1 Optimization Methods

Beginning of Aug. 25, 2022

1.1 Loss function and risk function

Suppose we are given a **loss function** $\ell(f(x), y)$. A frequent example is the squared loss for $y = \mathbb{R}$ defined by $\ell(f(x), y) = (f(x) - y)^2$.

A big question of interest in ML is *what* to minimize this function over? For example, if we were to minimize loss over some distribution D over all (x, y), we want to minimize the **risk function** (risk of prediction $f(x)$ defined by

$$R(f) := \mathbb{E}_{(x,y)\sim D}[\ell f(x), y] = \sum_{(\tilde{x},\tilde{y})} \mathbb{P}((x, y) = (\tilde{x}, \tilde{y}))\ell(f(\tilde{x}), \tilde{y}).$$

Challenges we naturally encounter:

(1) i.i.d. assumption. We assume that we have a set of labelled instances drawn identically and independently from a distribution D but often this assumption is too idealized.

(2) Theoretical abstraction – often useful.

Minimizing Risks

Definition: Empirical Risk

Given a set of labelled data points $S = \{(x_i, y_i)\}_{i=1}^n$, we define the **empirical risk** of any predictor f with respect to S to be

$$\hat{R}_S(f) := \frac{1}{n}\sum_{i=1}^n \ell(f(x_i), y_i).$$

Note that this risk function coincides with the more general one under the discrete uniform distribution.

Definition: Empirical Risk Minimizer (ERM)

Given a function class (i.e., a collection of functions) $\mathcal{F} : \{f : \mathcal{X} \to \mathcal{Y}\}$ and a set S of labelled data points, we define the **empirical risk minimizer** to be

$$\min_{f\in\mathcal{F}} \hat{R}_S(f) = \min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^n \ell(f(x_i), y_i).$$

Generalizing Risks

What we *really* want to do is to generalize the results to beyond data points we already have. Given a function f, a data set S, the following is a trivial tautology:

$$R(f) = \underbrace{\hat{R}_S(f)}_{\text{emp. risk}} + \underbrace{(R(f) - \hat{R}_S(f))}_{\text{generalization gap}}.$$

That is, to minimize $R(f)$, it suffices to minimize both the empirical risk $\hat{R}_S(f)$ and the remaining term $R(f) - \hat{R}_S(f)$, known as the **generalization gap**, a quantifier of how well our prediction generalizes to unseen examples.

Measuring Generalization: Training / Test Paradigm

In theory, we derive *generalization bounds* based on complexity of the model to obtain upper bounds for the generalization gap. In practice we conduct **empirical evaluation** — we divide the data into two parts, the **training set**, a subset of data points on which we train our model, and a **test set**, another subset on which we test the model. Ideally, we only use test set once or a few times. Our major concern is that our algorithm does well on training set only because it has memorized everything rather than actually doing prediction. A good algorithm, on the other hand, should have a small generalization gap.

Supervised Learning in a Nutshell

(1) Loss function: what is the right loss function for the task?

(2) Representation: what class of functions should we use?

- Inductive bias: *no model can do well on every task. "All models are wrong, but some are useful."*

(3) Optimization: how can we efficiently find the ERM?

(4) Generalization: will the predictions of our model transfer gracefully to unseen examples?

- Note we can have trivial models that have zero generalization gap by outputting a constant, but such model violates the optimization criteria in almost all cases.

1.2 Formal Setup of Linear Regression

- Input: $x \in \mathbb{R}^d$.
- Output: $y \in \mathbb{R}$.
- Training data: $S = \{(x_i, y_i)\}_{i=1}^n$.
- Linear model: $f : \mathbb{R}^d \to \mathbb{R}$ with $f(x) = w_0 + w \cdot x$ where $w := (w_1, \dots, w_d) \in \mathbb{R}^d$ is the **weight factor**. For convenience, we define $\tilde{x} := (1, x) \in \mathbb{R}^{d+1}$ and $\tilde{w} := (w_0, w_1, \dots, w_d)$. By doing so, $f(x)$ can be re-written as $f(x) := \tilde{w}^T \tilde{x}$.

Our goal is to minimize total squared error,

$$\hat{R}_S(\tilde{w}) = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)^2 = \frac{1}{n} \sum_{i=1}^{n} (\tilde{x}_i^T \tilde{w} - y_i)^2.$$

We define the **residual sum of squares**, RSS, to be

$$\text{RSS}(\tilde{w}) := n\hat{R}_S(\tilde{w}) = \sum_{i=1}^{n} (\tilde{x}_i^T \tilde{w} - y_i)^2.$$

Note that under such notation, ERM is identical to finding

$$\tilde{w}^* := \underset{\tilde{w} \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \text{RSS}(\tilde{w}),$$

and the solution is known as the **least squares solution**.

Example: $d = 0$. Let $d = 0$ so we are trying to find the best constant function $f(x) = w_0$ that predicts a set of data. In this case,

$$\text{RSS}(w_0) = \sum_{i=1}^{n}(w_0 - y_i)^2 = nw_0^2 - 2w_0\sum_{i=1}^{n} y_i + C = n\Big(w_0 - \frac{1}{n}\sum_{i=1}^{n} y_i\Big)^2 + \tilde{C}$$

where $C, \tilde{C}$ are constants of little interest. It follows that w_0^* is simply the mean of the y_i's.

