\underset{S=\{S_1,\ldots,S_k\}}{\operatorname{argmin}} \sum_{j=1}^{k} \sum_{x_i \in S_j} \max(\|x_j - \mu_j\|_2^2, \tau) \tag{2}$$

where τ is some (given) constant and γ_k' is the optimal value of this new objective function. Compare the values of γ_k and γ_k' ($<, \leq, =, \geq, >$) and prove this relation.

Proof. We claim that $\gamma_k \leq \gamma_k'$. Let $S = \{S_1, \cdots, S_k\}$ be the optimal partition for γ_k', and let $f(S)$ be the objective function for γ_k, and let $g(S, \tau)$ be the objective function for γ_k', both evaluated on the partition S. For any point x_i, we have

$$\begin{aligned} \|x_i - \mu_j\|_2^2 &\leq \max\left\{\|x_i - \mu_j\|_2^2, \tau\right\} \\ \implies f(S) = \sum_{j=1}^{k} \sum_{x_i \in S_j} \|x_i - \mu_j\|_2^2 &\leq \sum_{j=1}^{k} \sum_{x_i \in S_j} \max\left\{\|x_i - \mu_j\|_2^2, \tau\right\} = g(S, \tau) = \gamma_k' \end{aligned}$$

Thus, since $\gamma_k \leq f(S)$, we conclude that $\gamma_k \leq \gamma_k'$.

We can have equality if $\tau \leq \|x_i - \mu_j\|_2^2$ for all i, j. However, the inequality does not necessarily hold in the other direction. Suppose $\gamma_k > \gamma_k'$ for some k and let $S = \{S_1, \cdots, S_k\}$ be the optimal partition for γ_k and let $T = \{T_1, \cdots, T_k\}$ be the optimal partition for γ_k'. Then

$$f(S) = \sum_{j=1}^{k} \sum_{x_i \in S_j} \|x_i - \mu(S_j)\|_2^2 > \sum_{j=1}^{k} \sum_{x_i \in T_j} \max\left\{\|x_i - \mu(T_j)\|_2^2, \tau\right\} = g(T, \tau)$$

for a given τ. Then we have

$$\sum_{j=1}^{k} \sum_{x_i \in T_j} \max\left\{\|x_i - \mu(T_j)\|_2^2, \tau\right\} \geq \sum_{j=1}^{k} \sum_{x_i \in T_j} \|x_i - \mu(T_j)\|_2^2 = f(T)$$

so it follows that $f(T) < f(S)$, which is a contradiction since we assumed that $f(S)$ was minimal. Thus, we have the final relation $\gamma_k \leq \gamma_k'$. □

(c) K-medoids is an algorithm similar to K-means. Both K-means and K-medoids attempt to minimize the squared error but unlike K-means, K-medoids chooses a provided example as a cluster center (medoids) rather than the mean of a subset of the examples. For a given data set $\mathbf{X}$, compare the optimal clusterings produced by K-means and K-medoids $(<, \leq, =, \geq, >)$ and prove this relation.

Proof. Let φ_k be the optimal objective function value for K-medoids on k clusters. We claim that $\gamma_k \leq \varphi_k$. Let $f(S)$ be the objective function value for K-means evaluated on the partition S, and let $g(S, M)$ be the objective function values for K-medoids evaluated on the partition S with set of median points M. Fix k, and let $S = \{S_1, \cdots, S_k\}, M = \{m_1, \cdots, m_k\}$ be the optimal clustering and choice of median for K-medoids, so that $\varphi_k = g(S, M)$.

Consider cluster j. Then if $E(\alpha) = \sum_{x_i \in S_j} \|x_i - \alpha\|_2^2$, it is well known that $\alpha = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i = \mu_j$ minimizes $E(\alpha)$, the squared error. Thus,

$$\sum_{x_i \in S_j} \|x_i - \mu_j\|_2^2 \leq \sum_{x_i \in S_j} \|x_i - m_j\|_2^2$$

$$\implies f(S) = \sum_{j=1}^{k} \sum_{x_i \in S_j} \|x_i - \mu_j\|_2^2 \leq \sum_{j=1}^{k} \sum_{x_i \in S_j} \|x_i - m_j\|_2^2 = g(S, M) = \varphi_k$$

Thus, since $\gamma_k \leq f(S)$, we conclude that $\gamma_k \leq \varphi_k$.

We can have equality if we take $k = N$ so that every data point is its own cluster. In this case, the mean and median of every cluster is the same, and the objective function values are both 0. However, the inequality does not necessarily hold in the other direction. Suppose $\gamma_k > \varphi_k$ for some k and let $S = \{S_1, \cdots, S_k\}$ be the optimal partition for K-means, and let $T = \{T_1, \cdots, T_k\}, M = \{m_1, \cdots, m_k\}$ be the optimal partition and choice of medians for K-medoids. Then

$$f(S) = \sum_{j=1}^{k} \sum_{x_i \in S_j} \|x_i - \mu_j\|_2^2 > \sum_{j=1}^{k} \sum_{x_i \in T_j} \|x_i - m_j\|_2^2 = g(T, M)$$

Then from above, we have

$$\sum_{j=1}^{k} \sum_{x_i \in T_j} \|x_i - m_j\|_2^2 \geq \sum_{j=1}^{k} \sum_{x_i \in T_j} \|x_i - \mu(T_j)\|_2^2 = f(T)$$

so it follows that $f(T) < f(S)$, which is a contradiction since we assumed that $f(S)$ was minimal. Thus, we have the final relation $\gamma_k \leq \varphi_k$. □

(d) Suppose you wanted to select k (the number of clusters) to minimize the objective function. Should you work with objective 1 or 2? If 2, how does your choice of τ effect your choice of k?

Solution. WLOG, all data points are unique. If we use objective 1, to achieve the minimum objective function we must have $k = N$, which will give an objective function value of 0.

If we use objective 2, assuming τ is a positive constant, then taking $k = N$ again we achieve the minimum objective function value of $N\tau$. WLOG the distance between x_1 and x_2 is minimal in the data set, so that $\|x_1 - x_2\|_2^2 \leq \|x_i - x_j\|_2^2$ for all $i \neq j$. Take $\tau \geq \frac{1}{4}\|x_1 - x_2\|_2^2$. Then if $S = \{\{x_1, x_2\}, x_3, \cdots, x_N\}$, we have

$$\begin{aligned}
g(S,\tau) &= \sum_{j=1}^{N-1} \sum_{x_i \in S_j} \max\left\{\|x_i - \mu_j\|_2^2, \tau\right\} \\
&= \max\left\{\left\|x_1 - \frac{x_1+x_2}{2}\right\|_2^2, \tau\right\} + \max\left\{\left\|x_2 - \frac{x_1+x_2}{2}\right\|_2^2, \tau\right\} + (N-2)\tau \\
&= \max\left\{\left\|\frac{x_1-x_2}{2}\right\|_2^2, \tau\right\} + \max\left\{\left\|\frac{x_2-x_1}{2}\right\|_2^2, \tau\right\} + (N-2)\tau \\
&= 2\max\left\{\frac{1}{4}\|x_1 - x_2\|_2^2, \tau\right\} + (N-2)\tau = 2\tau + (N-2)\tau = N\tau
\end{aligned}$$

Thus, the minimum objective function value can be achieved using $k-1$ clusters. Generalizing, if we increase τ, we can achieve the minimum objective function value using smaller k. □