Learning From Data
Caltech
http://work.caltech.edu/telecourse
Spring 2013

Online Homework # 5

*All questions have multiple-choice answers (***[a], [b], [c], ...***). You can collaborate with others, but do not discuss the selected or excluded choices in the answers. You can consult books and notes, but not homework solutions. Definitions and notation follow the lectures. Your solutions should be based on your own work, and should be entered online by logging into your account at the course web site.*

Note about the homeworks

- The goal of the homeworks is to facilitate a deeper understanding of the course material. The questions are not designed to be puzzles with catchy answers. They are meant to make you roll your sleeves, face uncertainties, and approach the problem from different angles.

- The problems range from easy to hard, and from theoretical to practical. Some problems require running a full experiment to arrive at the answer.

- The answer may not be obvious or numerically close to one of the choices, but one (and only one) choice will be correct if you follow the exact instructions in each problem. You are encouraged to explore the problem further by experimenting with variations of these instructions, for the learning benefit.

- You are also encouraged to take part in the forum

  http://book.caltech.edu/bookforum

  where there are many threads about each homework. We hope that you will contribute to the discussion as well. Please follow the forum guidelines for posting answers (see the "BEFORE posting answers" announcement at the top there).

Linear Regression Error

Consider a noisy target $y = \mathbf{w}^{*T}\mathbf{x} + \epsilon$, where $\mathbf{x} \in \mathbb{R}^d$ (with the added coordinate $x_0 = 1$), $y \in \mathbb{R}$, $\mathbf{w}^*$ is an unknown vector, and ϵ is a noise term with zero mean and σ^2 variance. Assume ϵ is independent of $\mathbf{x}$ and of all other ϵ's. If linear regression is carried out using a training data set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, and outputs the parameter vector $\mathbf{w}_{\text{lin}}$, it can be shown that the expected in-sample error E_{in} with respect to $\mathcal{D}$ is given by:

$$\mathbb{E}_{\mathcal{D}}[E_{\text{in}}(\mathbf{w}_{\text{lin}})] = \sigma^2 \left(1 - \frac{d+1}{N}\right)$$

1. For $\sigma = 0.1$ and $d = 8$, which among the following choices is the smallest number of examples N that will result in an expected E_{in} greater than 0.008?

[**a**] 10

[**b**] 25

[**c**] 100

[**d**] 500

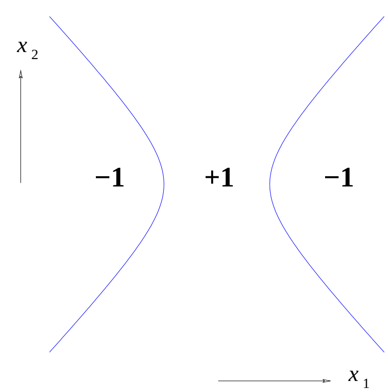
[**e**] 1000

Nonlinear Transforms

Consider the feature transform $\Phi : \mathbb{R}^2 \to \mathbb{R}^2$ (plus the added zeroth coordinate) given by:

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

2. Which of the following sets of constraints on the weights in the $\mathcal{Z}$ space could correspond to the hyperbolic decision boundary in $\mathcal{X}$ depicted in the figure? You may assume that $\tilde{w}_0$ can be selected to achieve the desired boundary.



[a] $\tilde{w}_1 = 0, \tilde{w}_2 > 0$

[b] $\tilde{w}_1 > 0, \tilde{w}_2 = 0$

[c] $\tilde{w}_1 > 0, \tilde{w}_2 > 0$

[d] $\tilde{w}_1 < 0, \tilde{w}_2 > 0$

[e] $\tilde{w}_1 > 0, \tilde{w}_2 < 0$

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

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

3. What is the smallest value among the following choices that is $\geq$ the VC dimension of a linear model in the transformed space?

[a] 3

[b] 5

[c] 15

[d] 20

[e] 21

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).

4. What is the partial derivative of $E(u, v)$ with respect to u: $\frac{\partial E}{\partial u}$?

[a] $(ue^v - 2ve^{-u})^2$

[b] $2(ue^v - 2ve^{-u})$

[c] $2(e^v + 2ve^{-u})$

[d] $2(e^v - 2ve^{-u})(ue^v - 2ve^{-u})$

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

5. How many iterations (among the given choices) does it take for the error $E(u, v)$ to fall below 10^{-14} for the first time? In your programs, make sure to use double precision to get the needed accuracy.

