CS 189 Introduction to Machine Learning
Fall 2017

DIS10

1 Fun with Lagrange Multipliers

(a) Minimize the function

$$f(x,y) = x + 2y$$

such that

$$x^2 + y^2 = 3.$$

Solution: The Lagrangian is:

$$L(x,y,\lambda) = x + 2y + \lambda(x^2 + y^2 - 3)$$

Taking all of the partial derivatives and setting them to 0, we get this system of equations:

$$\lambda x = -\frac{1}{2}$$
$$\lambda y = -1$$
$$x^2 + y^2 = 3$$

We can infer that $y = 2x$. Plugging this into the constraint, we have:

$$x^2 + 4x^2 = 3$$

which shows that $x = \pm\sqrt{\frac{3}{5}}$. We have two critical points, $(-\sqrt{\frac{3}{5}}, -2\sqrt{\frac{3}{5}})$ and $(\sqrt{\frac{3}{5}}, 2\sqrt{\frac{3}{5}})$. Plugging these into our objective function f, we find that the minimizer is the former, with a value of $-3\sqrt{\frac{3}{5}}$.

(b) Minimize the function

$$f(x,y,z) = x^2 - y^2$$

such that

$$x^2 + 2y^2 + 3z^2 = 1.$$

Solution: The Lagrangian is:

$$x^2 - y^2 + \lambda(x^2 + 2y^2 + 3z^2 - 1)$$

Taking all of the partial derivatives and setting them to 0, we get this system of equations:

$$x = \lambda x$$
$$y = 2\lambda y$$
$$0 = \lambda z$$
$$x^2 + 2y^2 + 3z^2 = 1$$

To solve this, we look at several cases:

Case 1: $\lambda = 0$. This implies that $x = y = 0$, and $z = \pm\frac{1}{3}$. We have two critical points: $(0,0,\pm\frac{1}{3})$.

Case 2: $\lambda \neq 0$. z must be 0.

Case 2a: $x = 0$. The constraint gives us that $y = \pm\frac{1}{\sqrt{2}}$. This gives us another two critical points: $(0,\pm\frac{1}{\sqrt{2}},0)$.

Case 2b: $y = 0$. The constraint gives us $x = \pm 1$, giving us another two critical points: $(\pm 1,0,0)$.

Plugging in all of our critical points, we find that $(0,\pm\frac{1}{\sqrt{2}},0)$ minimizes our function with a value of $-\frac{1}{2}$.

2 Support Vector Machines

(a) We typically frame an SVM problem as trying to *maximize* the margin. Explain intuitively why a bigger margin will result in a model that will generalize better, or perform better in practice.

Solution: One intuition is that if points are closer to the border, we are less certain about their class. Thus, it would make sense to create a boundary where our "certainty" is highest about all the training set points.

Another intuition involves thinking about the process that generated the data we are working with. Since it's a noisy process, if we drew a boundary close to one of our training points of some class, it's very possible that a point of the same class will be generated across the boundary, resulting in an incorrect classification. Therefore it makes sense to make the boundary as far away from our training points as possible.

(b) Will moving points which are not support vectors further away from the decision boundary effect the SVM's hinge loss?

Solution: No, the hinge loss is defined as $\sum_{i=1}^{N} max(0, 1 - y_i(\vec(w) * \vec(x) + b))$. For nonsupport vectors, the right hand side of the max function is already negative and moving the point further away from the boundary will make it only more negative. The max will return a zero regardless. This means that the loss function and the consequent decision boundary is entirely determined by the orientation of the support vectors and nothing else.

(c) Show that the width of an SVM slab with linearly separable data is $\frac{2}{\|w\|}$.

Solution: The width of the margin is defined by the points that lie on it, also called support vectors. Let's say we have a point, $\vec{x}'$, which is a support vector. The distance between $\vec{x}'$ and the separating hyperplane can be calculated by projecting the vector starting at the plane and ending at $\vec{x}$ onto the plane's unit normal vector. The equation of the plane is $\vec{w}^T\vec{x} + b = 0$.

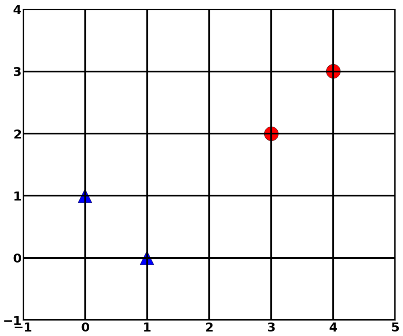
Since $\vec{w}$ by definition is orthogonal to the hyperplane, we want to project $\vec{x}' - \vec{x}$ onto the unit vector normal to the hyperplane, $\frac{\vec{w}}{\|\vec{w}\|}$.

$$\frac{\vec{w}^T}{\|\vec{w}\|}(\vec{x}' - \vec{x}) = \frac{1}{\|\vec{w}\|}(\vec{w}^T\vec{x}' - \vec{w}^T\vec{x}) = \frac{1}{\|\vec{w}\|}(\vec{w}^T\vec{x}' + b - \vec{w}^T\vec{x} - b)$$

Since we set $\vec{w}^T\vec{x}' + b = 1$ (or -1) and by definition, $\vec{w}^T\vec{x} + b = 0$, this quantity just turns into $\frac{1}{\|\vec{w}\|}$, or $\vec{1}\|\vec{w}\|$, so the distance is the absolute value, $\frac{1}{\|\vec{w}\|}$.

