Discussion 4: Logistic Regression
Probabilistic Machine Learning, Fall 2016

1 Formulation and Coordinate Descent

1.1 Formulation

For dataset $\mathbf{x}_i \in \mathbb{R}^D$, $y_i \in \{0, 1\}$, $i = 1, ..., N$, prove the negative log-likelihood of logistic regression

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log \frac{1}{1+e^{-\boldsymbol{\theta}^\top \mathbf{x}_i}} + (1-y_i)\log \frac{e^{-\boldsymbol{\theta}^\top \mathbf{x}_i}}{1+e^{-\boldsymbol{\theta}^\top \mathbf{x}_i}}\right)$$

is convex w.r.t. $\boldsymbol{\theta}$.

Answer

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) =& -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log \frac{1}{1+e^{-\boldsymbol{\theta}^\top \mathbf{x}_i}} + (1-y_i)\log \frac{e^{-\boldsymbol{\theta}^\top \mathbf{x}_i}}{1+e^{-\boldsymbol{\theta}^\top \mathbf{x}_i}}\right)\\ =& -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \boldsymbol{\theta}^\top \mathbf{x}_i - \log(1+e^{\boldsymbol{\theta}^\top \mathbf{x}_i})\right)\end{aligned}$$

For each term $f = y\boldsymbol{\theta}^\top \mathbf{x} + \log(1 + e^{\boldsymbol{\theta}^\top \mathbf{x}})$ in the sum, since $y\boldsymbol{\theta}^\top \mathbf{x}$ is linear, thus we only need to prove $\log(1 + e^{\boldsymbol{\theta}^\top \mathbf{x}})$ is convex, whose Hessian $\mathbf{H}(\boldsymbol{\theta})$ is

$$\mathbf{H}(\boldsymbol{\theta}) = \frac{\partial^2 \log(1+e^{\boldsymbol{\theta}^\top \mathbf{x}})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top} = \frac{e^{\boldsymbol{\theta}^\top \mathbf{x}}}{(1+e^{\boldsymbol{\theta}^\top \mathbf{x}})^2}\mathbf{x}\mathbf{x}^\top \succeq 0$$

Based on the Lemma that *a function f is convex if and only if $\nabla f \succeq 0$*, we know f is convex.

Based on the Lemma that *the sum of convex functions is convex*, we can show the sum over N is still convex.

For those who are interested in the proof of the lemmas, please refer [1].

Other useful properties of convex functions:

If f and g are convex functions and g is non-decreasing, then $h(x) = g(f(x))$ is convex. As an example, if $f(x)$ is convex, then so is $e^{f(x)}$, because e^x is convex and monotonically increasing.

If f is concave and g is convex and non-increasing, then $h(x) = g(f(x))$ is convex.

1.2 Interpretation

We change notation slightly and assume $y_i \in \{\pm 1\}$. In this case, the logistic regression model is defined by

$$p(y \,|\, \mathbf{x};\, \boldsymbol{\theta}) = \sigma(y\boldsymbol{\theta}^\top \mathbf{x}),$$

where σ is the logistic sigmoid function defined by

$$\sigma(z) = \frac{1}{1+e^{-z}}.$$

The *log odds* of $y = 1$ conditioned on $\mathbf{x}$ is defined as

$$\log \frac{p(+1 \,|\, \mathbf{x};\, \boldsymbol{\theta})}{p(-1 \,|\, \mathbf{x};\, \boldsymbol{\theta})}.$$

(a) Prove that the log odds is equal to the simple expression $\boldsymbol{\theta}^\top \mathbf{x}$.

Answer: This is a simple calculation:

$$\begin{aligned}
\log \frac{p(+1 \,|\, \mathbf{x};\, \boldsymbol{\theta})}{p(-1 \,|\, \mathbf{x};\, \boldsymbol{\theta})} &= \log \frac{\sigma(\theta^\top x)}{\sigma(-\theta^\top x)} \\
&= \log \frac{1+e^{\theta^\top x}}{1+e^{-\theta^\top x}} \\
&= \log \frac{e^{.5\theta^\top x}(e^{-.5\theta^\top x}+e^{.5\theta^\top x})}{e^{-.5\theta^\top x}(e^{.5\theta^\top x}+e^{-.5\theta^\top x})} \\
&= .5\theta^\top x - (.5\theta^\top x) \\
&= \theta^\top x.
\end{aligned}$$

(b) In light of (a), give an interpretation for each $\boldsymbol{\theta}_i$. For example, if we increase the i-th component of $\mathbf{x}$ while holding the others constant, what effect does this have on the log odds.

