10-315 Introduction to Machine Learning: Homework 3

Due 11:59 p.m. Monday, February 25, 2019

Instructions

- **Submit your homework on time electronically by submitting to Autolab by 11:59 pm, Monday, February 25, 2019**.

  We recommend that you use LaTeX, but we will other typesetting as well. You do not need to download any extra files from Autolab. To submit this homework, you should submit a pdf of your solutions on Autolab by navigating to Homework 3 and clicking the "Submit File" button.

- **Late homework policy**: Homework 3 is worth full credit if submitted before the due date. Up to 50% credit can be received if the submission is less than 48 hours late. The lowest homework grade at the end of the semester will be dropped. Please talk to the instructor in the case of extreme extenuating circumstances.

- **Collaboration policy**: You are welcome to collaborate on any of the questions with anybody you like. However, you must write up your own final solution, and you must list the names of anybody you collaborated with on this assignment.

Problem 1: Gradient Descent

Let $f : \mathbb{R}^d \mapsto \mathbb{R}$ be a differentiable function. Recall that the gradient descent (GD) algorithm starts at some point $\mathbf{x}^{(\mathbf{0})} = \{x_1^{(0)}, \ldots, x_d^{(0)}\}$, where subscript i denotes the coordinate and superscript (k) denotes the kth step of the algorithm, and iteratively updates the position with the following update rule at each coordinate i:

$$x_i^{(k+1)} \leftarrow x_i^{(k)} - \eta \partial_{x_i} f(\mathbf{x}^{(\mathbf{k})})$$

Here $\partial_{x_i} f : \mathbb{R}^d \mapsto \mathbb{R}$ is the derivative of the function with respect to the ith coordinate, $\partial_{x_i} f(\mathbf{x}^{(\mathbf{k})})$ is its evaluation at $\mathbf{x}^{(\mathbf{k})}$, and $\eta > 0$ is called the *step-size* or *learning rate*.

1.1: Implementation

1. One Dimension [6 Points]

Let $f(x) = 4x^2 - 2x + 1$ and $\eta = 0.1$. First, write down an expression for $\partial_x f(x)$. Then, starting at $x^{(0)} = 1$, write down the current gradient $\partial_x f(x^{(k)})$, the new position $x^{(k+1)}$, and the new function value $f(x^{(k+1)})$ for steps $k = 0, 1, 2$ of GD.

2. Two Dimensions [9 Points]

Let $f(x_1, x_2) = x_1^2 + \sin(x_1 + x_2) + x_2^2$. First, write down the gradient descent update rule for any $\eta > 0$. Then, make the the following two plots, using contour plots for the function and arrows from one point of GD to the next (an example plot is provided in `gdplot.png`, created using example code in `gdplot.py`; both files will be available on Autolab). A contour plot is a way to draw a 3D surface by plotting the lines at which the function takes on specific values, similar to elevation maps of mountainous regions. You need only submit the plots and analysis; do not submit the code used.

1. The function on the domain $[\pm 4] \times [\pm 4]$ together with the points $(x_1^{(k)}, x_2^{(k)})$ for 10 steps of GD starting from $(x_1^{(0)}, x_2^{(0)}) = (3, -3)$ using $\eta = 0.4$.
2. The function on the domain $[\pm 4] \times [\pm 4]$ together with the points $(x_1^{(k)}, x_2^{(k)})$ for 10 steps of GD starting from $(x_1^{(0)}, x_2^{(0)}) = (3, -3)$ using $\eta = 0.8$.

What do you observe?

1.2: Analysis

Gradient descent is a local search algorithm: at each step, it observes information at the current point (the gradient) and moves to where this information tells it to go in order to minimize the function (the descent direction). Thus, the algorithm does not know how the function behaves globally. However, even though GD is a local search algorithm, in fact if f is particularly nice (convex and β-smooth - defined below) and has a minimum at $\mathbf{x}^*$ then there exists $\eta > 0$ for which $\mathbf{x}^{(\mathbf{k})} \to \mathbf{x}^*$ as $k \to \infty$. Here we will do a guided proof of this fact in the single-dimensional case.

