IntroML - Lecture Notes Week 2

Ruben Schenk, ruben.schenk@inf.ethz.ch

March 9, 2022

1 Optimization

1.1 1D-Case

The general idea for **optimization** in 1D is to iteratively try to minimize $L(w)$:

> **General iterative algorithm to minimize $L(w)$**
>
> 1. Start at initial w^{start}
> 2. At each step calculate $w^{next} = w^{now} + \tilde{\eta} \cdot v$, where $\tilde{\eta}$ defines how far we move and v is the update direction depending on L and w^{now}
> 3. When we can't improve much more, stop
> 4. Output $\hat{w} = w^{final}$

At each step of the algorithm above, we have to ask ourselves:

- Which direction v should we choose?
- How far $\tilde{\eta}$ to go in that direction?
- Can we still improve more?

With common sense, if $L(w)$ decreases as w increases (i.e. $L'(w) < 0$), we should increase w and vice versa. In other words, we choose the direction $v = -\text{sign}(L'(w))$.

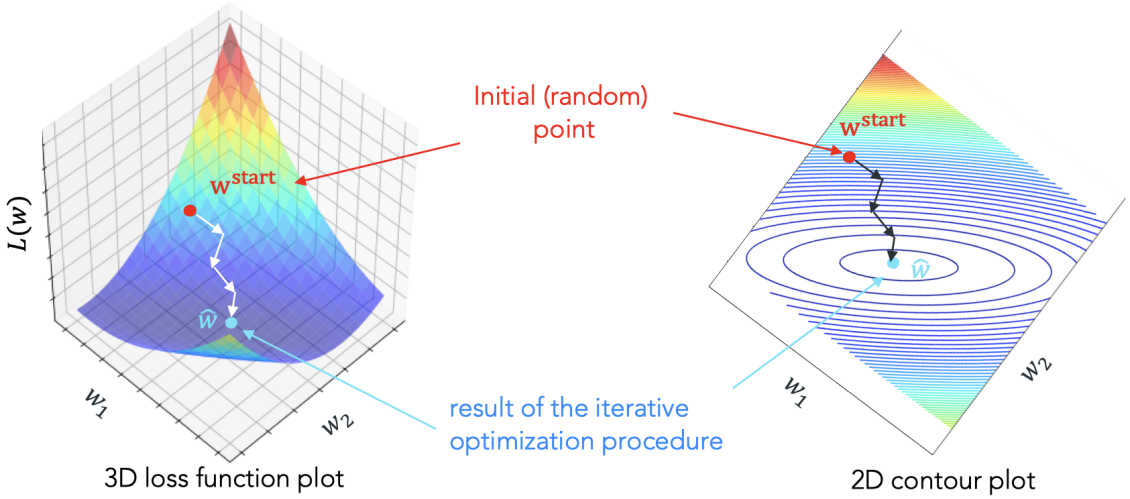
Choosing $\tilde{\eta}$ is harder. Generally speaking, we have to decrease $\tilde{\eta}$ as the slope gets smaller. Even then, η shouldn't be too large because else it can potentially begin to grow again.

Another important question to consider is when do we stop.

