UMass CMPSCI 383 (AI) HW6: Chapters 18 & 20

Jason Rowley

Assigned: Nov 27 2017; Due: Dec 11 2017 @ 11:55 PM EST

Abstract

Submit a (.zip) file to both Moodle and Gradescope containing your latex (.tex) file and rendered pdf. All written HW responses should be done in latex (use sharelatex.com or overleaf.com). All code should be written in Python and should run on the edlab machines ([yourusername]@elnux[1,2,...].cs.umass.edu). You may NOT use `sklearn` for this assignment.

1 Regression (20 pts)

L2-regularized Linear Regression, also called "Ridge" Regression, adds the L2 norm of the weights to the loss function: $||w||^2 = \sum_d w_d^2$. In this problem, you will train a "Ridge" Regression model using gradient descent on a dataset of (X,y) pairs we have provided . We have provided you with template code for loading the dataset, training the model, and plotting the resulting fit (see `linearRegression.py`). The relevant loss and gradient for training the model are below.

$$L(y, f(x;w)) = \frac{1}{2}\sum_i (w^T x_i - y_i)^2 + \alpha ||w||^2$$

$$\nabla_w L(y, f(x;w)) = \sum_i (w^T x_i - y_i)x_i + 2\alpha w = (X^T X + 2\alpha I)w - X^T y$$

$$\nabla_w L(y, f(x;w)) = 0 \Rightarrow w = (X^T X + 2\alpha I)^{-1} X^T y$$

In the previous homework, you used the pseudoinverse to exactly learn the utility of a given policy in Policy Iteration. As a reminder, the pseudoinverse is defined as

$$A^+ = (A^T A)^{-1} A^T = \texttt{np.linalg.pinv(A)}$$

In this problem, you'll need to compute the L2-regularized pseudoinverse yourself.

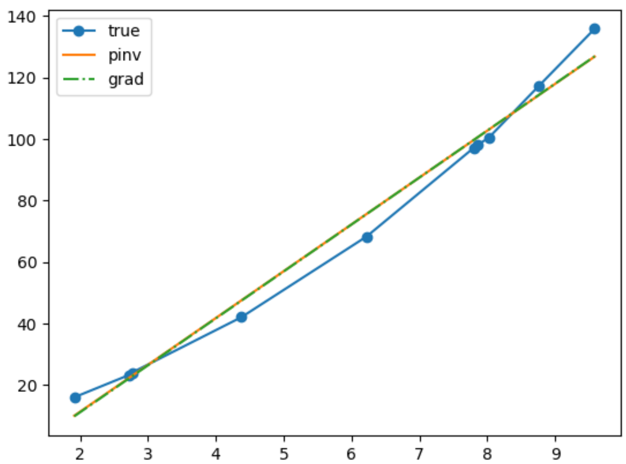
$$\tilde{A}_\alpha = A^T A + 2\alpha I = \texttt{np.dot(A.T,A) + 2*}\alpha\texttt{*np.eye(A.shape[1])}$$

$$A_\alpha^+ = \tilde{A}_\alpha^{-1} A^T = \texttt{np.linalg.inv(}\tilde{A}_\alpha\texttt{).dot(A.T)}$$

Initialize the weights to be zero and use $\alpha = 0.01$. Run gradient descent for 10,000 steps with a step size of 0.001.

1. Show that gradient descent returns the same weights as the exact solution using the pseudoinverse (to within 3 digits). Report the weights of both solutions below. (10 pts)
   Diff: 0.258202005369
   Exact: [15.23341778 -19.11470679]
   Gradient Descent: [15.26964405 -19.37035486]

2. Plot your fitted line with the data and display the plot below. Are you over-fitting, under-fitting, or fitting the data just right? Explain your thought process. (10 pts)



Based on the observation of the graph, I am under-fitting the data. The true line appears to be parabolic in nature, while the grad line is linear.

