Homework 5

Sara Beery
CS 156A - Learning Systems

November 5, 2018

1 Linear Regression Error

Consider a noisy target $y = w^T x + \epsilon$, where $x \in \mathbb{R}^d$ (with the added coordinate $x_0 = 1$), $y \in \mathbb{R}$, w is an unknown vector, and ϵ is a noise term with zero mean and σ^2 variance. Assume ϵ is independent of x and of all other ϵs. If linear regression is carried out using a training data set $D = \{(x_1, y_1), ..., (x_N, y_N)\}$, and outputs the parameter vector w_{lin}, it can be shown that the expected in-sample error E_{in} with respect to D is given by:

$$\mathbb{E}_D[E_{in}(w_{lin})] = \sigma^2\Big(1 + \frac{d+1}{N}\Big)$$

For $\sigma = 0.1$ and $d = 8$, we can find the smallest N that will result in expected $E_{in} > 0.008$ by setting

$$\sigma^2\Big(1 + \frac{d+1}{N}\Big) > 0.008$$

and solving for N. We get

$$N > \frac{d+1}{1 - \frac{\epsilon}{\sigma^2}} = \frac{8+1}{1 - \frac{0.008}{0.1^2}} = 45$$

and the closest answer is

1. [**c**] 100

2 Nonlinear Transforms

In linear classification, consider the feature transform $\Phi : \mathbb{R}^2 \to \mathbb{R}^2$ (plus the added zeroth coordinate) given by:

$$\Phi(1, x_1, x_2) = (1, x_1^2, x_2^2)$$

Which of the following sets of constraints on the weights in the Z space could correspond to the hyperbolic decision boundary in X depicted in the figure?

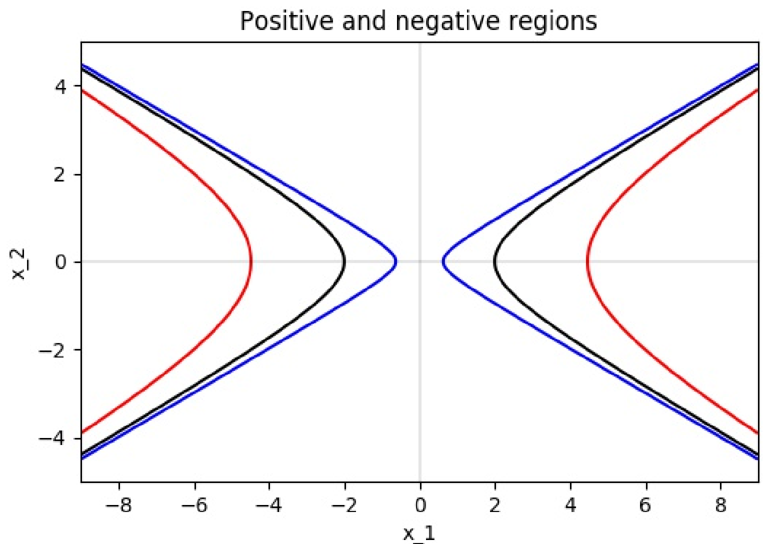

Figure 1: The decision boundary is in black, the negative region is in red, and the positive region is in blue.

First note that the equation given weights in the Z space will be

$$y = w^T z = w_0 + w_1 x_1^2 + w_2 x_2^2$$

in the X space. Also note that the equation of a vertically oriented hyperbola centered at the origin is

$$(\frac{1}{a^2})x_1^2 + \frac{-1}{b^2}x_2^2 - 1 = 0$$

which would represent the decision boundary where $y = 0$. Further, in Fig. 1 you can see that the vertically oriented hyperbola has the desired positive region. Therefore we can see that we should select $w_0 < 0$, $w_1 > 0$, and $w_2 < 0$. So:

2. [**e**] $w_1 > 0$, $w_2 < 0$

Now, consider the 4th order polynomial transform from the input space $\mathbb{R}^2$:

$$\Phi_4 x \rightarrow (1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1, x_2^2, x_2^3, x_1^4, x_1^3x_2, x_1^2x_2^2, x_1x_2^3, x_2^4)$$

Recall that after a nonlinear transformation, $d_{vc} \leq \tilde{d} + 1$ where $\tilde{d}$ is the dimension of the transformed Z space. In this case $\tilde{d} = 14$, so $d_{vc} \leq 15$.

3. [**c**] 15

3 Gradient Descent

Consider the nonlinear error surface $E(u, v) = (ue^v - 2ve^u)^2$. We start at the point $(u, v) = (1, 1)$ and minimize this error using gradient descent in the uv space. Use $\eta = 0.1$ (learning rate, not step size).

