COS 324, Precept #6:
A Closer Look at Loss Functions

October 24, 2017

1 Overview

In class, in the context of regression, we noted that there is a whole menagerie of loss functions to pick from. In this precept, we consider the different use cases of these losses and some computational concerns. With the material we've covered in class so far, we can start looking at fundamental problems used by statisticians, and propose efficient algorithms.

Some warm-up thoughts:

- Suppose you're trying to find a global minimum b of a convex function, by querying local gradients. Which function is easier to minimize: $\ell(z) = |z-b|$, or $\ell(z) = (z-b)^2$?
- How much should a predictor $\hat{f}$ be penalized for being wrong? Linearly ($|\hat{f}(z) - f(z)|$) or quadratically ($(\hat{f}(z) - f(z))^2$)?

2 Review of linear regression

Recall the problem of *linear regression* (using notation from the slides), where we would like to find an unknown linear relationship that maps every independent variable[1] $\mathbf{x} \in \mathbb{R}^n$ to a dependent variable[2] $y \in \mathbb{R}$.

The algorithmic problem at hand: we are given a training set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$, and we would like to find some vector $\mathbf{w}$ for which $y_i \approx \langle \mathbf{w}, \mathbf{x} \rangle$, "as closely as possible".[3] We pick a loss function $\ell(\cdot)$ (a penalty for erroneous predictions), and write down the problem of finding a loss:

$$\min_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^{m} \ell(\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i).$$

[1] a.k.a. feature vector, regressor, controlled variable, explanatory variable, input, etc.

[2] a.k.a. response, output, explained variable, regressand, etc.

[3] Side note: incorporating bias $y = \langle \mathbf{w}, \mathbf{x} \rangle + b$ requires no more work. Just add a "dummy" feature to $\mathbf{x}$ whose value is always 1. Then, the corresponding "dummy" coefficient in $\mathbf{w}$ will be the bias.

For short, let $f_i(\mathbf{w})$ denote the summand, and $f(\mathbf{w})$ the entire sum, so we have

$$f(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} f_i(\mathbf{w}).$$

If ℓ is convex, then so is each f_i, and thus f. We are now armed with two techniques to solve this optimization problem:

- Gradient descent: repeatedly move $\mathbf{w}$ in the direction of $-\nabla f(\mathbf{w})$.
- Stochastic gradient descent: repeatedly move $\mathbf{w}$ in the direction of $-\nabla f_i(\mathbf{w})$, for a randomly sampled i each time.

We analyzed both in class (as special cases of online gradient descent), showing that both algorithms converge! This means that for *any* convex loss function, we now have an algorithm to find a $\mathbf{w}$ arbitrarily close to the minimizer of f.

3 p-norms and losses they induce

Let us consider two widely-used loss functions. For convenience, let $\mathbf{z} \in \mathbb{R}^m$ denote the *residual vector*, whose entries are the prediction errors $z_i = \langle \mathbf{w}, \mathbf{x}_i \rangle - y_i$.

- $\ell(z) = \frac{1}{2}z^2$ (ℓ_2 loss, a.k.a. squared error), so that $f(\mathbf{w}) = \frac{1}{2m}\|\mathbf{z}\|_2^2$, the Euclidean distance between the prediction and the truth. In this case, we can write $f(\mathbf{w}) = \frac{1}{2m}\|X\mathbf{w} - \mathbf{y}\|^2$, where $X \in \mathbb{R}^{m\times n}$ is the matrix whose rows are $\mathbf{x}_i$, and $\mathbf{y} \in \mathbb{R}^m$ is the vector of y_i's.[4]
- $\ell(z) = |z|$ (ℓ_1 loss, a.k.a. absolute error), so that $f(\mathbf{w}) = \frac{1}{m}\|\mathbf{z}\|_1$. This replaces the Euclidean distance with the "Manhattan" distance: to get from $(42, 7)$ to $(34, 5)$, any shortest path takes $8 + 2$ blocks.

