1 Outline

In this lecture, we study

- stochastic optimization,
- stochastic gradient descent,
- the perceptron algorithm,
- the support vector machine,
- logistic regression.

2 Stochastic Optimization

Stochastic optimization (SO) is an optimization problem of the following form.

$$\min_{x\in\mathcal{X}} \quad f(x)=\mathbb{E}_{\xi\sim\mathbb{P}}\left[h(x,\xi)\right]$$

where

- ξ is a random parameter vector whose underlying distribution is given by $\mathbb{P}$,
- $\mathcal{X}$ is the feasible set for the decision vector x.

If $h(x,\xi)$ is convex with respect to x for any fixed ξ, then

$$f(x)=\mathbb{E}_{\xi\sim\mathbb{P}}\left[h(x,\xi)\right]$$

is convex. When the distribution $\mathbb{P}$ has finite support $\{\xi_1,\dots,\xi_n\}$ with $\mathbb{P}[\xi=\xi_i]=1/n$ for $i\in[n]$, then we have

$$f(x)=\frac{1}{n}\sum_{i=1}^{n}h(x,\xi_i).$$

For linear regression, we have a set of n data points $(a_1,b_1),\dots,(a_n,b_n)$ where a_i is a feature vector and b_i is its associated response for $i\in[n]$. Then the empirical distribution over the data set is given by $\mathbb{P}[(a,b)=(a_i,b_i)]=1/n$. A linear model with coefficient vector x predicts the response of feature vector a would be $x^\top a$. Then we can consider the squared loss of the model with respect to a data point (a,b) as

$$h(x,(a,b))=(b-x^\top a)^2.$$

As (a,b) is distributed with $\mathbb{P}$, the expected loss is given by

$$\min_{x} \quad \mathbb{E}_{(a,b)\sim\mathbb{P}}\left[h(x,(a,b))\right] \quad = \quad \min_{x} \quad \frac{1}{n}\sum_{i=1}^{n}(b_i-x^\top a_i)^2.$$

3 Stochastic Gradient Descent

When we have $f(x) = \mathbb{E}_{\xi\sim\mathbb{P}}[h(x,\xi)]$, the gradient of f is given by $\nabla f(x) = \mathbb{E}_{\xi\sim\mathbb{P}}[\nabla h(x,\xi)]$, computing which requires knowledge of the underlying distribution $\mathbb{P}$. Instead of computing the exact gradient, one would consider a more efficient way of considering a stochastic estimate of $\nabla f(x)$. For example, one may obtain a sample ξ from distribution $\mathbb{P}$, in which case $\nabla h(x,\xi)$ is an unbiased estimator of the gradient $\nabla f(x)$. Here, the descent method with a stochastic estimate of the gradient is referred to as **stochastic gradient descent (SGD)**. Given a solution x_t, we obtain a stochastic estimate g_t of the gradient $\nabla f(x_t)$ and apply

$$x_{t+1} = \text{proj}_{\mathcal{X}}(x_t - \eta_t g_t)$$

for some step size $\eta_t > 0$.

Algorithm 1 Stochastic Gradient Descent

Initialize $x_1 \in \mathcal{X}$.
for $t = 1, \ldots, T$ **do**
  Obtain an estimator g_t of $\nabla f(x_t)$.
  Update $x_{t+1} = \text{proj}_{\mathcal{X}}\{x_t - \eta_t g_t\}$ for a step size $\eta_t > 0$.
end for
Return $(1/T)\sum_{t=2}^{T+1} x_t$.

In this section, we consider the special case where $\mathcal{X} = \mathbb{R}^d$ (the unconstrained case) and the objective function f is a smooth convex function.

Theorem 7.1. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function that is β-smooth in the ℓ_2-norm. Let $x_1, \ldots, x_{T+1}$ be the iterates generated by Algorithm 1. Assume that for each $t \geq 1$,*

$$\mathbb{E}[g_t \mid x_t] = \nabla f(x_t) \quad \textit{and} \quad \mathbb{E}\left[\|g_t - \nabla f(x_t)\|_2^2 \mid x_t\right] \leq \sigma^2.$$

Then setting step size $\eta_t = \min\left\{1/\beta,\ 1/\sqrt{T}\right\}$ for $t \geq 1$, we deduce

$$\mathbb{E}\left[f\left(\frac{1}{T}\sum_{t=2}^{T+1} x_t\right)\right] - f(x^*) \leq \frac{\beta\|x_1 - x^*\|_2^2}{2T} + \frac{2\sigma^2 + \|x_1 - x^*\|_2^2}{2\sqrt{T}}$$

where $x^* \in \textit{argmin}_{x\in\mathbb{R}^d} f(x)$. Setting step size $\eta_t = \min\left\{1/\beta,\ \|x_1 - x^*\|_2/\sigma\sqrt{2T}\right\}$ for $t \geq 1$,

$$\mathbb{E}\left[f\left(\frac{1}{T}\sum_{t=2}^{T+1} x_t\right)\right] - f(x^*) \leq \frac{\beta\|x_1 - x^*\|_2^2}{2T} + \frac{\sigma\|x_1 - x^*\|_2\sqrt{2}}{\sqrt{T}}.$$