2 Classification (30 pts)

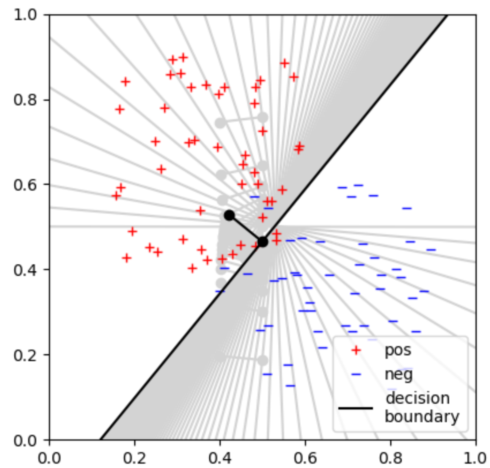
2.1 Logistic Regression

In this problem, you will learn to classify using Logistic Regression. Logistic Regression outputs the probability of x belonging to the positive class $y = T$, $P(y = T|x, w)$, where w is the weight vector. You can simply use $1 - P(y = T|x, w)$ to obtain the probability for the negative class. Note that you can add an L2-regularizer to Logistic Regression too, but we will not in this problem. We have provided you with template code for loading the dataset, training the model, and plotting the decision boundary (see `logisticRegression.py`). Logistic Regression is typically trained with a cross entropy loss. As before, we are providing you with the cross entropy loss for logistic regression along with its gradient (derived below).

$$
\begin{aligned}
P(y|x, w) &= f(x; w) = \sigma(z) \text{ where } z = w^T x, \quad \sigma(z) = \frac{1}{1 + e^{-z}}, \quad \frac{\partial \sigma}{\partial z} = \sigma(z)(1 - \sigma(z)) \\
l(y, \sigma(z)) &= -y \log(\sigma(z)) - (1 - y) \log(1 - \sigma(z)) \\
\frac{\partial l}{\partial w} &= \frac{\partial l}{\partial \sigma} \frac{\partial \sigma}{\partial z} \frac{\partial z}{\partial w} = \Big(\frac{-y}{\sigma(z)} + \frac{1 - y}{1 - \sigma(z)} \Big) \sigma(z)(1 - \sigma(z)) x = \Big(\sigma(z) - y \Big) x \\
L(\mathbf{y}, \sigma(X)) &= \frac{1}{N} \sum_i l(y_i, \sigma(z_i)) \text{ where } N = \text{ \# of samples} = \sum_i 1 \\
\nabla_w L &= \Big(\sigma(Z) - \mathbf{y} \Big)^T X \text{ where } Z = X^T w
\end{aligned}
$$

Initialize the weights to $[0, -1, 0.5]$. Run gradient descent for 10,000 steps with a step size of 0.0001.

1. Train a Logistic Regression model with gradient descent on the entire data set (X,y) and plot the decision boundary. Report the loss, L. (10 pts)



Loss: 1.376342

2. Train LR on the training set only (`X_train,y_train`) and then compute and report the loss on the test set (`X_test,y_test`). In your own words, explain why, in general, the loss on a held out test set gives a better estimate for how the model will perform in the "real world". (10 pts)

   Loss: 0.252447216254

   In general, the loss on a held out test set gives a better estimate for how the model will perform in the "real world" because when our agent begins interacting with the "real world", it generally does not have the test cases, just the training set. In this scenario, the loss on the test set will depend on the fitting and training done with the training set. If we include the test set in training, our agent will have extra information on performing on a specific instance, which is not necessarily available prior to testing and adds bias. Thus, the loss on a held out test set is a more accurate value on performance.

2.2 Decision Trees

In this problem, we are revisiting the Decision Tree problem from the book (p. 697-704). We will be considering the same dataset as well (Figure 18.3).

