CSCI 5521 Spring 2017 Homework #3

Craig Ching
#1452647
chin0007@umn.edu

April 8, 2017

1 Problem 1

1.1 (a) Clearly show and explain the steps of the projected gradient descent algorithm for optimizing the regularized logistic regression objective function. The steps should include an exact expression for the gradient.

Our hypothesis function is:

$$h(\mathbf{w}^T x_i) = \frac{\exp(\mathbf{w}^T x_i)}{1 + \exp(\mathbf{w}^T x_i)} \tag{1}$$

Which gives us a regularized objective function of:

$$f(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\{-y_i \mathbf{w}^T x_i + \log(1 + \exp(\mathbf{w}^T x_i))\} + \frac{\lambda}{2}||\mathbf{w}||^2 \tag{2}$$

To derive the gradient of the $f(\mathbf{w})$, we will need the following:

$$\frac{\partial}{\partial \mathbf{w}} \log(1 + \exp(\mathbf{w}^T x_i)) = \frac{\exp(\mathbf{w}^T x_i) x_i}{1 + \exp(\mathbf{w}^T x_i)} \tag{3}$$

Using the chain rule:

$$\frac{\partial}{\partial \mathbf{w}} \log(1 + \exp(\mathbf{w}^T x_i)) = \frac{1}{1 + \exp(\mathbf{w}^T x_i)} \frac{\partial}{\partial \mathbf{w}}(1 + \exp(\mathbf{w}^T x_i)), \tag{4}$$

$$\frac{\partial}{\partial \mathbf{w}}(\exp(\mathbf{w}^T x_i) = \exp(\mathbf{w}^T x_i) \frac{\partial}{\partial \mathbf{w}}(\mathbf{w}^T x_i) = \exp(\mathbf{w}^T x_i) x_i \tag{5}$$

Therefore:

$$\frac{\partial}{\partial \mathbf{w}} \log(1 + \exp(\mathbf{w}^T x_i)) = \frac{\exp(\mathbf{w}^T x_i) x_i}{1 + \exp(\mathbf{w}^T x_i)}. \tag{6}$$

With that in hand, we can start:

$$\frac{\partial f(\mathbf{w})}{\partial \mathbf{w}} = -(y_i x_i) + \frac{x_i \exp(\mathbf{w}^T x_i)}{1 + \exp(\mathbf{w}^T x_i)} + \lambda \mathbf{w} \tag{7}$$

We can simplify that to:

$$\frac{\partial f(\mathbf{w})}{\partial \mathbf{w}} = x_i(\frac{\exp(\mathbf{w}^T x_i)}{1 + \exp(\mathbf{w}^T x_i)} - y_i) + \lambda \mathbf{w} \tag{8}$$

Which we can write as:

$$\nabla_f = x_i(h(\mathbf{w}^T x_i) - y_i) + \lambda \mathbf{w} \tag{9}$$

Our update rule for projected gradient descent is then

$$
\begin{aligned}
&\mathbf{w}_t = 0 \\
&\text{Repeat} \\
&\quad \mathbf{w}_{t+1}^{'} = \mathbf{w}_t - \eta[x_i(h(\mathbf{w}^T x_i) - y_i) + \lambda \mathbf{w}] \\
&\quad \mathbf{w}_{t+1} = \Pi_X(\mathbf{w}_{t+1}^{'}) \\
&\text{Until convergence}
\end{aligned} \tag{10}
$$

where $\Pi_X(\mathbf{x}) = arg\min_{y \in X} ||x - y||$

The expression $\Pi_X(\mathbf{w}_{t+1}^{'})$ has the effect of projecting the update $\mathbf{w}_{t+1}^{'}$ back to the point in the constrained region that is nearest to it if it should end up outside the constrained region.

The notation $\Pi_X(\mathbf{w}_{t+1}^{'})$ is from Bubeck. An alternative way to express it is given in the slides for lecture 10:

$$
\mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_{t+1}^{'}, & \text{if} ||\mathbf{w}_{t+1}^{'}|| \leq R \\ \frac{R}{||\mathbf{w}_{t+1}^{'}||}\mathbf{w}_{t+1}^{'}, & \text{if} ||\mathbf{w}_{t+1}^{'}|| > R \end{cases}
$$

Where R is a radius for a ball that constrains updates to the constrained region.