1. Descent Lemma [12 Points]

Let us assume that f is β-smooth, which for $\beta > 0$ means that:

$$f(y) \leq f(x) + \partial_x f(x)(y - x) + \frac{\beta}{2}(y - x)^2$$

Suppose $x^{(k+1)} = x^{(k)} - \eta \partial_x f(x^{(k)})$ is a step of GD. We can substitute $x = x^{(k)}, y = x^{(k+1)}$ into the above expression to get

$$f(x^{(k+1)}) \leq f(x^{(k)}) + \partial_x f(x^{(k)}) \left(x^{(k+1)} - x^{(k)}\right) + \frac{\beta}{2} \left(x^{(k+1)} - x^{(k)}\right)^2$$

Please do the follow two steps:

1. Use the GD update rule to show that the above implies:

$$f(x^{(k+1)}) \leq f(x^{(k)}) - \eta \left(1 - \frac{\eta\beta}{2}\right) \left(\partial_x f(x^k)\right)^2$$

2. Give the range of values of η where $f(x^{(k+1)}) < f(x^{(k)})$, i.e. such that taking a gradient step decreases the objective value.

2. Convergence [Bonus: 10 Points]

Assume that there exists a unique minimizer of f, i.e. a point x^* such that $f(x^*) < f(x)\ \forall\ x \neq x^*$. Show that for $\eta = \frac{1}{\beta}$ we have $\partial_x f(x^{(k)}) \to 0$ as $k \to \infty$ and explain why this implies that $x^{(k)} \to x^*$ as $k \to \infty$. Hint: use the Descent Lemma above to show that for all $K \geq 1$, $\sum_{k=1}^{K} \left(\partial_x f(x^{(k)})\right)^2 \leq 2\beta(f(x^{(1)}) - f^*)$.

Note: A proof of convergence of gradient descent for more general functions can be found in *Understanding Machine Learning: From Theory to Algorithms* by Shalev-Schwartz and Ben-David.

Problem 2: Logistic Regression

2.1: Logistic regression in two dimensions

In this question, we will derive the logistic regression algorithm (the M(C)LE and its gradient). For simplicity, we assume the dataset is two-dimensional. Given a training set $\{(x^i, y^i), i = 1, \dots, n\}$ where $x^i \in \mathbb{R}^2$ is a feature vector and $y^i \in \{0, 1\}$ is a binary label, we want to find the parameters $\hat{w}$ that maximize the likelihood for the training set, assuming a parametric model of the form

$$p(y = 1|x; w) = \frac{1}{1 + \exp(-w_0 - w_1 x_1 - w_2 x_2)} = \frac{\exp(w_0 + w_1 x_1 + w_2 x_2)}{1 + \exp(w_0 + w_1 x_1 + w_2 x_2)}. \tag{1}$$

1. **[12 Points]** Below, we give a derivation of the conditional log likelihood. In this derivation, **provide a short justification for why each line follows from the previous one**.

$$\ell(w) \equiv \ln \prod_{j=1}^{n} p(y^j \mid x^j, w) \tag{2}$$

$$= \sum_{j=1}^{n} \ln p(y^j \mid x^j, w) \tag{3}$$

$$= \sum_{j=1}^{n} \ln \left(p(y^j = 1 \mid x^j, w)^{y^j} p(y^j = 0 \mid x^j, w)^{1-y^j} \right) \tag{4}$$

$$= \sum_{j=1}^{n} \left[y^j \ln p(y^j = 1 \mid x^j, w) + (1 - y^j) \ln p(y^j = 0 \mid x^j, w) \right] \tag{5}$$

$$= \sum_{j=1}^{n} \left[y^j \ln \frac{\exp(w_0 + w_1 x_1^j + w_2 x_2^j)}{1 + \exp(w_0 + w_1 x_1^j + w_2 x_2^j)} + (1 - y^j) \ln \frac{1}{1 + \exp(w_0 + w_1 x_1^j + w_2 x_2^j)} \right] \tag{6}$$

$$= \sum_{j=1}^{n} \left[y^j \ln \left(\exp(w_0 + w_1 x_1^j + w_2 x_2^j) \right) + \ln \left(\frac{1}{1 + \exp(w_0 + w_1 x_1^j + w_2 x_2^j)} \right) \right] \tag{7}$$

$$= \sum_{j=1}^{n} \left[y^j \left(w_0 + w_1 x_1^j + w_2 x_2^j \right) - \ln \left(1 + \exp(w_0 + w_1 x_1^j + w_2 x_2^j) \right) \right]. \tag{8}$$

Next, we will derive the gradient of the previous expression with respect to w_0, w_1, w_2, i.e., $\frac{\partial \ell(w)}{\partial w_i}$, where $\ell(w)$ denotes the log likelihood from part 1. We will perform a few steps of the derivation, and then ask you to do one step at the end. If we take the derivative of Expression 8 with respect to w_i for $i \in \{1, 2\}$, we get the following expression:

$$\frac{\partial \ell(w)}{\partial w_i} = \frac{\partial}{\partial w_i} \sum_{j=1}^{n} \left[y^j \left(w_0 + w_1 x_1^j + w_2 x_2^j \right) \right] - \frac{\partial}{\partial w_i} \sum_{j=1}^{n} \ln \left[1 + \exp \left(w_0 + w_1 x_1^j + w_2 x_2^j \right) \right]. \tag{9}$$

