EECS545: Machine Learning, Winter 2020
Homework #1

Due 11:55pm on 1/28 (Tue).

Reminder: While you are encouraged to think about problems in small groups, all written solutions must be independently generated. Please type or hand-write solutions legibly. While these questions require thought, please be as concise and clear in your answer as possible. Please address all questions to `http://piazza.com/class#winter2020/eecs545` with a reference to the specific question in the subject line (E.g. RE: Homework 1, Q2(c)). For any solutions that require programming, please use Numpy to implement the machine learning algorithms from scratch. Please submit code as well as the written solution with any figures you are required to plot.

1 [21 points] Logistic regression

(a) **[8 points]** Consider the log-likelihood function for logistic regression:

$$\ell(\mathbf{w}) = \sum_{i=1}^{N} y^{(i)} \log h(\mathbf{x}^{(i)}) + (1 - y^{(i)}) \log(1 - h(\mathbf{x}^{(i)})),$$

where $h(\mathbf{x}) = sigmoid(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x})}$.

Find the Hessian H of this function, and show that is negative semi-definite and thus ℓ is concave and has no local maxima other than the global one. That is, show that

$$\mathbf{z}^T H \mathbf{z} \leq 0$$

for any vector $\mathbf{z}$. [Hint: You might want to start by showing the fact that $\sum_i \sum_j z_i x_i x_j z_j = (\mathbf{x}^T\mathbf{z})^2 \geq 0$.]

(b) **[8 points]** Using the H you calculated in part (a), write down the update rule implied by Newton's method for optimizing $\ell(\mathbf{w})$. Now use this rule (and not a library function) to implement Newton's method and apply it to binary classification problem specified in files `q1x.npy` and `q1y.npy`. The two columns of `q1x.npy` represent the inputs ($x^{(i)}$) and `q1y.npy` represents the outputs ($y^{(i)} \in \{0, 1\}$), with one training example per row. Initialize Newtons method with $\mathbf{w} = \mathbf{0}$ (the vector of all zeros). What are the coefficients $\mathbf{w}$, including the intercept term, resulting from your fit? [Hint: You might refer to the code in `q1_hint.py` to start programming.]

(c) **[5 points]** Plot the training data (your axes should be x_1 and x_2, corresponding to the two coordinates of the inputs, and you should use a different symbol for each point plotted to indicate whether that example had label 1 or 0). Also plot on the same figure the decision boundary fit by logistic regression. (I.e., this should be a straight line showing the boundary separating the region where $h(\mathbf{x}) > 0.5$ from where $h(\mathbf{x}) \leq 0.5$.)

2 [32 points] Linear regression on a polynomial

The files `q2xTrain.npy`, `q2xTest.npy`, `q2yTrain.npy` and `q2yTest.npy` specify a linear regression problem for a polynomial. `q2xTrain.npy` represent the inputs ($\mathbf{x}^{(i)} \in \mathbb{R}$) and `q2yTrain.npy` represents the outputs ($y^{(i)} \in \mathbb{R}$) of the training set, with one training example per row.

(a) **[12 points]** For each of the following optimization methods, find the coefficients (slope and intercept) that minimize the error for a first degree polynomial.

- Batch gradient descent
- Stochastic gradient descent
- Newton's method

i. **[9 points]** Give the coefficients generated by each of the three optimization methods.

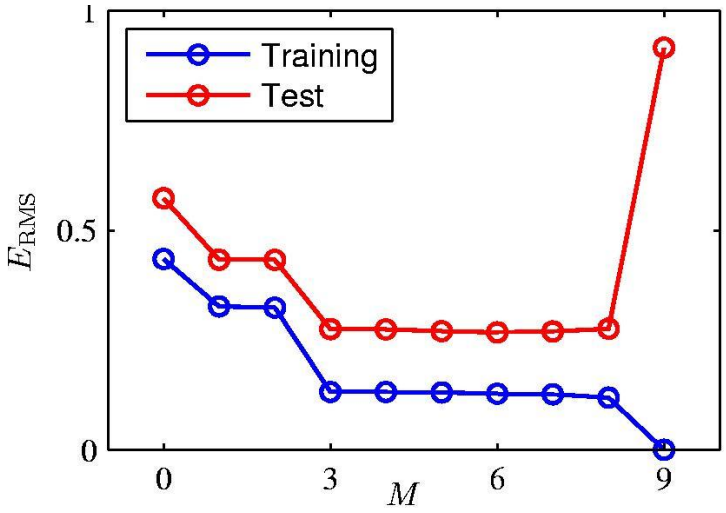
ii. **[3 points]** How did the methods compare in terms of rate of convergence? [Hint: The training process can be viewed as convergent when the mean squared error on the training dataset is low enough (e.g ≤ 0.2) and stable.]

