Machine Learning HW#3

Ella Pavlechko
Due 10/25/19

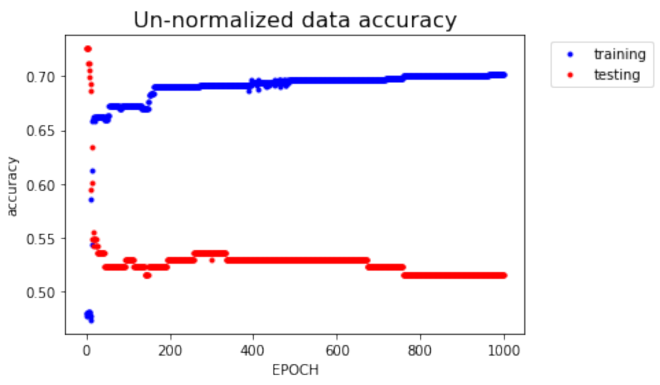
1 Use the same credit card application data file provided in the MATLAB file, SampleCredit.mat, from the Perceptron homework to train a neural network model. Use the first 500 data for the training set and the rest for the testing set. For the artificial neural network (ANN) model, you can use 3 layers: layer 1 (30 neurons and linear transfer function for the activation function); layer 2 (30 neurons and tanh function for the activation function); layer 3, the output layer (2 neurons and softmax for the activation function). For the output, report the test (prediction) accuracy for the following two cases:

- Data is used unchanged.
- Data is normalized by the maximum value of each feature.

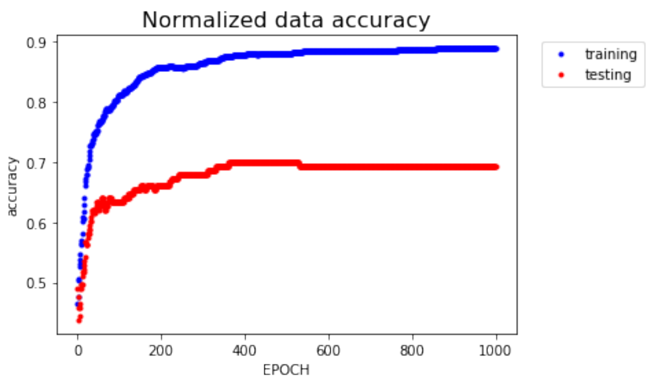
How is the test accuracy from the ANN compared with the test accuracy from the Perceptron model from the Perceptron homework?

We train the 3 layer Neural Network by:

1. Initializing the weights and biases to be randomly selected from a normal distribution with mean zero and standard deviation 1
2. Reshuffle the training data and pass through the 3 layers, then predict classes for each data point
3. Backprop. using the Gradient Descent Optimizer, which heads in the direction of largest gradient to minimize cross-entropy loss
4. Update the weights and biases
5. Pass the testing data through, and make predictions, then compute the accuracy
6. Repeat steps 2-5 for 1000 iterations.



(a)



(b)

Figure 1: Accuracy of the data as we train the Artificial Neural Network

Figure 1 plots the accuracies of the network as it trains. At the final iteration the un-normalized data has accuracy of :

$$(\%\ accuracy)_{train} = 70.6\%$$
$$(\%\ accuracy)_{test} = 51.6\%$$

The normalized data has accuracy of:

$$(\%\ accuracy)_{train} = 88.4\%$$
$$(\%\ accuracy)_{test} = 67.1\%$$

We can also observe from Figure 1b that the updating of accuracies is a much smoother process with the normalized data, and that it has a better percent accuracy than the un-normalized case.

Recall from Homework 2, with the Perceptron model, we had accuracies of un-normalized data

$$(\% \ accuracy)_{train} = 64.8\%$$
$$(\% \ accuracy)_{test} = 68.0\%$$

and normalized,

$$(\% \ accuracy)_{train} = 77.0\%$$
$$(\% \ accuracy)_{test} = 81.7\%$$

So we observe that in both cases, the training accuracies have increased by using a Neural Network. Although, now the testing accuracies have decreased, and are lower than the training accuracy. This is typical, but because they are close, it's a good sign that we're not overfitting the data. In other words, we can apply our network to new data the network hasn't seen before, and it will perform fairly well.