Answer: θ_i is the additive change in log odds under a unit marginal increase in the i-th component of x.

1.3 Coordinate Descent

If we assume $y_i \in \{-1, 1\}$ $(0 \to -1)$, the negative log-likelihood will be $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \log\left(1 + e^{-y_i \boldsymbol{\theta}^\top \mathbf{x}_i}\right)$, is this still a convex function? Derive the quadratic approximation by Taylor expansion on $\mathcal{L}(\boldsymbol{\theta})$. We also add an l_2 regularization on parameter $\boldsymbol{\theta}$, we have the regularized loss function $\tilde{\mathcal{L}}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \frac{\lambda}{2}\boldsymbol{\theta}^\top \boldsymbol{\theta}$. If we use the coordinate descent to optimize this surrogate, what is the best step-size for the jth dimension of parameter $\boldsymbol{\theta}$? (This method is implemented as `R` package: `glmnet`)

Hint1: Denote $p(\mathbf{x}) = \left(1 + e^{-\boldsymbol{\theta}^\top \mathbf{x}}\right)^{-1}$, thus if $y = 1$, $\left(1 + e^{-y_i \boldsymbol{\theta}^\top \mathbf{x}}\right)^{-1} = p(\mathbf{x})$; if $y = -1$, $\left(1 + e^{-y_i \boldsymbol{\theta}^\top \mathbf{x}}\right)^{-1} = 1 - p(\mathbf{x})$. We have

$$\frac{\partial^2 \log\left(1 + e^{-y_i \boldsymbol{\theta}^\top \mathbf{x}}\right)}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top} = p(\mathbf{x}_i)(1 - p(\mathbf{x}_i))\mathbf{x}\mathbf{x}^\top$$

which contains no y. This also proves that $\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log\left(1 + e^{-y_i \boldsymbol{\theta}^\top \mathbf{x}_i}\right)$ is convex.

Hint2: Since the quadratic approximation with l_2 regularization is still a quadratic function, it is easy to derive the **closed-form** of best step-size.

Answer

(b, c) If we do not use Taylor expansion, we have the linear search for coordinate descent as

$$\frac{\partial \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\boldsymbol{\theta} + \Delta\theta_j \mathbf{e}_j) + \lambda/2 \|\boldsymbol{\theta} + \Delta\theta_j \mathbf{e}_j\|^2}{\partial \Delta\theta_j} = -\frac{1}{N}\sum_{i=1}^{N} \frac{e^{-y_i(\boldsymbol{\theta} + \Delta\theta_j \mathbf{e}_j)^\top \mathbf{x}_i}}{1 + e^{-y_i(\boldsymbol{\theta} + \Delta\theta_j \mathbf{e}_j)^\top \mathbf{x}_i}} y_i x_{ij} + \lambda(\theta_j + \Delta\theta_j) = 0$$

Thus, no closed form for $\Delta\theta_j$. If we assume $\frac{e^{-y_i(\boldsymbol{\theta}+\Delta\theta_j\mathbf{e}_j)^\top\mathbf{x}_i}}{1+e^{-y_i(\boldsymbol{\theta}+\Delta\theta_j\mathbf{e}_j)^\top\mathbf{x}_i}} \approx \frac{e^{-y_i\boldsymbol{\theta}^\top\mathbf{x}_i}}{1+e^{-y_i\boldsymbol{\theta}^\top\mathbf{x}_i}}$, $\Delta\lambda_j \approx -\theta_j + \frac{1}{N\lambda}\left(\sum_{y_i=1}(1-p(\mathbf{x}_i)) - \sum_{y_i=-1} p(\mathbf{x}_i)\right)$ is a bad approximation.

If we used Taylor approximation,

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}+\Delta\boldsymbol{\theta}) \approx& \mathcal{L}(\boldsymbol{\theta}) + \nabla\mathcal{L}(\boldsymbol{\theta})^\top\Delta\boldsymbol{\theta} + \frac{1}{2}\Delta\boldsymbol{\theta}^\top\nabla^2\mathcal{L}(\boldsymbol{\theta})\Delta\boldsymbol{\theta} \\
=& \frac{1}{N}\sum_{y_i=1}\left(\frac{1}{2}p(\mathbf{x}_i)(1-p(\mathbf{x}_i))\Delta\boldsymbol{\theta}^\top\mathbf{x}_i\mathbf{x}_i^\top\Delta\boldsymbol{\theta} - (1-p(\mathbf{x}_i))\mathbf{x}_i^\top\Delta\boldsymbol{\theta}\right) \\
&+\frac{1}{N}\sum_{y_i=-1}\left(\frac{1}{2}p(\mathbf{x}_i)(1-p(\mathbf{x}_i))\Delta\boldsymbol{\theta}^\top\mathbf{x}_i\mathbf{x}_i^\top\Delta\boldsymbol{\theta} - p(\mathbf{x}_i)\mathbf{x}_i^\top\Delta\boldsymbol{\theta}\right) \\
&+C(\boldsymbol{\theta})
\end{aligned}$$