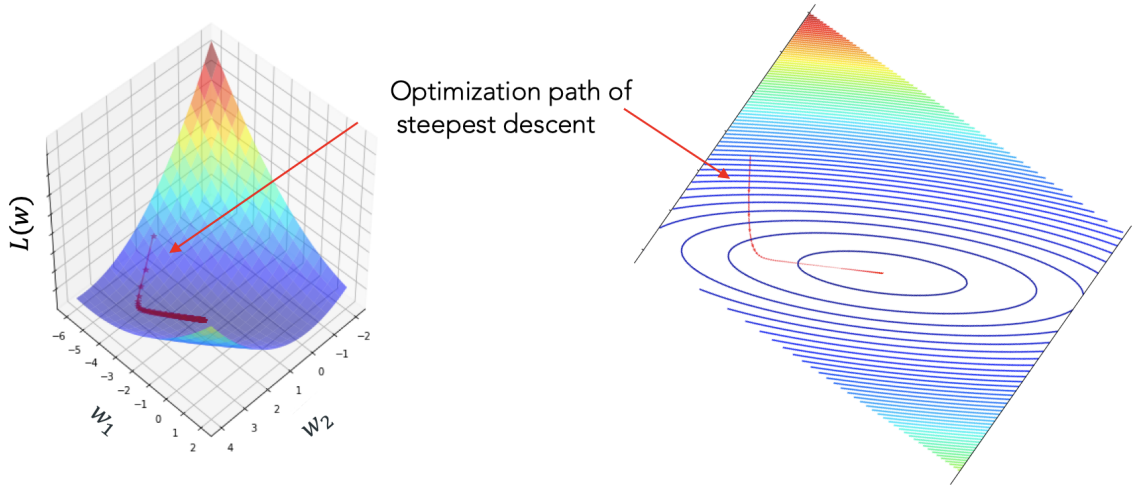
1.2 Multidimensional Case

1.2.1 Visualization of The Loss

Again, visualization beyond $d = 2$ is hard, hence our visualizations are for $w \in \mathbb{R}^2$. Example:



1.2.2 Steepest Descent



Remember, the general update formula is of the form $w^{next} = w^{now} + \tilde{\eta}v$ for some unit direction $||v|| = 1$. What is the **steepest descent direction** v?

If we assume that L is differentiable, then the linear approximation of the loss is given by:

$$L(w^{next}) \simeq L(w^{now}) + \tilde{\eta}\langle \nabla_w L(w^{now}),\, v\rangle$$

The idea is that for small $\tilde{\eta}$, the steepest direction on the tangent plane is similar to the steepest gradient on the true function. With the derivation of the steepest descent direction on the tangent plane we get $v = -\frac{\nabla L(w^{now})}{||\nabla L(w^{now})||}$. This leads to the **gradient descent update:**

$$w^{next} = w^{now} - \eta \nabla_w L(w^{now}), \quad \text{with stepsize } \eta = \frac{\tilde{\eta}}{||\nabla L(w)||}$$

This finally leads to the following algorithm:

Gradient descent algorithm to minimize $L(w)$

1. Start at initial w^0
2. Repeat $w^{t+1} \leftarrow w^t - \eta \nabla_w L(w^t)$
3. Stop when $||w^{t+1} - w^t|| \propto ||\nabla_w L(w^t)|| \leq 10^{-5}$
4. Output $\hat{w} = w^T$

The linear approximation for the loss value at step $t+1$ is given by:

$$L(w^{t+1}) = L(w^t - \eta \nabla_w L(w^t)) \simeq L(w^t) - \eta \langle \nabla_w L(w^t),\, \nabla_w L(w^t)\rangle < L(w^t)$$

The negative gradient direction is a descent direction (for small enough stepsize η)!

1.2.3 Convergence and Stepsize for Linear Regression

Remember that for *linear regression* we have $L(w) = \frac{1}{n}||y - Xw||^2 = \frac{1}{n}[w^T X^T X w - 2w^T X^T y + y^T y]$.
If we use the gradient descent update $w^{t+1} = w^t - \eta \nabla L(w^t)$ and if we write $w_{min} = \text{argmin}_{w \in \mathbb{R}^d} L(w)$. Then indeed, $w^t \to w_{min}$ and $||w^t - w_{min}||_2 \leq \rho^t ||w^0 - w_{min}||_2$, where $\rho = ||I - \eta X^T X||_{op}$.

Using the chain rule, the gradient at any w reads $\nabla L(w) = -X^T(y - Xw)$, and we obtain $w^{t+1} = w^t - \eta \nabla L(w^t) = w^t + \nabla X^T(y - Xw^t)$.