2 Consider the following training data with two categories (labels):

$$C_1 : (1,3)^T \quad (2,3)^T \quad (2,4)^T$$
$$C_{-1} : (3,1)^T \quad (3,2)^T \quad (4,2)^T$$

That is, there are six training data points, each data point has two features and a corresponding label.

(a) Plot these six training data points and observe that they are separable.

(i) Construct by inspection the weight vector w and the bias b for the optimal hyperplane from the support vector machine (SVM) model. What are the support vectors?

(ii) Find the SVM model by solving the corresponding quadratic programming problem in the dual Lagrangian formulation. Show that the SVM solution is the same as those obtained in part (i).

(b) Find the separating hyperplane by using the Linear Discriminant Analysis (LDA) model.

For this training data set, is the LDA hyperplane the same as the SVM hyperplane?

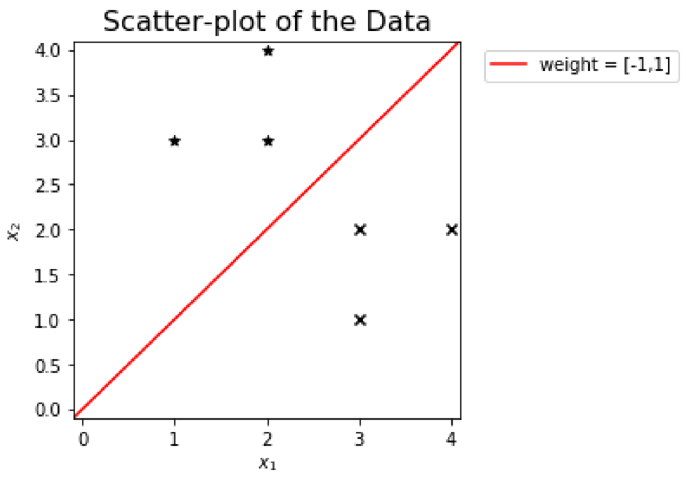


Figure 2: The six data points and the line $w^T \ x = 0$

(a) By visual inspection, the data reflects about the line $y = x$, so we may suspect this as our optimal hyperplane. We can confirm this by showing the vector $[-1,1]^T$ is the difference between $[2,3]^T$ and $[3,2]^T$, the nearest two points from opposing classes, and whose length defines our margin. Since the optimal hyperplane must be orthogonal to $[-1,1]^T$, and from linear algebra a line is expressed in terms of it's normal vector, we get our optimal hyperplane is

$$[-1 \, 1] \begin{bmatrix} x \\ y \end{bmatrix} = 0$$

defined by weight, $w = \begin{bmatrix}-1\,1\end{bmatrix}$ and bias, $b = 0$. Note, the above formula is the matrix-vector representation of $y = x$.
For the rest of the problem we will notate $[x, y]$ as $[x_1, x_2]$.
In a more rigorous setting, if we wanted to show we should set our margin to be $m = \frac{2}{\|[-1,1]\|}$ we would need to show it's the solution to the optimization problem:

$$\begin{aligned} \underset{w}{\text{maximize}} \quad & \frac{2}{\|w\|} \\ \text{subject to} \quad & y_i(w^T x(i) + b) \geq 1 \end{aligned}$$

which is equivalent to solving

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \|w\| \\ \text{subject to} \quad & y_i(w^T x(i) + b) \geq 1 \end{aligned}$$

and can be simplified to the Primal

$$\begin{aligned} \underset{w,b}{\text{minimize}} \quad & \frac{1}{2} w^T w \\ \text{subject to} \quad & -y_i(w^T x(i) + b) + 1 \leq 0 \end{aligned} \tag{1}$$

using the KKT conditions, we set Q to be the outer product $Q = [y_1 x_1, ..., y_N x_N]^T [y_1 x_1, ..., y_N x_N]$ so the Dual becomes

$$\begin{aligned} \underset{\lambda}{\text{maximize}} \quad & \lambda^T 1 - \frac{1}{2}\lambda^T Q \lambda \\ \text{subject to} \quad & \lambda_i \geq 0 \\ & \lambda^T y = 0 \end{aligned} \tag{2}$$