Proof. Since $\mathbb{E}[g_t \mid x_t] = \nabla f(x_t)$ and $\mathbb{E}\left[\|g_t - \nabla f(x_t)\|_2^2 \mid x_t\right] \leq \sigma^2$, it follows that

$$\mathbb{E}\left[\|g_t\|_2^2 \mid x_t\right] = \mathbb{E}\left[\|g_t - \nabla f(x_t)\|_2^2 \mid x_t\right] + \|\nabla f(x_t)\|_2^2 \leq \|\nabla f(x_t)\|_2^2 + \sigma^2.$$

Let η be such that $\eta_t = \eta$ for $t \geq 1$. Next, consider

$$\begin{aligned}
\mathbb{E}\left[f(x_{t+1}) - f(x_t) \mid x_t\right] &\leq \mathbb{E}\left[\nabla f(x_t)^\top (x_{t+1} - x_t) + \frac{\beta}{2}\|x_{t+1} - x_t\|_2^2 \mid x_t\right] \\
&= \mathbb{E}\left[-\eta \nabla f(x_t)^\top g_t + \frac{\beta\eta^2}{2}\|g_t\|_2^2 \mid x_t\right] \\
&\leq -\eta\|\nabla f(x_t)\|_2^2 + \frac{\beta\eta^2}{2}\|\|\nabla f(x_t)\|_2^2 + \frac{\beta\eta^2}{2}\sigma^2 \\
&\leq -\frac{\eta}{2}\|\nabla f(x_t)\|_2^2 + \frac{\eta}{2}\sigma^2
\end{aligned} \tag{7.1}$$

where the first inequality follows from smoothness of f, the second inequality holds because $\mathbb{E}\left[g_t \mid x_t\right] = \nabla f(x_t)$ and $\mathbb{E}\left[\|g_t\|_2^2 \mid x_t\right] \leq \|\nabla f(x_t)\|_2^2 + \sigma^2$, and the third inequality is from $\eta \leq 1/\beta$. Moreover, by convexity of f, we deduce the following. Note that

$$\begin{aligned}
\mathbb{E}\left[\|x_{t+1} - x^*\|_2^2 \mid x_t\right] &= \|x_t - x^*\|_2^2 - 2\eta\mathbb{E}\left[g_t^\top (x_t - x^*) \mid x_t\right] + \eta^2\mathbb{E}\left[\|g_t\|_2^2 \mid x_t\right] \\
&= \|x_t - x^*\|_2^2 - 2\eta\nabla f(x_t)^\top (x_t - x^*) + \eta^2\mathbb{E}\left[\|g_t\|_2^2 \mid x_t\right] \\
&\leq \|x_t - x^*\|_2^2 - 2\eta(f(x_t) - f(x^*)) + \eta^2\mathbb{E}\left[\|g_t\|_2^2 \mid x_t\right] \\
&\leq \|x_t - x^*\|_2^2 - 2\eta(f(x_t) - f(x^*)) + \eta^2\|\nabla f(x_t)\|_2^2 + \eta^2\sigma^2
\end{aligned}$$

which implies in turn that

$$f(x_t) - f(x^*) \leq \frac{1}{2\eta}\left(\mathbb{E}\left[\|x_{t+1} - x^*\|_2^2 \mid x_t\right] - \|x_t - x^*\|_2^2\right) + \frac{\eta}{2}\|\nabla f(x_t)\|_2^2 + \frac{\eta}{2}\sigma^2. \tag{7.2}$$

Combining (7.1) and (7.2), it follows that

$$\mathbb{E}\left[f(x_{t+1}) - f(x^*)\right] \leq \frac{\mathbb{E}\left[\|x_{t+1} - x^*\|_2^2\right]}{2\eta} - \frac{\mathbb{E}\left[\|x_t - x^*\|_2^2\right]}{2\eta} + \eta\sigma^2.$$

This implies that

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} f(x_{t+1})\right] - f(x^*) \leq \frac{\|x_1 - x^*\|_2^2}{2\eta T} + \eta\sigma^2.$$