Example: $d = 1$. Now let us consider $d = 1$ so that

$$\text{RSS}(\tilde{w}) = \sum_{i=1}^{n}(w_0 + w_1 x_i - y_i)^2. \tag{*}$$

Taking gradient gives

$$\frac{\partial}{\partial w_0}\text{RSS}(\tilde{w}) \propto \sum_{i=1}^{n}(w_0 + w_1 x_i - y_i)$$

and

$$\frac{\partial}{\partial w_1}\text{RSS}(\tilde{w}) \propto \sum_{i=1}^{n} x_i(w_0 + w_1 x_i - y_i).$$

Setting both to 0, we obtain a linear system

$$\begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix}\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix},$$

whose solution would be (assuming the 2×2 matrix is invertible)

$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix}^{-1}\begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix}.$$

Since (*) is a convex function of both arguments, a stationary point is guaranteed to be a (global) minimum.

Example: General Case. Now we generalize to $\mathbb{R}^d$. Here,

$$\text{RSS}(\tilde{w}) = \sum_{i=1}^{n}(\tilde{x}_i^T\tilde{w} - y_i)^2.$$

Setting $\nabla\text{RSS}(\tilde{w})$ to $0 \in \mathbb{R}^d$, we have

$$\begin{aligned}\nabla\text{RSS}(\tilde{w}) &= 2\sum_{i=1}^{n}\tilde{x}_i(\tilde{x}_i^T\tilde{w} - y_i) \propto \tilde{w}\sum_{i=1}^{n}(\tilde{x}_i^T\tilde{x}_i) - \sum_{i=1}^{n}\tilde{x}_i y_i \\ &= (\tilde{x}^T\tilde{x})\tilde{w} - \tilde{x}^T y.\end{aligned}$$

The general solution is (assuming $\tilde{x}^T\tilde{x}$ is invertible) $\tilde{w}^* = (\tilde{x}^T\tilde{x})^{-1}\tilde{x}^T y$.
In particular, suppose that $\tilde{x}^T\tilde{x} = I$ so that $\tilde{w}^* = \tilde{x}^T y$.

Alternate approach:

$$\begin{aligned}\mathrm{RSS}(\tilde{w}) &= \sum_{i=1}^{n}(\tilde{w}^T\tilde{x}_i - y_i)^2 = \|\tilde{x}\tilde{w} - y\|_2^2 \\ &= (\tilde{x}\tilde{w} - y)^T(\tilde{x}\tilde{w} - y) \\ &= \tilde{w}^T\tilde{x}^T\tilde{x}\tilde{w} - y^T\tilde{x}\tilde{w} - \tilde{w}^T\tilde{x}^T y + C \\ [\text{completion of squares}] &= (\tilde{w} - (\tilde{x}^T\tilde{x})^{-1}\tilde{x}^T y)^T(\tilde{x}^T\tilde{x})(\tilde{x} - (\tilde{x}^T\tilde{x})^{-1}\tilde{x}^T y) + C.\end{aligned}$$

It remains to notice that $u^T(\tilde{x}^T\tilde{x})u = \|\tilde{x}u\|_2^2 \geqslant 0$ and $= 0$ iff $u = 0$. Hence the minimizer of RSS takes place when $\tilde{w}^* = (\tilde{x}^T\tilde{x})^{-1}\tilde{x}^T y$.

1.3 Gradient Descent

The bottleneck of computing

$$\tilde{w}^* = (\tilde{x}^T\tilde{x})^{-1}\tilde{x}^T y$$

is to invert the matrix $\tilde{x}^T\tilde{x} \in \mathbb{R}^{(d+1)^2}$ which takes $\mathcal{O}(d^3)$ time (faster algorithms exist in theory but may not be practical).

Problem: minimize a function $F(w)$.

Gradient descent: start with some $w^{(0)}$; for $t \in \{0, 1, \ldots, T\}$, define $w^{(t+1)} := w^{(t)} - \eta\nabla F(w^{(t)})$ where η is called the step size / learning rate.

Example. Let $w = (w_1, w_2) \in \mathbb{R}^2$ and define

$$F(w) := 0.5(w_1^2 - w_2)^2 + 0.5(w_1 - 1)^2.$$

The gradient is

$$\frac{\partial F}{\partial w_1} = 2(w_1^2 - w_2)w_1 + w_1 - 1 \qquad \text{and} \qquad \frac{\partial F}{\partial w_2} = -(w_1^2 - w_2).$$

For GD, we initialize $w^{(0)} = (w_1^{(0)}, w_2^{(0)})$ (maybe $(0, 0)$ or randomly) and $t = 0$. We set η as well. Then we iteratively set

$$\begin{aligned} w_1^{(t+1)} &\leftarrow w_1^{(t)} - \eta[2((w_1^{(t)})^2 - w_2^{(t)})w_1^{(t)} + w_1^{(t)} - 1] \\ w_2^{(t+1)} &\leftarrow w_2^{(t)} - \eta[(w_1^{(t)})^2 - w_2^{(t)}] \\ t &\leftarrow t + 1. \end{aligned}$$

We stop either when w converges or when t reached a prescribed number.

Why GD?

Intuition: we consider the first-order Taylor approximation

$$F(w) \approx F(w^{(t)}) + \nabla F(w^{(t)})^T(w - w^{(t)}).$$

Consequently

$$F(w^{(t+1)}) \approx F(w^{(t)}) - \eta\|\nabla F(w^{(t)})\|_2^2 \leqslant F(w^{(t)}),$$

so GD never increases function value, assuming we have a good choice of η (so we don't trave l too far and actually move away from the minima).

Convergence Guarantees for GD

For *convex objectives*, given ϵ there exists a lower bound for t such that $F(w^{(t)}) - F(w^*) < \epsilon$ for large t (so we eventually converge to the minima). Even for non-convex objectives, some guarantees still exist, e.g., how many iterations $t = t(\epsilon)$ are needed to achieve $\|\nabla F(w^{(t)})\| < \epsilon$ and approximate a **stationary point**.

It is known mathematically that a stationary point for a convex objective is a global minimizer.