1.2 (b) Is the objective function strongly convex? Clearly explain your answer using the definition of strong convexity.

The objective function is strongly convex. My proof relies on the fact that if we can rewrite the function in a form such that the new form is strongly convex, we will conclude that the function is strongly convex. My proof also relies on the fact that properties of convexity (e.g. convexity, strong convexity, and smoothness) are additive properties.

Define:

$$
\begin{aligned}
g_1(\mathbf{w}) &= -y_i \mathbf{w}^T x_i \\
g_2(\mathbf{w}) &= \log(1 + \exp(\mathbf{w}^T x_i) \\
g_3(\mathbf{w}) &= \frac{\lambda}{2}||\mathbf{w}||^2 \\
f(\mathbf{w}) &= \frac{1}{n}\sum_{i=1}^{n} g_1(\mathbf{w}) + g_2(\mathbf{w}) + g_3(\mathbf{w})
\end{aligned} \tag{11}
$$

Then prove that g_1 and g_2 are convex and g_3 is strongly convex, then, by the additive properties of convexity, we show that our original function is strongly convex.

Proof that g_1 is convex:

$$
\begin{aligned}
-y_i \mathbf{w}_1^T x_i &\geq -y_i \mathbf{w}_2^T x_i + (\mathbf{w}_1 - \mathbf{w}_2)(-y_i x_i) \\
(\frac{1}{y_i x_i})(-y_i \mathbf{w}_1^T x_i) &\geq (\frac{1}{y_i x_i})(-y_i \mathbf{w}_2^T x_i + (\mathbf{w}_1 - \mathbf{w}_2)(-y_i x_i)) \\
-\mathbf{w}_1 &\geq -\mathbf{w}_2 - \mathbf{w}_1 + \mathbf{w}_2 \\
-\mathbf{w}_1 &\geq -\mathbf{w}_1
\end{aligned} \tag{12}
$$

To prove that g_2 is convex, we first make a simplification and assume that the $\exp(\mathbf{w}^T x_i)$ term dominates in the log, and we start by rewriting g_2 as:

$$
\begin{aligned}
g_2(x) &= \log(\exp(\mathbf{w}^T x_i)) \\
&= \mathbf{w}^T x_i
\end{aligned} \tag{13}
$$

Then we can prove convexity:

$$
\begin{aligned}
\mathbf{w}_1^T x_i &\geq \mathbf{w}_2^T x_i + (\mathbf{w}_1 - \mathbf{w}_2) x_i \\
(\frac{1}{x_i})(\mathbf{w}_1^T x_i) &\geq (\frac{1}{x_i})(\mathbf{w}_2^T x_i + (\mathbf{w}_1 - \mathbf{w}_2) x_i) \\
\mathbf{w}_1 &\geq \mathbf{w}_2 + \mathbf{w}_1 - \mathbf{w}_2 \\
\mathbf{w}_1 &\geq \mathbf{w}_1
\end{aligned} \tag{14}
$$

Proof that g_3 is strongly convex:

$$\begin{aligned}\mathbf{w}_1^2 &\geq \mathbf{w}_2^2 + (\mathbf{w}_1 - \mathbf{w}_2)2\mathbf{w}_2 + \frac{\alpha}{2}(\mathbf{w}_1 - \mathbf{w}_2)^2 \\ \mathbf{w}_1^2 &\geq \mathbf{w}_2^2 + (\mathbf{w}_1 - \mathbf{w}_2)2\mathbf{w}_2 + \frac{\alpha}{2}(\mathbf{w}_1 - \mathbf{w}_2)^2\end{aligned} \tag{15}$$

At this point, we're going to assume that $\alpha = 2$:

$$\begin{aligned}\mathbf{w}_1^2 &\geq \mathbf{w}_2^2 + (\mathbf{w}_1 - \mathbf{w}_2)2\mathbf{w}_2 + (\mathbf{w}_1 - \mathbf{w}_2)^2 \\ \mathbf{w}_1^2 &\geq \mathbf{w}_2^2 + 2\mathbf{w}_1\mathbf{w}_2 - 2\mathbf{w}_2^2 + \mathbf{w}_1^2 - 2\mathbf{w}_1\mathbf{w}_2 + \mathbf{w}_2^2 \\ \mathbf{w}_1^2 &\geq \mathbf{w}_1^2, \text{ if } \alpha \leq 2\end{aligned} \tag{16}$$