1. Assume we are building our decision tree, starting at the root. Section 18.3.4 explains the reasoning for choosing to split on the *Patrons* attribute over the *Type* attribute. Instead, calculate the expected entropy of the *Estimated Wait Time* attribute (EST). Is the expected entropy higher or lower than Patrons ($\mathbb{E}_{v=\{None,Some,Full\}}[H(v)] = 0.459$)? Of those two, which attribute should you choose to split on when learning a decision tree and why? (10 pts)

EST Values

EST Value	Yes	No
0-10	4	2
10-30	1	2
30-60	1	1
> 60	0	2

$H(\frac{4}{6}) = -\frac{4}{6}log_2(\frac{4}{6}) - \frac{2}{6}log_2(\frac{2}{6}) = .918002$
$H(\frac{1}{2}) = -\frac{1}{2}log_2(\frac{1}{2}) - \frac{1}{2}log_2(\frac{1}{2}) = 1$
$H(1) = -(0)log_2(0) - log_2(1) = 0$

$.918002(\frac{6}{12}) + (1)(\frac{1}{6}) + (1)(\frac{1}{6}) = .459001 + .1666667 + .1666667 = .792334$

Since splitting on Patrons has an entropy of .459, EST has an entropy of .792334, and $.459 < .792334$, we should split on Patrons, since it has more information gain and more accurately distributes the samples according to True and False values.

2. (Bonus) Implement a decision tree and train it on Stuart Russel's restaurant preferences. We have provided you with template code for loading the dataset, training the model, and printing the resulting tree (see `decisionTree.py`). Draw the tree that your model learns (using Google Drive drawings) and display it below. (10 pts)

3 Clustering (20 pts)

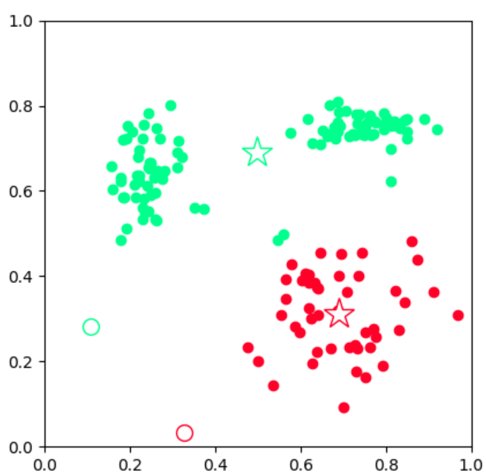
In this problem, you will learn to cluster using K-Means. We have provided you with template code for loading the dataset, training the model, and plotting the clusters (see `kmeans.py`).

1. Implement k-means clustering with k=3 using randomly initialized centers. Does k-means return the same result every time? If not, why not? (10 pts)

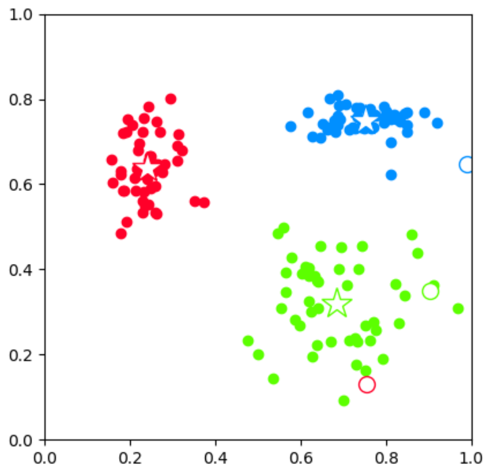
   K-means does not return the same results for randomly initialized centers. If we had poorly assigned clusters, where initialized centers were close together and far from data points, the algorithm would need more time to converge to a solution. Also, the order in which data points are acquired into a cluster determines which data points will be acquired in the future (due to the update function). This influences the assignment of data points near the "border" of the clusters.

