# Lecture 2: Learning with Gradient Descent

CS4787 — Principles of Large-Scale Machine Learning Systems

**Review: The empirical risk.** Suppose we have a dataset $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x_i \in \mathcal{X}$ is an example and $y_i \in \mathcal{Y}$ is a label. Let $h : \mathcal{X} \to \mathcal{Y}$ be a hypothesized model (mapping from examples to labels) we are trying to evaluate. Let $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be a *loss function* which measures how different two labels are The *empirical risk* is

$$R(h) = \frac{1}{n} \sum_{i=1}^{n} L(h(x_i), y_i).$$

Most notions of error or accuracy in machine learning can be captured with an empirical risk. A simple example: measure the error rate on the dataset with the *0-1 Loss Function*

$$L(\hat{y}, y) = 0 \text{ if } \hat{y} = y \text{ and } 1 \text{ if } \hat{y} \neq y.$$

Other examples of empirical risk?

We need to compute the empirical risk a lot during training, both during validation (and hyperparameter optimization) and testing, so it's nice if we can do it fast.

**Question: how does computing the empirical risk scale?** Three things affect the cost of computation.

- The number of training examples $n$. The cost will certainly be proportional to $n$.
- The cost to compute the loss function $L$.
- The cost to evaluate the hypothesis $h$.

**Review: Empirical Risk Minimization.** We don't just want to compute the empirical risk: we also want to minimize it. To do so, we *parameterize* the hypotheses using some parameters $w \in \mathbb{R}^d$. That is, we assign each hypothesis a $d$-dimensional vector of parameters and vice versa and solve the optimization problem

$$\text{minimize: } R(h_w) = \frac{1}{n} \sum_{i=1}^{n} L(h_w(x_i), y_i) = f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w) \text{ over } w \in \mathbb{R}^d$$

where $h_w$ denotes the hypothesis associated with the parameter vector $w$. This is an instance of a principle of scalable ML: **Write your learning task as an optimization problem, then solve it with an optimization algorithm.**

**Gradient descent (GD).** Initialize the parameters at some value $w_0 \in \mathbb{R}^d$, and decrease the value of the empirical risk iteratively by running

$$w_{t+1} = w_t - \alpha_t \cdot \nabla f(w_t) = w_t - \alpha_t \cdot \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w)$$

where $w_t$ is the value of the parameter vector at time $t$, $\alpha_t$ is a parameter called the *learning rate* or *step size*, and $\nabla f$ denotes the gradient (vector of partial derivatives) of $f$.

**Another iterative method of optimization: Newton's method.** Initialize the parameters at some value $w_0 \in \mathbb{R}^d$, and decrease the value of the empirical risk iteratively by running

$$w_{t+1} = w_t - \left(\nabla^2 f(w_t)\right)^{-1} \nabla f(w_t)$$

where $\nabla^2 f$ denotes the Hessian (matrix of second partial derivatives) of $f$. Newton's method can converge more quickly than gradient descent, because it is a *second-order optimization method* (it uses the second derivative of $f$) whereas gradient descent is only a *first-order method* (it uses the first derivative of $f$).

**Question: what is the computational cost of gradient descent and Newton's method? Not including the training set, how much memory is required?** Suppose that computing gradients of the examples can be done in $O(d)$ time, and express your answer in terms $n$ and $d$.

Gradient descent:

Time

$O(nd)$

Space

$O(d)$

Newton's method:

Time

$O(d^3 + nd^2)$

$O(d^2)$

**Takeaway message:**

**Why does gradient descent work? And what does it mean for it to work?** Intuitively, it decreases the value of the objective at each iteration, as long as the learning rate is small enough and the value of the gradient is nonzero. Eventually, this should result in gradient descent *coming close to a point where the gradient is zero*. We can prove that gradient descent works under the assumption that the second derivative of the objective is bounded.[1] Suppose that for some constant $L > 0$, for all $x$ in the space and for any vector $u \in \mathbb{R}^d$,

$$\left|u^T \nabla^2 f(x) u\right| \leq L \|u\|^2 .$$

Here, $\nabla^2 f(x)$ denotes the matrix of partial derivatives of the function $f$. This is equivalent to the condition that $\left\|\nabla^2 f(x)\right\|_2 \leq L$ (where the norm here denotes the induced norm).

Starting from this condition, let's look at how the objective changes over time as we run gradient descent with a fixed step size. From Taylor's theorem, there exists a $\xi_t$ such that

$$\begin{aligned} f(w_{t+1}) &= f(w_t - \alpha \nabla f(w_t)) \\ &= f(w_t) - \alpha \nabla f(w_t)^T \nabla f(w_t) + \frac{1}{2} \left(\alpha \nabla f(w_t)\right)^T \nabla^2 f(\xi_t) \left(\alpha \nabla f(w_t)\right) \\ &\leq f(w_t) - \alpha \|\nabla f(w_t)\|^2 + \frac{\alpha^2 L}{2} \|\nabla f(w_t)\|^2 = f(w_t) - \alpha \left(1 - \frac{\alpha L}{2}\right) \|\nabla f(w_t)\|^2 . \end{aligned}$$

If we choose our step size $\alpha$ to be *small enough* that $1 \geq \alpha L$, then,

$$f(w_{t+1}) \leq f(w_t) - \frac{1}{2}\alpha \|\nabla f(w_t)\|^2 \qquad \Rightarrow \qquad \frac{1}{2}\alpha \|\nabla f(w_t)\|^2 \leq f(w_t) - f(w_{t+1}).$$