So, if $\alpha \leq 2$ then g_3 is strongly convex. Therefore:

$$\begin{aligned}g_1(x) &\geq -y_i\mathbf{w}^T x_i + (x - y)(-y_i x_i) \\ g_2(x) &\geq \log(1 + \exp(\mathbf{w}^T x_i) + (x - y)\frac{x_i \exp(\mathbf{w}^T x)}{1 + \exp(\mathbf{w}^T x_i)} \\ g_1(x) + g_2(x) &\geq -y_i\mathbf{w}^T x_i + \log(1 + \exp(\mathbf{w}^T x_i) + [(x - y)(-y_i x_i) + (x - y)\frac{x_i \exp(\mathbf{w}^T x)}{1 + \exp(\mathbf{w}^T x_i)}] \\ g_1(x) + g_2(x) &\geq -y_i\mathbf{w}^T x_i + \log(1 + \exp(\mathbf{w}^T x_i) + (x - y)[(-y_i x_i) + \frac{x_i \exp(\mathbf{w}^T x)}{1 + \exp(\mathbf{w}^T x_i)}] \\ f(y) &= -y_i\mathbf{w}^T x_i + \log(1 + \exp(\mathbf{w}^T x_i) \\ \nabla f(y) &= (-y_i x_i) + \frac{x_i \exp(\mathbf{w}^T x)}{1 + \exp(\mathbf{w}^T x_i)} \\ f(x) &\geq f(y) + (x - y)\nabla f(y) + \frac{\lambda}{2}||\mathbf{w}||^2, \text{ if } \lambda = \alpha \leq 2\end{aligned} \tag{17}$$

Therefore since our function can be written in the form for a strongly convex function and we've proved convexity and strong-convexity of the components, we conclude that the objective function is strongly convex.

1.3 (c) Is the objective function smooth? Clearly explain your answer using the definition of smoothness.

The objective function is smooth. As in 1b, we will examine additive components, but this time, all components must be smooth in order for our objective to be smooth.

Define:

$$\begin{aligned}g_1(\mathbf{w}) &= -y_i\mathbf{w}^T x_i \\ g_2(\mathbf{w}) &= \log(1 + \exp(\mathbf{w}^T x_i) \\ g_3(\mathbf{w}) &= \frac{\lambda}{2}||\mathbf{w}||^2 \\ f(\mathbf{w}) &= \frac{1}{n}\sum_{i=1}^{n} g_1(\mathbf{w}) + g_2(\mathbf{w}) + g_3(\mathbf{w})\end{aligned} \tag{18}$$

Prove that g_1, g_2 and g_3 are smooth, then, by the additive properties of smoothness, we show that our original function is also smooth. For all three functions we're going to rely on the fact that a function is smooth if the derivative is continuous for our domain.

$$\nabla g_1 = -y_i x_i \tag{19}$$

∇g is simply a constant function and is continuous for all $\mathbf{w}$, therefore g_1 is smooth.

To prove that g_2 is smooth, we start with the derivative:

$$\nabla g_2 = \frac{\exp(\mathbf{w}^T x_i)x_i}{1 + \exp(\mathbf{w}^T x_i)} \tag{20}$$

This function is continuous for all $\mathbf{w}^T x$, $\lim_{\mathbf{w}^T x \to -\infty} \nabla g_2 = 0$ and $\lim_{\mathbf{w}^T x \to \infty} \nabla g_2 = 1$, therefore g_2 is smooth.

To prove g_3 is smooth, we start with the derivative:

$$\nabla g_3 = \lambda||\mathbf{w}|| \tag{21}$$

∇g_3 is defined for all $\mathbf{w}$ and is, therefore, smooth.

Since g_1, g_2, and g_3 are all smooth, we conclude that the objective function is also smooth.

1.4 (d) Let $\mathbf{w}_T$ be the iterate after T steps of the projected gradient descent algorithm. What is a bound on the difference $f(\mathbf{w}_T) - f(\mathbf{w}^*)$? Clearly explain all quantities in the bound.

$$f(\mathbf{x}_T) - f(\mathbf{x}^*) \leq \frac{\beta}{2} \exp(-\frac{4T}{\frac{\beta}{\alpha} + 1})||\mathbf{x}_0 - \mathbf{x}^*||^2 \tag{22}$$

Since our function is strongly convex and smooth, we have the rate of convergence given by the above equation. The rate at which $\mathbf{x}_T$, our estimate after T iterations, approaches our optimum, $\mathbf{x}^*$ with a fixed step size, is exponential, given roughly by $\exp(-CT)$.