2. Run k-means using randomly initialized centers and k=2,3,5. Provide a single plot of the clusters for each k. Which k is best? (10 pts)

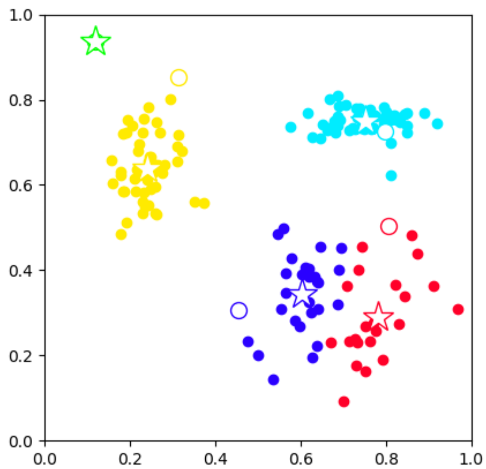
   Plot for k = 2:



   Plot for k = 3:



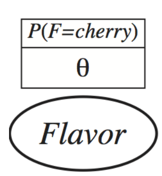
Plot for k = 5:



The best k is k = 3. K-means for k = 2 clearly underfits the data, as there are three distinct cluster of data points, and the green star centroid is between the two clumps (with no data points close to it). K-means with k = 5 overfits the data. This is apparent since there are no data points assigned to the green centroid. The two clumps (purple and red) in the bottom right corner of the graph could work as a classification (since the groups are different to an extent), but due to the lack of assignment of data points to the green centroid, k = 5 overfits the data. K-means with k = 3, fits the data best, since the assignment of data points to centroid makes sense. the centroids are central to their clusters and there is reasonable space between clusters, there is not underfitting or overfitting.

4 Bayesian Learning (30 pts)

In class (Lecture 20, slides 49-51), we derived the MAP result for the cherry-lime candy bag (a BN with one node) given the number of limes (l) and cherries (c) seen so far and a $beta[a, b]$ prior: $\theta_{\text{MAP}} = (c + a - 1)/(c + l + a + b - 2)$. In this problem,



we will assume the candy bag contains only limes.

1. Consider the five hypotheses with their priors listed at the bottom of page 802.

   (a) h_1: 100% cherry, $P(h_1) = 0.1$
   (b) h_2: 75% cherry, 25% lime, $P(h_2) = 0.2$
   (c) h_3: 50% cherry, 50% lime, $P(h_3) = 0.4$
   (d) h_4: 25% cherry, 75% lime, $P(h_4) = 0.2$
   (e) h_5: 100% lime, $P(h_5) = 0.1$

   Use full Bayesian learning (last Eqn of slide 9) to derive a formula for the probability that the $(N+1)$th candy is lime using all these hypotheses:

$$P(X_{N+1}|X_{1:N}) = \alpha \sum_{i=1}^{5} P(X_{N+1}|h_i)P(h_i) \prod_{j=1}^{N} P(X_j|h_i)$$

   where X_j represents the flavor of the jth candy drawn from the bag (with replacement). Use the priors listed above. You can check your result by plotting it and comparing it to Figure 20.1(b). You can check your intermediate results for $P(h_i|d)$ by comparing to Figure 20.1(a). (10 pts)

$$P(X_{N+1} = lime|X_{1:N}) = \alpha[(0)(.1)(0)^N + (.25)(.2)(.25)^N + (.5)(.4)(.5)^N + (.75)(.2)(.75)^N + (1.0)(.1)(1.0)^N]$$
$$P(X_{N+1} = lime|X_{1:N}) = \alpha[(.2)(.25)^{N+1} + (.4)(.5)^{N+1} + (.2)(.75)^{N+1} + (.1)]$$

$$P(X_{N+1} = cherry|X_{1:N}) = \alpha[(1.0)(.1)(0)^N + (.75)(.2)(.25)^N + (.5)(.4)(.5)^N + (.25)(.2)(.75)^N + (0)(.1)(1.0)^N]$$
$$P(X_{N+1} = cherry|X_{1:N}) = \alpha[(.75)(.2)(.25)^N + (.4)(.5)^{N+1} + (.25)(.2)(.75)^N]$$