Using the quadratic programming solver *quadprog* in Python (and recalling that $max\ f(x) = -min\ -f(x)$ to encode it into the solver), we find the optimal solution to the Dual (Formulation 2)

$$\lambda^* = [0, 1, 0, 0, 1, 0]$$

and can now find w^* and b^*, the optimal solutions to the Primal (Formulation 1)

$$w^* = \sum_{i=1}^{6} \lambda_i^* y_i x(i) = [-1, 1]$$

$$b_i^* = \frac{1 - y_i(w^* \cdot x(i))}{y_i}$$

$$b^* = [-1, 0, -1, 1, 0, 1]$$

Since we only consider the b_i^* with $\lambda_i > 0$, corresponding to active constraints in our dual, then our resulting optimal is

$$b^* = 0$$

Therefore, the optimal hyperplane is defined by vectors $w = [-1, 1]$ and $b = 0$, the same result as our visual inspection from part i).

(b) For the LDA we're looking for the plane $w_0 + w^T x = 0$ such that the projection of the data onto the hyperplane $\hat{w}$ has maximum distribution between means, while minimizing the variance. We'll start by finding the distribution of the data in each one of the classes.

$$n_1 = 3 \qquad\qquad n_{-1} = 3$$

$$\mu_1 = \left[\frac{5}{3}, \frac{10}{3}\right]^T \qquad\qquad \mu_{-1} = \left[\frac{10}{3}, \frac{5}{3}\right]^T$$

$$\Sigma_1 = \begin{bmatrix} \frac{2}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{2}{6} \end{bmatrix} \qquad\qquad \Sigma_{-1} = \begin{bmatrix} \frac{2}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{2}{6} \end{bmatrix}$$

Using Fisher's optimility criteria, to maximize the differences in the mean, but minimize the variance, we would like to find w^* such that

$$w^* = \underset{w}{\text{argmax}} \frac{(w^T\mu_1 - w^T\mu_{-1})^2}{w^T\Sigma_1 w + w^T\Sigma_{-1}w}$$
$$= \underset{w}{\text{argmax}} \frac{(\left[-\frac{5}{3}, \frac{5}{3}\right] w)^2}{w^T \begin{bmatrix}\frac{2}{3} & \frac{1}{3}\\ \frac{1}{3} & \frac{2}{3}\end{bmatrix} w}$$

which occurs when

$$w^* = \frac{1}{c}\begin{bmatrix}\frac{2}{3} & \frac{1}{3}\\ \frac{1}{3} & \frac{2}{3}\end{bmatrix}^{-1}\begin{bmatrix}-\frac{5}{3}\\ \frac{5}{3}\end{bmatrix}$$

where c is an arbitrary constant,

$$= \begin{bmatrix}\frac{2}{3} & -\frac{1}{3}\\ -\frac{1}{3} & \frac{2}{3}\end{bmatrix}\begin{bmatrix}-\frac{5}{3}\\ \frac{5}{3}\end{bmatrix}$$
$$= \begin{bmatrix}-\frac{5}{3}\\ \frac{5}{3}\end{bmatrix}$$

Now if $w_0^* = \frac{1}{6}\sum_{i=1}^{6} y_i - w^* x(i)$, then $w_0^* = 0$. Observe that the hyperplane $w_0^* + w^{*T}x = 0$ is now the exact same hyperplane as the SVM and visual observation techniques. The only difference here is that our weight vector is $\left[-\frac{5}{3}, \frac{5}{3}\right]^T$, but this is just a constant multiple of $[-1, 1]^T$, the weight predicted by the other techniques.