Taking the partial derivative with respect to u:

$$\frac{\partial}{\partial u}E(u,v) = 2(ue^v - 2ve^{-u})(e^v + 2ve^{-u})$$

So we have:

4. [**e**] $2(e^v + 2ve^{-u})(ue^v - 2ve^{-u})$

Using gradient descent, our direction of error will be

$$\frac{\partial}{\partial u}E(u,v) = 2(ue^v - 2ve^{-u})(e^v + 2ve^{-u})$$
$$\frac{\partial}{\partial v}E(u,v) = 2(ue^v - 2ve^{-u})(ue^v - 2e^{-u})$$

When following the gradient descent algorithm, we converge to error $< 10^{-14}$ in 10 steps

5. [**d**] 10

The associate values of (u, v) are $(0.04473629, 0.02395871)$. The closest values to (u, v) in euclidean space from the given choices are:

6. [**e**] (.045,.024)

Coordinate descent (alternating between descending in the u coordinate direction and the v coordinate direction) gives us an error of 0.13981379199615315 after 15 iterations (30 steps). This is closest to:

7. [**a**] 10^{-1}

4 Logistic Regression

After running Logistic Regression to learn a random linear target function 100 times (see an example of a single iteration in Fig. 2), we find that on average $E_{out} = 0.0234$, which is closest to:

8. [**a**] 0.025

We also find that it takes on average 338.93 epochs to converge, which is closest to:

9. [**a**] 350

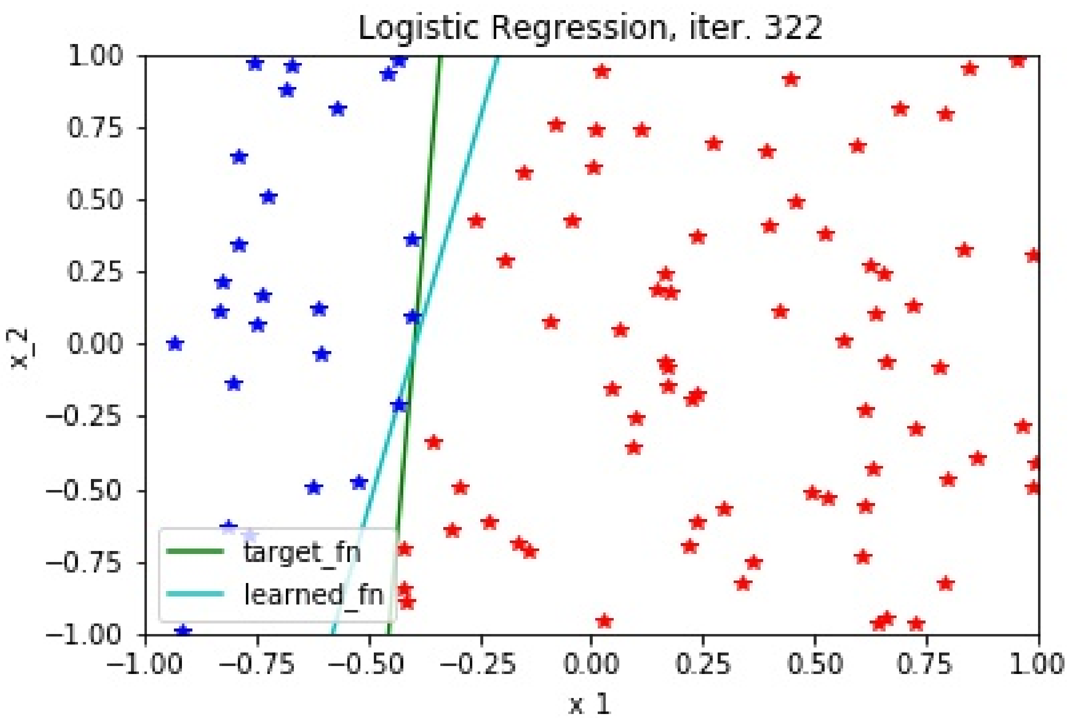

Figure 2: The learned function colors the points appropriately, but does not perfectly match the target function.

5 PLA as SGD

The Perceptron Learning Algorithm uses the following target function:

$$f(x) = \begin{cases} +1 & w^T x > 0 \\ 0 & otherwise, \end{cases}$$

so it can be implemented as SGD using:

10. [**e**] $e_n(w) = -min(0, y_n w^T x_n)$