In short, ℓ_2 loss penalizes error quadratically, while ℓ_1 penalizes linearly. Let us address the warm-up questions now:

- The least-squares penalty is smooth and strongly convex, as long as X^TX is well-conditioned (equivalently, the regressors are not too correlated with each other). When this is true, GD/SGD exhibit exponentially fast convergence. This resolves the computational question: least-squares is generally easier. On a quadratic function, when you're far away from the minimum, the gradient is informative about the location of the minimum.

[4]This should look familiar: it's the same form as the potential we defined two precepts ago, to solve a system of linear equations! This is no surprise: you can view as linear regression as trying to solve an overdetermined system of linear equations, with your choice of penalty on only approximately satisfying a linear constraint.

- The question of when to use which loss is more subtle; it is one of the central questions of in data science. If you know that your data is clean (no horrendous outliers), ℓ_2 regression requires fewer samples to stabilize. On the other hand, ℓ_1 loss is more resistant to outliers.

- (Skip if short on time.) The argument about outliers can be made formal. Suppose you knew that your data were corrupted by noise, i.e. $y_i = \langle \mathbf{w}, \mathbf{x}_i \rangle + \xi_i$, with some independent random variables ξ_i. As it turns out, if the noise is Gaussian (each $\xi_i \sim \mathcal{N}(0, \sigma^2)$), the least-squares fit $\mathbf{w}$ gives the *maximum-likelihood estimator*. The ℓ_1 fit is the MLE for Laplace noise, which produce more outliers than Gaussian noise.

A natural question (and excuse to bring up some more math) is whether it makes sense to penalize other powers. We can define a whole family of losses $\ell(z) = |z|^p$, so that

$$f(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{d} |z_i|^p = \frac{1}{m} \|\mathbf{z}\|_p^p.$$

Here, $\|\mathbf{z}\|_p$ is the p-norm of the residual $\mathbf{z}$.e Some notes (in decreasing order of relevance, so some may not be discussed in precept):

- For any $1 \leq p < \infty$, $\|\mathbf{z}\|_p^p$ is a convex function in $\mathbf{z}$. So, for any such choice, we can optimize for $\mathbf{w}$ efficiently with GD or SGD. A solution $\mathbf{w}$ is called an p-norm estimator.

- As $p \to \infty$, it can be shown that $\|\mathbf{z}\|_p$ approaches $\max_i |z_i|$, the largest absolute deviation. Thus, we define $\|\mathbf{z}\|_\infty = \max_i |z_i|$, which is also convex. This is not to be confused with $\|\mathbf{z}\|_1$, the *sum* of absolute deviations.

- As p increases, outliers are penalized more harshly. When $p = \infty$, the empirical loss is completely determined by the *worst* outlier!

- For $0 < p < 1$, $\|\mathbf{z}\|_p$ is well-defined, but non-convex. By an abuse of notation, we define $\|\mathbf{z}\|_0$ to be the number of nonzero entries in $\mathbf{z}$; this is *very* non-convex.

- When $p = 1$ or ∞, $f(\mathbf{w})$ is not only convex, but is piecewise *linear*. Although this insight doesn't help gradient-based methods, it does lend additional structure. Specialized methods exist for these cases, based on linear programming. There is also a rich theory on accelerating algorithms for $p = 2$; some of this work is happening right now at Princeton.

4 Some more exotic losses

Here are some other loss functions that you might encounter in the wild:

- Huber loss: a hybrid between ℓ_1 and ℓ_2, obtained by stitching together a quadratic around the origin (for stability) and linear tails outside some threshold (for outlier tolerance).

- ε-insensitive loss: $\ell(z) = \max(|z| - \varepsilon, 0)$. Use this when you want to assign *zero* penalty within some tolerance ε. Of course, you can modify any loss in this way to get an insensitive version.
- Symmetrized exponential loss: $\ell(z) = e^z + e^{-z}$. Penalize outliers *exponentially*, which is more aggressively than $|z|^p$ for any constant p.
- Relatedly: $\ell(z) = e^{z-\delta} + e^{-z-\delta}$. Similar to above, but creates a basin of insensitivity around the origin.
- Smoothed ℓ_1 loss: $\ell(z) = \log(1 + e^z) + \log(1 + e^{-z})$. Similar to Huber.