(b) **[10 points]** Next, you will investigate the problem of over-fitting. Recall the figure from lecture that explored over-fitting as a function of the degree of the polynomial M, where the Root-Mean-Square (RMS) Error is define as

$$E_{RMS} = \sqrt{2E(\mathbf{w}^*)/N},$$

where

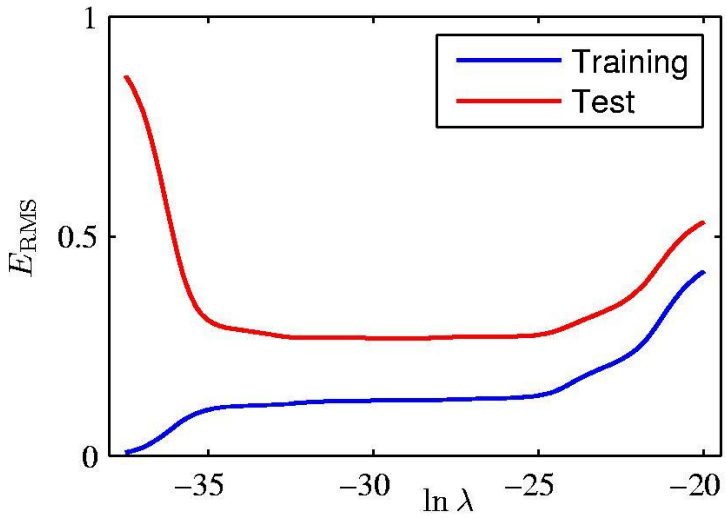
$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(\sum_{j=0}^{M} w_j \phi_j(\mathbf{x}^{(i)}) - y^{(i)})^2 = \frac{1}{2}\sum_{i=1}^{N}(\mathbf{w}^T\phi(\mathbf{x}^{(i)}) - y^{(i)})^2 :$$



i. **[7 points]** Using Newton's method, find the coefficients that minimize the error for a M-degree polynomial (for $M = 0 \ldots 9$) for the training data specified in `q2xTrain.npy` and `q2yTrain.npy`. Now use these parameters to regenerate the above chart, using `q2xTest.npy` and `q2yTest.npy` as the test data.

ii. **[3 points]** Which degree polynomial would you say best fits the data? Was there evidence of under/over-fitting the data? Use your generated charts to defend your answer.

(c) **[10 points]** Finally, you will explore the role of regularization. Recall the image from lecture that explored the affect of the regularization factor λ:



i. **[7 points]** Using Newton's method, find the coefficients that minimize the error for a ninth degree polynomial ($M = 9$) given regularization factor λ (for $\lambda = \{0, 10^{-6}, 10^{-5}, \ldots, 10^{-1}, 10^{0}(1)\}$) for the training data specified in `q2xTrain.npy` and `q2yTrain.npy`. Now use these parameters to plot the E_{RMS} of both the training data and test data as a function of λ (regenerate the above chart, using `q2xTest.npy` and `q2yTest.npy` as the test data). Specifically, use the following regularized objective function:

$$\frac{1}{2}\sum_{i=1}^{N}(\mathbf{w}^T\phi(\mathbf{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2.$$

for optimizing the parameters $\mathbf{w}$, but please use the original (unregularized) E_{RMS} for plotting.

ii. **[3 points]** Which λ value seemed to work the best?

3 [24 points] Locally weighted linear regression

Consider a linear regression problem in which we want to weight different training examples differently. Specifically, suppose we want to minimize

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N} r^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})^2.$$

In class, we worked out what happens for the case where all the weights (the $r^{(i)}$'s) are the same. In this problem, we will generalize some of those ideas to the weighted setting, and also implement the locally weighted linear regression algorithm. (Note that the weight $r^{(i)}$ can be different for each of the training example.)

(a) **[2 points]** Show that $E_D(w)$ can also be written

$$E_D(\mathbf{w}) = (X\mathbf{w} - \mathbf{y})^T R(X\mathbf{w} - \mathbf{y})$$

for an appropriate diagonal matrix R, and where X is a matrix whose i-th row is $x^{(i)}$ and $\mathbf{y}$ is the vector whose i-th entry is $y^{(i)}$. State clearly what R is.

(b) **[6 points]** If all the $r^{(i)}$'s equal 1, then we saw in class that the normal equation is

$$X^T X\mathbf{w} = X^T\mathbf{y},$$

and that the value of $\mathbf{w}^*$ that minimizes $E_D(\mathbf{w})$ is given by $(X^T X)^{-1}X^T\mathbf{y}$. By finding the derivative $\nabla_{\mathbf{w}} E_D(\mathbf{w})$ and setting that to zero, generalize the normal equation to this weighted setting, and give the new value of $\mathbf{w}^*$ that minimizes $E_D(w)$ in closed form as a function of X, R and $\mathbf{y}$.