Since the margin is half of the slab, we double it to get the full width of $\frac{2}{\|\vec{w}\|}$.

(d) You are presented with the following set of data (triangle = +1, circle = -1):



Find the equation (by hand) of the hyperplane $\vec{w}^T x + b = 0$ that would be used by an SVM classifier. Which points are support vectors?

Solution: The equation of the hyperplane will pass through point $(2,1)$, with a slope of -1. The equation of this line is $x_1 + x_2 = 3$. We know that from this form, $w_1 = w_2$. We also know that the at the support vectors, $w^T x + b = \pm 1$. This gives us the equations:

$$1w_1 + 0w_2 + b = 1$$

$$3w_1 + 2w_2 + b = -1$$

Solving this system of equations, we get $\vec{w} = [-\frac{1}{2}, -\frac{1}{2}]^T$ and $b = \frac{3}{2}$.

The support vectors are $(1,0), (0,1)$, and $(3,2)$.

3 Simple SGD updates

Let us consider a simple least squares problem, where we are interested in optimizing the function

$$F(w) = \frac{1}{2n}\|Aw - y\|_2^2 = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(a_i^\top w - y_i)^2.$$

(a) What is the closed form OLS solution? **What is the time complexity of computing this solution in terms of flops?**

Solution: The closed form solution is

$$\hat{w} = (A^\top A)^{-1} A^\top y. \tag{1}$$

This takes time $d^2n + dn + d^3$ to compute – d^2n to find $A^\top A$ since it takes n multiplications to compute each entry of this $d \times d$ matrix, dn to find $A^\top y$ since it takes d multiplications to compute each entry of this n-vector, and d^3 time to invert a matrix via Gaussian elimination.

(b) Write down the gradient descent update. **What is the time complexity of computing an ε optimal solution?** **Solution:** For gradient descent, we have the update $w_{t+1} = w_t - \frac{\gamma}{n} A^\top (Aw_t - y)$. We know from HW that denoting $e_k = \|w_k - w^*\|_2^2$ and letting Q denote the condition number of $A^\top A$, we have $e_k \leq \left(\frac{Q-1}{Q+1}\right)^2 e_{k-1}$. We therefore obtain geometric convergene to the optimum, and the number of iterations is roughly

$T \sim Q \log(1/\varepsilon)$ to converge to within ε of optimum (write this out to see why, using the approximation $1 - x \sim e^{-x}$). Also note that during each iteration, we perform work dn, since Aw_t takes dn time to compute, and performing the multiplication $A^\top(Aw_t - y)$ takes dn time as well. So the total cost is

$dn \log(1/\varepsilon)$.

(c) Write down the stochastic gradient descent update. **What is the time complexity of computing an ε optimal solution?** You may want to quickly go through a derivation here. **What happens when $Aw^* = y$?**

Discuss why you would use any of these methods for your problem.

Solution: Let us derive the SGD convergence rate from first principles. We have the update equation

$$w_{t+1} = w_t - \gamma a_J (a_J^\top w_t - y_J),$$

where J is chosen uniformly at random from the set $\{1, 2, 3, \ldots, n\}$. Notice that this makes sense as a noisy gradient, since $\mathbb{E}_J[a_J(a_J^\top w_t - y_J)] = \nabla f_J(w_t) = \frac{1}{n}(A^\top A w_t - A^\top y) = \nabla f(w_t)$ and so the gradient estimate is unbiased.

Let us now compute the convergence rate. We have

$$\|w_{t+1} - w^*\|_2^2 = \|w_t - w^*\|_2^2 + \gamma^2 \|\nabla f_J(w_t)\|_2^2 - \gamma (w_t - w^*)^\top \nabla f_J(w_t).$$

Now notice that there are two sources of randomness in the RHS. The iterate w_t is in itself random, since we have chosen random indices up until that point. The index J is also random. Crucially, these two sources of randomness are independent of each other.

In particular, we may now take the inner product and compute the expectation over the index

J to obtain

$$\begin{aligned}\mathbb{E}_J\left[\gamma(w_t - w^*)^\top \nabla f_J(w_t)\right] &= \gamma(w_t - w^*)^\top \nabla \mathbb{E}[f_J(w_t)] \\ &= \gamma(w_t - w^*)^\top \nabla f(w_t) \\ &= \gamma(w_t - w^*)^\top A^\top A(w_t - w^*) \\ &= \|A(w_t - w^*)\|_2^2 \\ &\geq \lambda_{\min}(A^\top A)\|w_t - w^*\|_2^2,\end{aligned}$$

where in the last step, we have used a simple eigenvalue bound; go back and look at HW6 if this is not clear.