That is, the objective is guaranteed to decrease at each iteration. Now, if we sum this up across $T$ iterations of gradient descent, we get

$$\frac{1}{2}\alpha \sum_{t=0}^{T-1} \|\nabla f(w_t)\|^2 \leq \sum_{t=0}^{T-1} \left(f(w_t) - f(w_{t+1})\right) = f(w_0) - f(w_T) \leq f(w_0) - f^*$$

[1] There are other assumptions we could use here also, but this is the simplest one.

where $f^*$ is the global minimum value of the loss function $f$. From here, we can get

$$\min_{t\in\{0,\ldots,T\}} \|\nabla f(w_t)\|^2 \leq \frac{1}{T}\sum_{t=0}^{T-1} \|\nabla f(w_t)\|^2 \leq \frac{2(f(w_0)-f^*)}{\alpha T}.$$

This means that the smallest gradient we observe after $T$ iterations is getting smaller proportional to $1/T$. So gradient descent converges (as long as we look at the smallest observed gradient).

**Question: does this ensure that gradient descent converges to the global optimum (i.e. the value of $w$ that minimizes $f$ over all $w \in \mathbb{R}^d$)? Why might this cause problems?**

When can we ensure that gradient descent *does* converge to the unique global optimum? Certainly, we can do this when there is only one global optimum. The simplest case of this is the case of **convex objectives**.

**Convex functions.** A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if for all $x, y \in \mathbb{R}^d$, and all $\eta \in [0, 1]$

$$f(\eta x + (1-\eta)y) \leq \eta f(x) + (1-\eta)f(y).$$

What this means graphically is that if we draw a line segment between any two points in the graph of the function, that line segment will lie above the function. There are a bunch of equivalent conditions that work if the function is differentiable. In terms of the gradient, for all $x, y \in \mathbb{R}^d$,

$$(x-y)^T(\nabla f(x) - \nabla f(y)) \geq 0,$$

and in terms of the Hessian, for all $x \in \mathbb{R}^d$ and all $u \in \mathbb{R}^d$

$$u^T \nabla^2 f(x) u \geq 0;$$

this is equivalent to saying that the Hessian is positive semidefinite.

**What are some examples of hypotheses and loss functions that result in a convex objective?**

An even easier case than convexity: **strong convexity**. A function is strongly convex with parameter $\mu > 0$ if for all $x, y \in \mathbb{R}^d$,

$$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\mu}{2}\|x-y\|^2$$

or equivalently, for all $x \in \mathbb{R}^d$ and all $u \in \mathbb{R}^d$

$$u^T \nabla^2 f(x) u \geq \mu \|u\|^2 .$$

One condition that is implied by strong convexity is

$$\|\nabla f(x)\|^2 \geq 2\mu \left(f(x) - f^*\right);$$

this is sometimes called the Polyak-Lojasiewicz condition. A useful note: you can make any convex objective strongly convex by adding $\ell_2$ regularization.

## Gradient descent on strongly convex objectives.

As before, let's look at how the objective changes over time as we run gradient descent with a fixed step size. This is a standard approach when analyzing an iterative algorithm like gradient descent. From our proof before, we had (as long as $1 \geq \alpha L$)

$$f(w_{t+1}) \leq f(w_t) - \frac{1}{2}\alpha \|\nabla f(w_t)\|^2 .$$

Now applying the Polyak-Lojasiewicz condition condition to this gives us

$$f(w_{t+1}) \leq f(w_t) - \alpha\mu \left(f(w_t) - f^*\right).$$

Subtracting the global optimum $f^*$ from both sides produces

$$f(w_{t+1}) - f^* \leq f(w_t) - f^* - \alpha\mu \left(f(w_t) - f^*\right) = (1 - \alpha\mu)\left(f(w_t) - f^*\right).$$

Applying this recursively,

$$f(w_T) - f^* \leq (1 - \alpha\mu)^T \left(f(w_0) - f^*\right) \leq \exp(-\alpha\mu T) \cdot \left(f(w_0) - f^*\right).$$

This shows that, for strongly convex functions, gradient descent with a constant step size converges exponentially quickly to the optimum. This is sometimes called *convergence at a linear rate*. If we use the largest step size that satisfies our earlier assumption that $1 \geq \alpha L$ (i.e. $\alpha = 1/L$), then this rate becomes

$$f(w_T) - f^* \leq \exp\left(-\frac{\mu T}{L}\right) \cdot \left(f(w_0) - f^*\right).$$

Equivalently, in order to ensure that $f(w_T) - f^*$ for some target error $\epsilon > 0$, it suffices to set $T$ large enough that

$$T \geq \frac{L}{\mu} \cdot \log\left(\frac{f(w_0) - f^*}{\epsilon}\right).$$

The main thing that affects how large $T$ needs to be here is the ratio $L/\mu$, which is a function just of the problem itself and not of how we initialize or how accurate we want our solutions to be. We call the ratio

$$\kappa = \frac{L}{\mu}$$

the *condition number of the problem*. The condition number encodes *how hard a strongly convex problem is to solve*. When the condition number is very large, the problem can be very hard to solve, and take many iterations of gradient descent to get close to the optimum. Even if the cost of running gradient descent is tractible as we scale, **if scaling a problem causes the condition number to blow up, this can cause issues**!

**What affects the condition number? What can we do to make the condition number smaller?**