(c) **[5 points]** Suppose we have a training set $\{(\mathbf{x}^{(i)}, y^{(i)}); i = 1\ldots, N\}$ of N independent examples, but in which the $y^{(i)}$'s were observed with differing variances. Specifically, suppose that

$$p(y^{(i)}|\mathbf{x}^{(i)}; \mathbf{w}) = \frac{1}{\sqrt{2\pi}\sigma^{(i)}} \exp(-\frac{(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2}{2(\sigma^{(i)})^2})$$

I.e., $y^{(i)}$ has mean $\mathbf{w}^T\mathbf{x}^{(i)}$ and variance $(\sigma^{(i)})^2$ (where the $\sigma^{(i)}$'s are fixed, known, constants). Show that finding the maximum likelihood estimate of $\mathbf{w}$ reduces to solving a weighted linear regression problem. State clearly what the $r^{(i)}$'s are in terms of the $\sigma^{(i)}$'s.

(d) **[11 points]** The following items will use the files `q3x.npy` which contains the inputs ($\mathbf{x}^{(i)}$) and `q3y.npy` which contains the outputs ($y^{(i)}$) for a linear regression problem, with one training example per row.

i. **[2 points]** Implement (unweighted) linear regression ($y = \mathbf{w}^T\mathbf{x}$) on this dataset (using the normal equations), and plot on the same figure the data and the straight line resulting from your fit. (Remember to include the intercept term.)

ii. **[6 points]** Implement locally weighted linear regression on this dataset (using the weighted normal equations you derived in part (b)), and plot on the same figure the data and the curve resulting from your fit. When evaluating $h(\cdot)$ at a query point x (which is real-valued in this problem), use weights

$$r^{(i)} = \exp(-\frac{(x - x^{(i)})^2}{2\tau^2})$$

with a bandwidth parameter $\tau = 0.8$. (Again, remember to include the intercept term.)

iii. **[3 points]** Repeat (ii) four times with $\tau = 0.1$, 0.3, 2 and 10. Comment **briefly** on what happens to the fit when τ is too small or too large.

4 Derivation and Proof

(a) **[8 points]** Consider the linear regression problem for 1D data, where we would like to learn a function $h(x) = w_1 x + w_0$ with parameters w_0 and w_1 to minimize the mean squared error $L = \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - h(x^{(i)}))^2$ for N pairs of data samples $(x^{(i)}, y^{(i)})$. Derive the solution for w_0 and w_1 for this 1D case of linear regression. Show the derivation to get the solution $w_0 = \bar{Y} - w_1\bar{X}$ and $w_1 = \frac{\frac{1}{N}\sum_{i=1}^{N} x^{(i)}y^{(i)} - \bar{Y}\bar{X}}{\frac{1}{N}\sum_{i=1}^{N} x^{(i)2} - \bar{X}^2}$

(b) **[15 points]** Consider the definition and property of positive (semi-)definite matrix. Let $\mathbf{A}$ be a real, symmetric $d \times d$ matrix. $\mathbf{A}$ is positive semi-definite (PSD) if, for all $\mathbf{z} \in \mathcal{R}^d$, $\mathbf{z}^T\mathbf{A}\mathbf{z} \geq 0$. $\mathbf{A}$ is positive definite (PD) if, for all $\mathbf{z} \neq \mathbf{0}$, $\mathbf{z}^T\mathbf{A}\mathbf{z} > 0$. We write $\mathbf{A} \succeq 0$ when $\mathbf{A}$ is PSD, and $\mathbf{A} \succ 0$ when $\mathbf{A}$ is PD. The spectral theorem says that every real symmetric matrix $\mathbf{A}$ can be expressed via the spectral decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ where $\mathbf{U}$ is a $d \times d$ matrix such that $\mathbf{U}\mathbf{U}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I}$ and $\mathbf{\Lambda} = diag(\lambda_1, \lambda_2, \cdots, \lambda_d)$. Multiplying on the right by $\mathbf{U}$ we see that $\mathbf{A}\mathbf{U} = \mathbf{U}\mathbf{\Lambda}$. If we let u_i denote the i-th column of $\mathbf{U}$, we have $\mathbf{A}u_i = \lambda_i u_i$ for each i. Then λ_i are eigenvalues of $\mathbf{A}$, and the corresponding columns are eigenvectors associated to λ_i. The eigenvalues constitute the "spectrum" of $\mathbf{A}$, and the spectral decomposition is also called the eigenvalue decomposition of $\mathbf{A}$.

i. [6 points] Prove $\mathbf{A}$ is PD iff $\lambda_i > 0$ for each i.

ii. [9 points] Consider the linear regression problem where Φ and $\mathbf{y}$ are as defined in class and the closed form solution is $(\Phi^T\Phi)^{-1}\Phi^T\mathbf{y}$. We can get the eigenvalues of symmetric matrix $\Phi^T\Phi$ using spectral decomposition. We apply ridge regression, and the symmetric matrix in solution is $\Phi^T\Phi + \beta\mathbf{I}$. Prove that the ridge regression has an effect of shifting all singular values by a constant β. For any $\beta > 0$, ridge regression makes the matrix $\Phi^T\Phi + \beta\mathbf{I}$ PD.

Credits

Some questions adopted/adapted from http://www.stanford.edu/class/cs229/ps/ps1.pdf and from Bishop PRML.