If we remember that the global minimum w_{min} satisfies $X^T X w_{min} = X^T y$, then:

$$w^{t+1} - w_{min} = (I - \eta X^T X)(w^t - w_{min})$$

Furthermore, remember that for any matrix A, $||A||_{op} = \sup_x \frac{||Ax||_2}{||x||_2} = \max\{|\lambda_{min}(A)|, |\lambda_{max}(A)|\}$.

For $A = X^T X$ this is just the largest eigenvalue. Furthermore, by definition, $||Ax||_2 \leq ||A||_{op}||x||_2$ and hence:

$$||w^{t+1} - w^*||_2 \leq ||I - \eta X^T X||_{op}||w^t - w_{min}||_2 \leq ||I - \eta X^T X||_{op}^{t+1}||w^0 - w_{min}||_2$$

If $\rho < 1 \Leftrightarrow \lambda_{min}(X^T X) > 0$, the error goes to 0 as t goes to ∞. We say that the **gradient descent converges linearly / exponentially.**

1.2.4 Speeding Up Gradient Descent

To *speed up the gradient descent,* we want large steps in flat areas, and small steps in high curvature ones. This leads to the momentum/accelerated method, where we combine previous direction with the negative gradient direction:

$$w^{t+1} - w^t = \alpha(w^t - w^{t-1}) - \eta \nabla_w L(w^t)$$

1.2.5 Stochastic Gradient Method (SGD)

Remember that the training loss definition for parameterized functions is $L(w) = \frac{1}{n}\sum_{i=1}^n l(y_i, f_w(x_i))$. We have already seen the following algorithm:

Gradient Descent Algorithm

1. Start at initial w^0
2. Until $w^{t+1} - w^t$ is sufficiently small, repeat:
3. (a) Update $w^{t+1} \leftarrow w^t - \eta \nabla_w L(w^t)$
4. Output $\hat{w} = w^T$

We see in line 3 of the algorithm above that we need to compute the gradients at all points. This is very costly! The memory required is $O(nd)$ and the time $O(n \times$ cost to compute $\nabla l)$. We propose another algorithm:

Stochastic Gradient Descent (SGD) Algorithm

1. Start at initial w^0
2. Until $w^{t+1} - w^t$ is sufficiently small, repeat:
3. (a) Update $w^{t+1} \leftarrow w^t - \eta \nabla_w L_S(w^t)$
4. Output $\hat{w} = w^T$

Here, in line 3, instead of computing the gradient for all points, we only consider a random subset of points $S \subset [1, ..., n]$ (so-called *minibatch SGD*). If S is just one random point, then it's called **SGD.**

1.3 Stationary Points and Minima

Remember that the gradient descent converges to a stationary point, but we want to get to the minimum! For differentiable losses:

- w is local minimum $\rightarrow$ w is a stationary point
- w is global minimum $\rightarrow$ w is a local minimum
- $L(w) < L(v) \, \forall v \in \mathbb{R}^d \rightarrow w$ is a global minimum

Mathematically, **convexity** is the function property that guarantees a local minimum to be a global minimum. We have three different conditions:

- 0-th order condition (iff.): $L(\lambda w + (1-\lambda)v) \leq \lambda L(w) + (1-\lambda)L(v)$
- 1st order condition (iff.): $L(v) \geq L(w) + \nabla L(w)^T (v - w)$
- 2nd order condition (iff.): Hessian $\nabla^2 L(w) \geq 0$

Furthermore, we can define a **strongly convex** functions. We say that $L(w)$ is strongly convex if $L(w) - \frac{m}{2}||w||^2$ is convex. We have the following conditions:

- 1st order condition (iff. for some $m > 0$): $L(v) \geq L(w) + \nabla L(w)^T (v - w) + \frac{m}{2}||v - w||^2$
- 2nd order condition (iff. for some $m > 0$): Hessian $\nabla^2 L(w) \geq mI$

1.4 Effect of Data on Global Minimum Prediction

Before we can determine how data affects the model quality, we need to formalize what a good model is:

- Goal: Intuition for how the sample size n can change the prediction performance. What is a good prediction?
- Goal standard: Assume the "true" average price is some *ground truth linear function* $f^*(x) = x^T w^*$
- Good prediction model $\hat{f}(x)$: should be close to $f^*(x)$, i.e. have a low error.