Letting $m = \lambda_{\min}(A^\top A)$, we have

$$\mathbb{E}_J\left[\|w_{t+1} - w^*\|_2^2\right] \leq (1 - \gamma m)\|w_t - w^*\|_2^2 + \gamma^2 \mathbb{E}_J\|\nabla f_J(w_t)\|_2^2.$$

We will now make some additional assumptions. First, we assume that $\|a_i\|_2 = 1$. Next, we assume that we will always stay within a region such that the function $F(w) \leq M^2$ (note that we can do this by evaluating the loss and ensuring that we don't take a step if this condition is violated, or by projection.) Consequently, we have

$$\begin{aligned}\mathbb{E}_J\|\nabla f_J(w_t)\|_2^2 &= \frac{1}{n}\sum_{i=1}^{n}\|a_i(a_i^\top w_t - y_i)^2\|_2^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}\|a_i\|_2^2(a_i^\top w_t - y_i)^2 \\ &= 2F(w_t) \\ &\leq 2M^2.\end{aligned}$$

We are now in a position to complete the analysis. We have

$$\mathbb{E}\left[\|w_{t+1} - w^*\|_2^2\right] \leq (1 - \gamma m)\mathbb{E}\left[\|w_t - w^*\|_2^2\right] + 2\gamma^2 M^2,$$

where we have taken an additional expectation with respect to the randomness up to and including time t. Rolling this out (think of induction in reverse), we have

$$\mathbb{E}\left[\|w_{t+1} - w^*\|_2^2\right] \leq (1 - \gamma m)^2\mathbb{E}\left[\|w_{t-1} - w^*\|_2^2\right] + 2\gamma^2 M^2(1 - \gamma m) + 2\gamma^2 M^2.$$

Do you spot the pattern? We effectively have

$$\begin{aligned}\mathbb{E}\left[\|w_t - w^*\|_2^2\right] &\leq (1 - \gamma m)^t\mathbb{E}\left[\|w_0 - w^*\|_2^2\right] + 2\gamma^2 M^2\sum_{i=0}^{t-1}(1 - \gamma m)^i \\ &\leq (1 - \gamma m)^t\mathbb{E}\left[\|w_0 - w^*\|_2^2\right] + 2\gamma^2 M^2\sum_{i=0}^{\infty}(1 - \gamma m)^i \\ &= (1 - \gamma m)^t\mathbb{E}\left[\|w_0 - w^*\|_2^2\right] + 2\gamma^2 M^2\frac{1}{\gamma m} \\ &= (1 - \gamma m)^t\mathbb{E}\left[\|w_0 - w^*\|_2^2\right] + 2\gamma\frac{M^2}{m}.\end{aligned}$$

Now if we want the LHS to be less than ε, it suffices to set each of the above terms to be less than $\varepsilon/2$. In particular, we have the relations

$$2\gamma\frac{M^2}{m} \leq \varepsilon/2 \text{ and}$$
$$(1-\gamma m)^t \mathbb{E}\left[\|w_0 - w^*\|_2^2\right] \leq \varepsilon/2.$$

Doing some algebra, we are led to the choices $\gamma = \frac{\varepsilon m}{4M^2}$, and

$$t = \frac{1}{\gamma m}\log(2D_0/\varepsilon) = \frac{4M^2}{m^2}\frac{\log(2D_0/\varepsilon)}{\varepsilon},$$

where D_0 denotes our initial distance to optimum.

In effect, we converge in $\frac{1}{\varepsilon}\log(1/\varepsilon)$ iterations, and each iteration takes $O(d)$ time (why?).

Let us now compare all three algorithms. Clearly, GD beats OLS provided $dn\log 1/\varepsilon < d^2 n$, which happens when $d > \log(1/\varepsilon)$. Think about what this means! Setting $\varepsilon = 10^{-6}$ (almost optimum), we see that GD wins for any problem in which $d > 20$!

Comparing SGD and GD, the quantities are $dn\log(1/\varepsilon) \sim \frac{d}{\varepsilon}\log(1/\varepsilon)$. In other words, SGD provides gains in convergence when $n \sim 1/\varepsilon$, i.e., when we have sufficiently many samples. There are also other advantages to SGD that this analysis doesn't quite illustrate; for instance scalability and generalization ability.

Comparing SGD and OLS, we see that SGD wins when $d^2 n > \frac{d}{\varepsilon}\log(1/\varepsilon)$, and so the relevant comparison is between dn and $1/\varepsilon$. SGD again wins for moderately sized problems.

(d) Write down the SGD update for logistic regression on two classes

$$F(w) = \frac{1}{n}\sum_{i=1}^{n} y_i \log\frac{1}{\sigma(w^\top x_i)} + (1-y_i)\log\frac{1}{1-\sigma(w^\top x_i)}.$$

Discuss why this is equivalent to minimizing a "cross-entropy" loss.