where $C(\boldsymbol{\theta})$ is some constant independent of $\Delta\boldsymbol{\theta}$ which can be disregarded safely, and define

$$\begin{aligned}
w_i =& p(\mathbf{x}_i)(1-p(\mathbf{x}_i)) \\
z_i =& \frac{(y_i+1)/2 - p(\mathbf{x}_i)}{w_i}
\end{aligned}$$

We have

$$\mathcal{L}(\boldsymbol{\theta}+\Delta\boldsymbol{\theta}) \approx \frac{1}{2N}\sum_i w_i\left(z_i - \Delta\boldsymbol{\theta}^\top\mathbf{x}_i\right)^2 + \tilde{C}(\boldsymbol{\theta})$$

Thus, we are supposed to optimize $\arg\min_{\Delta\boldsymbol{\theta}} \frac{1}{2N}\sum_i w_i\left(z_i - \Delta\boldsymbol{\theta}^\top\mathbf{x}_i\right)^2 + \frac{\lambda}{2}\|\boldsymbol{\theta}+\Delta\boldsymbol{\theta}\|^2$ by coordinate descent, i.e., setting $\Delta\boldsymbol{\theta} = (0, ..., \Delta\theta_j, ..., 0)^\top$. Thus, the optimization is reduced to

$$\arg\min_{\Delta\theta_j} \frac{1}{2N}\sum_i w_i\left(z_i - \Delta\theta_j\cdot x_{ij}\right)^2 + \frac{\lambda}{2}(\theta_j+\Delta\theta_j)^2$$

It is easy to compute the **closed-form** of the optimal $\Delta\theta_j$ by taking derivative on $\Delta\theta_j$ and setting the derivative as 0.

$$\Delta\theta_j^* = \frac{-\lambda\theta_j + \frac{1}{N}\sum_i w_i z_i x_{ij}}{\lambda + \frac{1}{N}\sum_i w_i x_{ij}^2}$$

(a) Based on the Lemma that *a function f is convex if and only if $\nabla f \succeq 0$*, we know f is convex, we can show the ℓ_2 regulized loss is convex. See (d) for the advantage.

(d) Define $\theta_k = k\theta_*$. Observe that

$$\begin{aligned}
\mathcal{L}(\theta_k) &= \frac{1}{N}\sum_{i=1}^N \log\left(1+e^{-y_i k\theta_*^\top x_i}\right) \\
&\le \frac{1}{N}\sum_{i=1}^N \log\left(1+e^{-k\delta}\right) \\
&= \log(1+e^{-k\delta}).
\end{aligned}$$

But

$$\lim_{k\to\infty}\log(1+e^{-k\delta}) = \log(1+0) = 0.$$

Hence, we have

$$0 \leq \lim_{k\to\infty} \mathcal{L}(\theta_k) \leq \lim_{k\to\infty} \log(1+e^{-k\delta}) = 0.$$

The objective is minimized, however the learned parameters $\boldsymbol{\theta}^*$ can take very large values *e.g.,* ∞. This is bad, because $\boldsymbol{\theta}^*$ is sensitive to input $\boldsymbol{x}$, and can easily make overfit decisions on the testing dataset.

This also answers question (a), where the benefit of ℓ_2 regularization forces the learned parameters $\boldsymbol{\theta}^*$ close to $\mathbf{0}$.

2 Practice and Gradient Descent

2.1 Gradient Descent

In practice, we often use *gradient descent* (or related variants) to learn the parameters in logistic regression: your optimization trajectory moves in the direction of the negative gradient at each step, which is the direction of steepest local descent. Is gradient descent going to perform better than coordinate descent when the same step is chosen?

Answer

Generally speaking, we would expect an improvement. But there are counterexamples where gradient descent is worse.

2.2 Large N Scenario

In the scenario of "big data", *i.e., N* is very large, running gradient descent can be very slow. What is the main reason? Could you suggest some solutions to speed up? Hint: *stochastic gradient descent.*

Answer

The regularized loss function is defined as

$$\mathcal{L}(\boldsymbol{\theta}) \triangleq \frac{1}{N}\sum_{i=1}^{N} \log\left(1+e^{-y_i\boldsymbol{\theta}^\top \mathbf{x}_i}\right) + \frac{\lambda}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta}, \tag{1}$$

where the dataset is $\mathbf{D}_n \triangleq (\boldsymbol{x}_n, y_n)$ with input $\boldsymbol{x}_n$ and output y_n.