   Determine α from the distribution over X:

$$\begin{aligned}
\Sigma_{x\in X} P(X_{N+1} = x|X_{1:N}) &= P(X_{N+1} = lime|X_{1:N}) + P(X_{N+1} = cherry|X_{1:N}) = 1 \\
&= \alpha[(.2)(.25)^{N+1} + (.4)(.5)^{N+1} + (.2)(.75)^{N+1} + (.1)] \\
&\qquad + \alpha[(.75)(.2)(.25)^N + (.4)(.5)^{N+1} + (.25)(.2)(.75)^N] = 1 \\
&= \alpha[(.2)(.25)^{N+1} + (.4)(.5)^{N+1} + (.2)(.75)^{N+1} + (.1) \\
&\qquad + (.75)(.2)(.25)^N + (.4)(.5)^{N+1} + (.25)(.2)(.75)^N] = 1 \\
&= \alpha[(.2)(.25)^{N+1} + (.8)(.5)^{N+1} + (.2)(.75)^{N+1} + (.1) \\
&\qquad + (.75)(.2)(.25)^N + (.25)(.2)(.75)^N] = 1 \\
\alpha &= \frac{1}{(.2)(.25)^{N+1} + (.8)(.5)^{N+1} + (.2)(.75)^{N+1} + (.1) + (.75)(.2)(.25)^N + (.25)(.2)(.75)^N}
\end{aligned}$$

   Determine $P(X_{N+1} = lime|X_{1:N})$:

$$P(X_{N+1} = lime|X_{1:N}) = \alpha[(.2)(.25)^{N+1} + (.4)(.5)^{N+1} + (.2)(.75)^{N+1} + (.1)]$$

$$\boxed{\mathrm{P}(\mathrm{X}_{N+1} = lime|X_{1:N}) = \frac{(.2)(.25)^{N+1}+(.4)(.5)^{N+1}+(.2)(.75)^{N+1}+(.1)}{(.2)(.25)^{N+1}+(.8)(.5)^{N+1}+(.2)(.75)^{N+1}+(.1)+(.75)(.2)(.25)^N+(.25)(.2)(.75)^N}}$$

2. Consider an infinite number of hypotheses with a hypothesis prior that is a beta distribution, beta[a,b]. Use full Bayesian learning to obtain a formula for the posterior as a function of N:

$$P(X_{N+1}|X_{1:N}) = \alpha \int P(X_{N+1}|\theta)P(\theta) \prod_{j=1}^{N} P(X_j|\theta) d\theta$$

You will also need to know the partition function of a beta:

$$Z = \int_{\theta} \theta^{a-1}(1-\theta)^{b-1} d\theta = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$$

You will need to install the scipy package to import the gamma function (`conda install scipy`). Report the formula below. (10 pts)

$$\begin{aligned}
P(X_{N+1} = lime|X_{1:N}) &= \alpha \int P(X_{N+1} = lime|\theta)P(\theta)\prod_{j=1}^{N} P(X_j|\theta)d\theta \\
&= \alpha \int (1-\theta) \cdot beta[a,b](\theta) \prod_{j=1}^{N}(1-\theta)d\theta \\
&= \alpha \int (1-\theta) \cdot \frac{1}{Z} \cdot (\theta)^{a-1} \cdot (1-\theta)^{b-1} \prod_{j=1}^{N}(1-\theta)d\theta \\
&= \hat{\alpha} \int (\theta)^{a-1} \cdot (1-\theta)^{b} \prod_{j=1}^{N}(1-\theta)d\theta \\
&= \hat{\alpha} \int (\theta)^{a-1} \cdot (1-\theta)^{b} \cdot (1-\theta)^{N} d\theta \\
&= \hat{\alpha} \int (\theta)^{a-1} \cdot (1-\theta)^{b+N} d\theta \\
&= \hat{\alpha} \frac{\Gamma(a) \cdot \Gamma(b+1+N)}{\Gamma(a+b+1+N)}
\end{aligned}$$