[a] 1

[b] 3

[c] 5

[d] 10

[e] 17

6. After running enough iterations such that the error has just dropped below 10^{-14}, what are the closest values (in Euclidean distance) to the final (u, v)?

[a] $(1.000, 1.000)$

[b] $(0.713, 0.045)$

[c] $(0.016, 0.112)$

[d] $(-0.083, 0.029)$

[e] $(0.045, 0.024)$

7. Now, we will compare the performance of "coordinate descent." In each iteration, we have two steps along the 2 coordinates. Step 1 is to move along the u coordinate only to reduce the error (assume first-order approximation holds like in gradient descent), and step 2 is to reevaluate and move along the v coordinate only to reduce the error (again, assume first-order approximation holds). Continue to use a learning rate of $\eta = 0.1$ as we did in gradient descent. What will the error $E(u, v)$ be closest to after 15 full iterations (30 steps)?

[a] 10^{-1}

[b] 10^{-7}

[c] 10^{-14}

[d] 10^{-17}

[e] 10^{-20}

Logistic Regression

In this problem you will create your own target function f (probability in this case) and data set $\mathcal{D}$ to see how Logistic Regression works. For simplicity, we will take f to be a 0/1 probability, so y is a deterministic function of $\mathbf{x}$.
Take $d = 2$ so you can visualize the problem, and choose a random line in the plane as the boundary between $f(\mathbf{x}) = 1$ (where y has to be $+1$) and $f(\mathbf{x}) = 0$ (where y has to be -1). Do this by taking two random uniformly distributed points on $[-1, 1] \times [-1, 1]$ and taking the line passing through them as the boundary between $y = \pm 1$. Take $\mathcal{X} = [-1, 1] \times [-1, 1]$ with uniform probability of picking each $\mathbf{x} \in \mathcal{X}$,

and take $N = 100$. Choose the inputs $\mathbf{x}_n$ of the data set as random points in $\mathcal{X}$, and evaluate the output y_n for each $\mathbf{x}_n$.
Run Logistic Regression with Stochastic Gradient Descent to find g and estimate E_{out} (the **cross entropy** error) by generating a sufficiently large separate set of points to evaluating the error. Repeat the experiment for 100 runs with different targets and take the average. Initialize the weight vector of Logistic Regression to all zeros in each run. Stop the algorithm when $\|\mathbf{w}^{(t-1)} - \mathbf{w}^{(t)}\| < 0.01$, where $\mathbf{w}^{(t)}$ denotes the weight vector at the end of epoch t. An epoch is a full pass through the N data points (use a random permutation of $1, 2, \cdots, N$ to present the data points to the algorithm within each epoch, and use different permutations for different epochs). Use a learning rate of 0.01.

8. Which of the following is closest to E_{out} for $N = 100$?

[**a**] 0.025
[**b**] 0.050
[**c**] 0.075
[**d**] 0.100
[**e**] 0.125

9. How many epochs does it take on average for Logistic Regression to converge for $N = 100$ using the above initialization and termination rules, and specified learning rate? Pick the value that is closest to your results.

[**a**] 350
[**b**] 550
[**c**] 750
[**d**] 950
[**e**] 1750

10. The Perceptron Learning Algorithm can be implemented as SGD using which of the following error functions $\mathrm{e}_n(\mathbf{w})$:

[**a**] $\mathrm{e}_n(\mathbf{w}) = e^{-y_n \mathbf{w}^\intercal \mathbf{x}_n}$
[**b**] $\mathrm{e}_n(\mathbf{w}) = -y_n \mathbf{w}^\intercal \mathbf{x}_n$
[**c**] $\mathrm{e}_n(\mathbf{w}) = (y_n - \mathbf{w}^\intercal \mathbf{x}_n)^2$
[**d**] $\mathrm{e}_n(\mathbf{w}) = \ln(1 + e^{-y_n \mathbf{w}^\intercal \mathbf{x}_n})$
[**e**] $\mathrm{e}_n(\mathbf{w}) = -\min(0, y_n \mathbf{w}^\intercal \mathbf{x}_n)$