When we say that the prediction model should have a *low error,* we mean that:

$$l(\hat{f}(x),\, f^*(x)) = (\hat{f}(x) - f^*(x))^2 = ((\hat{w} - w^*)^T x)^2,$$

should be low. We often measure:

- Average error $\mathbb{E}(\hat{f}(x) - f^*(x))^2$ over all x, or
- for linear functions $||\hat{w} - w^*||_2$ since $((\hat{w} - w^*)^T x)^2 \leq (||\hat{w} - w^*||_2 ||x||_2)^2$

We often model the observed data as $y_i = f^*(x_i) + \epsilon_i = \langle w^*,\, x_i \rangle + \epsilon_i$ and call ϵ_i **noise.**

1.5 Different Dimensionalities

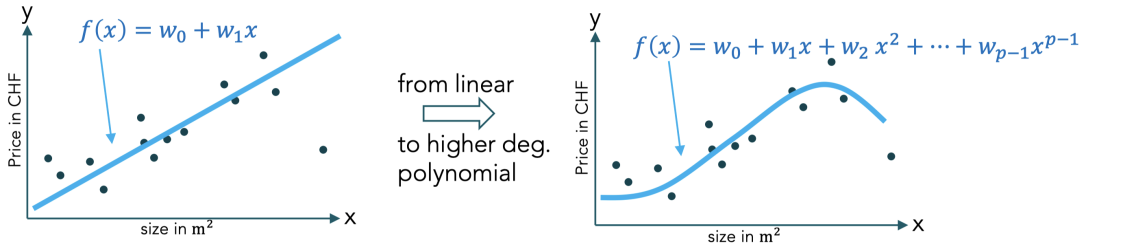
Let us assume $n = 2$ points and w_0 to be unknown. There are infinitely many w that satisfy $y = \mathbf{1}w_0 + Xw$, i.e. that interpolates the training data. If we now assume $n = 3$ points, for $d = 2$ there is now only one interpolating solution that satisfies $y = \mathbf{1}w_0 + Xw$.

In general: *the dimensionality of interpolating solutions decreases.*

Again, assuming $n > d$, we still have infinitely many w interpolating the training data. Which solution does the gradient descent actually find? If initialized at 0, the gradient descent converges to the minimum norm solution $\hat{w}_{GD} = \text{argmin}_w ||w||_2$ such that $y = \mathbf{1}w_0 + Xw$ that interpolates the data.

1.6 Going Beyond Linear Functions

Instead of linear functions, we might also consider **polynomial functions.** Both can be written as $f(x) = \sum_{j=1}^{p} w_j \phi_j(x)$ with feature vector $\phi(x) = (\phi_1(x), ..., \phi_p(x)) \in \mathbb{R}^p$:



Linear function with $\phi_1(x) = 1, \phi_2(x) = x$ uses additional monomials $\phi_3(x) = x^2 \ldots \phi_p(x) = x^{p-1}$

For the loss function we again consider the squared loss. We seek the minimizer of the square loss

$$\hat{f} = \text{argmin}_{f \in F_\phi} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2,$$

this time in some fixed feature function space F_ϕ. This is equivalent to minimizing over vector w:

$$\hat{w} = \text{argmin}_{w \in \mathbb{R}^p} L(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} w_j \phi_j(x))^2$$

We can rewrite the training loss L in matrix vector notation as follows:

$$L(w) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - w^\top \phi_j(x_i)\right)^2 = \frac{1}{n} ||y - \Phi w||^2$$