Determine $\hat{\alpha}$ from distribution over X!

$$\begin{aligned}
P(X_{N+1} = cherry|X_{1:N}) &= \alpha \int P(X_{N+1} = cherry|\theta)P(\theta)\prod_{j=1}^{N} P(X_j|\theta)d\theta \\
&= \alpha \int (\theta) \cdot beta[a,b](\theta) \prod_{j=1}^{N}(1-\theta)d\theta \\
&= \alpha \int (\theta) \cdot \frac{1}{Z} \cdot (\theta)^{a-1} \cdot (1-\theta)^{b-1} \prod_{j=1}^{N}(1-\theta)d\theta \\
&= \hat{\alpha} \int (\theta)^{a} \cdot (1-\theta)^{b-1} \prod_{j=1}^{N}(1-\theta)d\theta \\
&= \hat{\alpha} \int (\theta)^{a} \cdot (1-\theta)^{b-1} \cdot (1-\theta)^{N} d\theta \\
&= \hat{\alpha} \int (\theta)^{a} \cdot (1-\theta)^{b-1+N} d\theta \\
&= \hat{\alpha} \frac{\Gamma(a+1) \cdot \Gamma(b+N)}{\Gamma(a+b+1+N)}
\end{aligned}$$

Solve for $\hat{\alpha}$:

$$\begin{aligned}
\Sigma_{x \in X} P(X_{N+1} = x|X_{1:N}) &= P(X_{N+1} = lime|X_{1:N}) + P(X_{N+1} = cherry|X_{1:N}) = 1 \\
&= \hat{\alpha} \frac{\Gamma(a) \cdot \Gamma(b+1+N)}{\Gamma(a+b+1+N)} + \hat{\alpha} \frac{\Gamma(a+1) \cdot \Gamma(b+N)}{\Gamma(a+b+1+N)} = 1 \\
&= \hat{\alpha}[\frac{\Gamma(a) \cdot \Gamma(b+1+N)}{\Gamma(a+b+1+N)} + \frac{\Gamma(a+1) \cdot \Gamma(b+N)}{\Gamma(a+b+1+N)}] = 1 \\
&= \hat{\alpha}[\frac{\Gamma(a) \cdot \Gamma(b+1+N) + \Gamma(a+1) \cdot \Gamma(b+N)}{\Gamma(a+b+1+N)}] = 1 \\
\hat{\alpha} &= \frac{\Gamma(a+b+1+N)}{\Gamma(a) \cdot \Gamma(b+1+N) + \Gamma(a+1) \cdot \Gamma(b+N)}
\end{aligned}$$

Determine $P(X_{N+1} = lime|X_{1:N})$:

$$P(X_{N+1} = lime|X_{1:N}) = \hat{\alpha}\frac{\Gamma(a)\cdot\Gamma(b+1+N)}{\Gamma(a+b+1+N)}$$
$$P(X_{N+1} = lime|X_{1:N}) = [\frac{\Gamma(a+b+1+N)}{\Gamma(a)\cdot\Gamma(b+1+N)+\Gamma(a+1)\cdot\Gamma(b+N)}]\cdot\frac{\Gamma(a)\cdot\Gamma(b+1+N)}{\Gamma(a+b+1+N)}$$

$$\boxed{\mathrm{P}(\mathrm{X}_{N+1} = lime|X_{1:N}) = \frac{\Gamma(a)\cdot\Gamma(b+1+N)}{\Gamma(a)\cdot\Gamma(b+1+N)+\Gamma(a+1)\cdot\Gamma(b+N)}}$$

3. Plot the three different posteriors against each other: MAP vs finite vs infinite as a function of N ($N = 0$ to 10). Use $a = b = 2$ for the beta prior. (10 pts)