2 Problem 2

2.1 (a) In your own words, describe the EM algorithm for mixture of Gaussians, highlighting the two key steps (E- and M-), illustrating the methods used in the steps on a high level, and what information they need.

Expectation-Maximization (EM) is an unsupervised technique where the training data is given as $X = \{\mathbf{x}^t\}_t$, the significance being that we don't have the labels $\mathbf{r}^t$. The goal of EM is, then, to estimate the labels representing the components (which are analagous to classes in the supervised setting).

EM consists of two steps, the E-step and the M-step. We describe the EM algorithm for the multivariate Gaussian mixture model.

During the E-step, the goal is to estimate the latent labels $\mathbf{z}_i^t$ given current estimates of the component prior, mean, and covariance. During the M-step we update the component prior, mean, and covariance given the labels estimated in the E-step. We define $\mathbf{z}^t$ to be a vector of indicator variables where $z_i^t = 1$ if $\mathbf{x}^t$ belongs to the cluster G_i and 0 otherwise.

Before we can begin EM, though, we need some initial estimates for the component parameters. We do this by running the k-means clustering algorithm to give us initial estimates of the component mean, $\mathbf{m}_i$, covariance, $\mathbf{S}_i$ and the prior $\sum_t b_i^t/N$. Note that here b_i^t are the labels given by the k-means algorithm. Once we have our initial component parameters and our labels, we begin with the E-step.

In the E-step, the goal is to estimate the labels from the estimated component parameters Φ. We define:

$$\mathcal{Q}(\Phi|\Phi^l) = \sum_t \sum_i E[z_i^t|X, \Phi^l][\log \pi_i + \log p_i(\mathbf{x}^t|\Phi^l] \tag{23}$$

where:

$$E[z_i^t|X, \Phi^l] = P(G_i|\mathbf{x}^t, \Phi^l) \equiv h_i^t \tag{24}$$

This basically says that the expected value of the hidden variable z_i^t is the posterior probability (h_i^t) that $\mathbf{x}^t$ is generated by component G_i Note that since h_i^t is a probability, its value is between 0 and 1 and is a soft label, compared to the hard labels of k-means. In the above equations, because we are describing EM for Gaussian Mixtures, Φ represents the estimated component prior, mean, and covariance as mentioned above.

In the M-step, we maximize $\mathcal{Q}$ to get the next set of parameters Φ^l, the component prior, mean, and covariance:

$$\Phi^l = arg \max_{\Phi} \mathcal{Q}(\Phi|\Phi^l) \tag{25}$$

Doing so estimates the component prior:

$$\pi_i = \frac{\sum_t h_i^t}{N} \tag{26}$$

Descriptions of the Gaussian component parameters estimated by the M-step are given below. Once the new parameters Φ^l are estimated, then we begin again with the E-step. Note that EM takes an infinite number of iterations to converge, so we stop when the parameters don't change much or when we've reached some maximum number of iterations. Also note that if we knew which component $\mathbf{x}_t$ came from, we wouldn't need the E-step and we could run one M-step to get our classification.

2.2 (b) Assuming the posterior probabilities h_i^t are known, show the estimates of the component prior, mean, and covariance $\pi_i, \mu_i, \Sigma_i, i = 1, \ldots, N$ given by the M-step (you do not need to show how they are derived)

The component prior is generally given by the following equation, dependent on the posterior probability h_i^t estimated from the E-step:

$$\pi_i = \frac{\sum_t h_i^t}{N} \tag{27}$$

Since h_i^t is a probability between 0 and 1, π_i is estimated by the proportion of data points for the component G_i
The component mean is given by:

$$\mathbf{m}_i^{l+1} = \frac{\sum_t h_i^t \mathbf{x}^t}{\sum_t h_i^t} \tag{28}$$

Again, we use the posterior probability h_i^t to estimate the component mean to be used in the next iteration of the E-step.
The component covariance is given by:

$$\mathbf{S}_i^{l+1} = \frac{\sum_t h_i^t (\mathbf{x}^t - \mathbf{m}_i^{l+1})(\mathbf{x}^t - \mathbf{m}_i^{l+1})^T}{\sum_t h_i^t} \tag{29}$$