The blue expression is linear in w_i, so it can be simplified to $\sum_{j=1}^{n} y^j x_i^j$. For the red expression, we use the chain rule as follows (first we consider a single $j \in [1, n]$)

$$
\begin{aligned}
&\frac{\partial}{\partial w_i} \ln \left[1 + \exp\left(w_0 + w_1 x_1^j + w_2 x_2^j\right)\right] && (10)\\
&= \frac{1}{1 + \exp\left(w_0 + w_1 x_1^j + w_2 x_2^j\right)} \cdot \frac{\partial}{\partial w_i}\left(1 + \exp\left(w_0 + w_1 x_1^j + w_2 x_2^j\right)\right) && (11)\\
&= \frac{1}{1 + \exp\left(w_0 + w_1 x_1^j + w_2 x_2^j\right)} \cdot \exp\left(w_0 + w_1 x_1^j + w_2 x_2^j\right) \frac{\partial}{\partial w_i}\left(w_0 + w_1 x_1^j + w_2 x_2^j\right) && (12)\\
&= x_i^j \cdot \frac{\exp\left(w_0 + w_1 x_1^j + w_2 x_2^j\right)}{1 + \exp\left(w_0 + w_1 x_1^j + w_2 x_2^j\right)} && (13)
\end{aligned}
$$

2. **[12 Points]** Now use Equation 13 (and the previous discussion) to show that overall, Expression 9, i.e., $\frac{\partial \ell(w)}{\partial w_i}$, is equal to

$$\frac{\partial \ell(w)}{\partial w_i} = \sum_{j=1}^{n} x_i^j (y^j - p(y^j = 1 \mid x^j; w)) \tag{14}$$

Hint: does Expression 13 look like a familiar probability?

Since the log likelihood is concave, it is easy to optimize using gradient ascent. The final algorithm is as follows. We pick a step size η, and then perform the following iterations until the change is $< \epsilon$:

$$w_0^{(t+1)} = w_0^{(t)} + \eta \sum_j \left[y^j - p(y^j = 1 \mid x^j; w^{(t)})\right] \tag{15}$$

$$w_i^{(t+1)} = w_i^{(t)} + \eta \sum_j x_i^j \left[y^j - p(y^j = 1 \mid x^j; w^{(t)})\right]. \tag{16}$$

2.2: General questions about logistic regression

1. **[7 Points]** Explain why logistic regression is a discriminative classifier (as opposed to a generative classifier such as Naive Bayes).

2. **[7 Points]** Recall the prediction rule for logistic regression is if $p(y^j = 1 \mid x^j) > p(y^j = 0 \mid x^j)$, then predict 1, otherwise predict 0. What does the decision boundary of logistic regression look like? Justify your answer (e.g., try to write out the decision boundary as a function of w_0, w_1, w_2 and x_1^j, x_2^j).

Problem 3: Kernels

3.1: Kernel computation cost

1. **[7 Points]** Consider we have a two-dimensional input space such that the input vector is $x = (x_1, x_2)^T$. Define the feature mapping $\phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T$. What is the cooresponding kernel function, i.e. $K(x, z)$? Do not leave $\phi(x)$ in your final answer.

2. **[7 Points]** Suppose we want to compute the value of the kernel function $K(x, z)$ from the previous question, on two vectors $x, z \in \mathbb{R}^2$. How many additions and multiplications are needed if you

   (a) map the input vector to the feature space and the perform the dot product on the mapped features?

   (b) compute through the kernel function you derived in question 1?

3.2: Kernel functions

Consider the following kernel function:

$$K(x, x') = \begin{cases} 1, & \text{if } x = x' \\ 0, & \text{otherwise} \end{cases}$$

1. **[7 Points]** Prove this is a legal kernel. That is, describe an implicit mapping $\Phi : X \to \mathcal{R}^m$ such that $K(x, x') = \Phi(x) \cdot \Phi(x')$. (You may assume the instance space X is finite.)

2. **[7 Points]** In this kernel space, any labeling of points in X will be linearly separable. Justify this claim.

3. **[7 Points]** Since all labelings are linearly separable, this kernel seems perfect for learning any target function. Why is this actually a bad idea?