For large N, stochastic approximations are often employed:

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}) \triangleq \frac{N}{M}\sum_{m=1}^{M} \log\left(1+e^{-y_i\boldsymbol{\theta}^\top \mathbf{x}_i}\right) + \frac{\lambda}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta}, \tag{2}$$

where $\mathcal{S}_m = \{i_1, \cdots, i_M\}$ is a *random* subset of the set $\{1, 2, \cdots, N\}$, with $M \ll N$. The gradient on this mini-batch is denoted as $\tilde{\boldsymbol{f}} = \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$, which is an unbiased estimate of the true gradient. The evaluation of (2) is cheap even when N is large, *i.e.,* the cost of gradient evaluation is reduced from $\mathcal{O}(N)$ to $\mathcal{O}(M)$, thus allowing for many more parameter updates per unit time.

3 KKT

3.1 Inequality Constraint

Let $\alpha \in \mathbb{R}$ and $a \in \mathbb{R}^n$ with $a \neq 0$. Define the halfspace $H = \{x \in \mathbb{R}^n : a^T x + \alpha \geq 0\}$. Consider the problem of finding the point in H with the smallest Euclidean norm.

(a) Formulate this problem as a constrained optimization problem.

(b) Solve the problem with the help of the KKT conditions. (Hint: you should consider different cases based on if α is negative or nonnegative.)

Answer

(a) This is just the optimization problem

$$\min_{x \in \mathbb{R}^n} x^T x \quad \text{such that } a^T x + \alpha \geq 0.$$

(b) We first start by writing the KKT conditions:

$$\mathcal{L}(x, \lambda) = x^T x - \lambda(a^T x + \alpha),$$

and

$$\nabla_x \mathcal{L}(x, \lambda) = 2x - \lambda a.$$

Hence, the KKT conditions are

$$\begin{aligned} 2x - \lambda a &= 0 \\ a^T x + \alpha &\geq 0 \\ \lambda &\geq 0 \\ \lambda(a^T x + \alpha) &= 0. \end{aligned}$$

First note that if $\alpha \geq 0$, then the 0 vector already satisfies the constraint, and clearly it has minimal norm, so in this case, the optimal solution is 0.

Now assume that $\alpha < 0$. Note that the final KKT condition we wrote above implies that $\lambda = 0$ or $a^T x + \alpha = 0$. If $\lambda = 0$, then the very first KKT condition shows that $0 = 2x - 0 = 2x$, so $x = 0$. But this violates the constraint $a^T x + \alpha \geq 0$, since $a^T 0 + \alpha = \alpha < 0$. Hence, we know that $\lambda \neq 0$. Then $a^T x = -\alpha$. On the other hand, from the first KKT condition, we have that $x = \frac{\lambda}{2} a$, and so by substitution, we have

$$-\alpha = \frac{\lambda}{2} a^T a,$$

which means that

$$\lambda = -\frac{2\alpha}{a^T a}.$$

Plugging this back into the first KKT condition gives

$$x = -\frac{\alpha}{a^T a} a.$$

This is the only point that satisfies the KKT conditions, and from there it is easy to verify that it is the solution.

3.2 Linear Programming

A *linear program* is an optimization problem that has a linear objective function and linear constraints. Myriad important problems can be formulated as a linear program, so it is important to be able to solve such problems efficiently. We can pose any linear program in the following standard form:

$$\min_{x \in \mathbb{R}^n} c^T x, \qquad \text{subject to } Ax = b,\ x \geq 0,$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and A is an $m \times n$ matrix. Here, $x \geq 0$ means that each component of the vector x must be nonnegative. Prove that the KKT conditions for this problem are

$$\begin{aligned} A^T\lambda + \mu &= c \\ Ax &= b \\ x &\geq 0 \\ \mu &\geq 0 \\ \mu_i x_i &= 0, \ i = 1, 2, \ldots, n. \end{aligned}$$

An important class of optimization algorithms known as *primal-dual interior point methods* seek to solve the linear program by directly finding solutions to the KKT conditions.

Answer

This is straightforward. First, form the Lagrangian:

$$\mathcal{L}(x, \mu, \lambda) = c^T x - \mu^T x - \lambda^T (Ax - b).$$

Next, compute the gradient

$$\nabla_x \mathcal{L}(x, \mu, \lambda) = c - \mu - A^T \lambda.$$

Setting this equal to zero and reading off the other KKT conditions gives the answer.

References

[1] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.