Again, the use of h_i^t is obvious in the estimation of $\mathbf{S}_i^{l+1}$. Note the similarities of these estimations to their classification counterparts in the multivariate Gaussian distribution:

$$\begin{aligned} \hat{P}(C_i) &= \frac{\sum_t r_i^t}{N} \\ \mathbf{m}_i &= \frac{\sum_t r_i^t \mathbf{x}^t}{\sum_t r_i^t} \\ \mathbf{S}_i &= \frac{\sum_t r_i^t (\mathbf{x}^t - \mathbf{m}_i)(\mathbf{x}^t - \mathbf{m}_i)^T}{\sum_t r_i^t} \end{aligned} \tag{30}$$

The difference being that in classification, we have the labels r_i^t but in EM we have posterior probability estimates of the labels h_i^t.

2.3 (c) Assuming the component prior, mean, and covariance $\pi_i, \mu_i, \Sigma_i, i = 1, \ldots N$, are known, show how the posterior probabilities, h_i^t are computed in the E-step

The component posterior probabilities h_i^t are computed be the following:

$$h_i^t = \frac{\pi_i |\mathbf{S}_i|^{-1/2} \exp[-(1/2)(\mathbf{x}^t - \mathbf{m}_i^t)^T \mathbf{S}_i^{-1} (\mathbf{x}^t - \mathbf{m}_i^t)]}{\sum_j \pi_j |\mathbf{S}_j|^{-1/2} \exp[-(1/2)(\mathbf{x}^t - \mathbf{m}_j)^T \mathbf{S}_j^{-1} (\mathbf{x}^t - \mathbf{m}_j)]} \tag{31}$$

The numerator consists of the multivairite Gaussian density equation for the component G_i calculated using the estimates π_i, $\mathbf{m}_i$, and $\mathbf{S}_i$ and the denominator is the total density estimated by the Gaussian density for all components G_j and acts as a normalizing term making h_i^t a proper probability.

3 Problem 3

3.1 Description

For this problem I developed a 2-class Logistic Regression implementation. To solve the logistic regression problem, I used batch gradient descent with a constant step size $\eta = 0.01$, 500 iterations, and a regularization parameter $\lambda = 1.0$. There is no checking of the loss function for convergence, I simply execute for the number of iterations and use the result. To find the right step-size and iterations, I played around with both (a lot!) until I found a combination that gave good results. For the most part, the error rates of my implementation are competitive with those of the logistic regression implementation in scikit-learn.

3.2 Results

method: MyLogisticReg2
dataset: Boston50

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	mean	std dev
0.1275	0.1881	0.1881	0.0792	0.1386	0.1443	0.0410

method: MyLogisticReg2
dataset: Boston75

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	mean	std dev
0.0686	0.0891	0.0891	0.1287	0.0990	0.0949	0.0196

method: LogisticRegression
dataset: Boston50

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	mean	std dev
0.1471	0.0891	0.1881	0.1485	0.0792	0.1304	0.0407

method: LogisticRegression
dataset: Boston75

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	mean	std dev
0.1373	0.0693	0.0990	0.0891	0.1287	0.1047	0.0252

4 Problem 4

4.1 Description

For this problem I developed a k-class Logistic Regression implementation, using the softmax function as the hypothesis. To solve the logistic regression problem, I used batch gradient descent with a constant step size $\eta = 0.00001$, 1000 iterations, and a regularization parameter $\lambda = 1.0$. There is no checking of the loss function for convergence, I simply execute for the number of iterations and use the result. To find the right step-size and iterations, I played around with both (a lot!) until I found a combination that gave good results. For the most part, the error rates of my implementation are competitive with those of the logistic regression implementation in scikit-learn.

4.2 Results

method: MyLogisticRegGen
dataset: Digits

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	mean	std dev
0.0361	0.0139	0.0362	0.0279	0.0362	0.0301	0.0087

method: LogisticRegression
dataset: Digits

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	mean	std dev
0.0333	0.0389	0.0390	0.0390	0.0362	0.0373	0.0022