The last step is to show that

$$\frac{\|x_1 - x^*\|_2^2}{2\eta T} + \eta\sigma^2 \leq \frac{\beta\|x_1 - x^*\|_2^2}{2T} + \frac{\sigma^2}{\sqrt{T}},$$

as required. □

4 Perceptron Algorithm

In this section, we consider the **perceptron algorithm** for binary classification. Given n data $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i \in \{-1, 1\}$ are labels, we want to find a separating hyperplane

$$w^\top x = b$$

to classify data with $+1$ and data with -1. Basically, we predict the label of x as $+1$ if $w^\top x \geq b$ and -1 if not. Here, without loss of generality, we may assume that $b = 0$. This is because, we

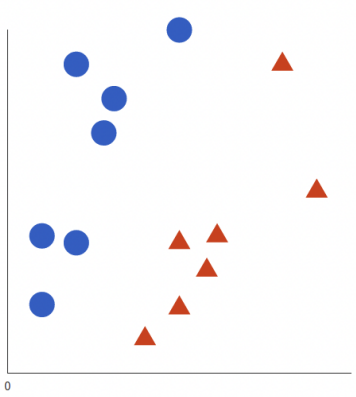

Figure 7.1: Data Points with Two classes

may consider the following augmentation step.

$$x \to \begin{bmatrix} x \\ 1 \end{bmatrix} \quad \text{and} \quad w^\top x = b \to \begin{bmatrix} w \\ -b \end{bmatrix}^\top \begin{bmatrix} x \\ 1 \end{bmatrix} = 0.$$

The **perceptron algorithm** is a fundamental supervised learning algorithm for the binary classification problem. The algorithm works with a **perceptron**, which is the simplest form of neural networks. At each iteration t, we sample a data point (x_t, y_t) at random, and update the weight

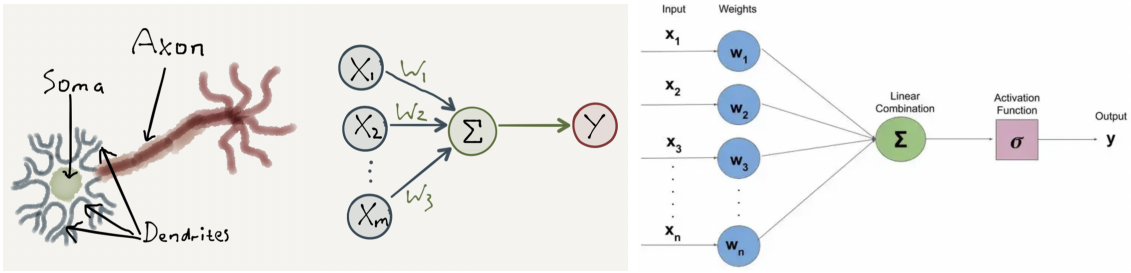

Figure 7.2: Perceptron for Binary Classification

vector w with the following rule. For some $\eta_t > 0$,

$$w_{t+1} = \begin{cases} w_t + \eta_t y_t x_t, & \text{if } y_t(w_t^\top x_t) < 0 \\ w_t, & \text{otherwise.} \end{cases}$$

In fact, the perceptron algorithm can be viewed as the process of training a perceptron. The perceptron is given as follows. Given a data point (x, y), the input vector is given by $-yx$ which, which is obtained by multiplying the feature vector x by the label y. Then the output of the input layer is given by $w^\top(-yx) = -y(w^\top x)$. Lastly, we use the ReLU (Rectified Linear Unit) for the activation function σ, i.e., $\sigma(z) = \max\{z, 0\}$. Then the output of the perceptron is given by

$$h(w, (x, y)) = \sigma(w^\top(-yx)) = \max\left\{-y(w^\top x), 0\right\}.$$

Using f itself as our loss function, training the neural network is done by solving

$$\min_w \quad \frac{1}{n}\sum_{i=1}^{n} \max\left\{-y_i(w^\top x_i), 0\right\}.$$

Then to run SGD, we sample a data point (x_t, y_t) at each iteration t, and the subgradient of $h(w_t, (x_t, y_t))$ is given by

$$g_t = \begin{cases} -y_t x_t, & \text{if } y_t(w_t^\top x_t) < 0, \\ 0, & \text{otherwise.} \end{cases}$$

In this case, SGD with step size η_t at step t would run in the same way as the perceptron algorithm.

We have just explained that the perceptron algorithm is equivalent to training a simple neural network with a specific loss function. Then the next question is, does $h(w, (x, y)) = \max\{-y(w^\top x), 0\}$ give rise to a valid loss function? Note that $h(w, (x, y)) = \max\{-y(w^\top x), 0\} > 0$ only if $w^\top x < 0$ and $y = 1$. Therefore, when (x, y) is correctly classified, i.e., the case $w^\top x \geq 0$ and $y = 1$ or the case $w^\top x < 0$ and $y = 0$, we have $h(w, (x, y)) = \max\{-y(w^\top x), 0\} = 0$.

5 Support Vector Machine

In this section, we provide another algorithm for the binary classification problem. Again, the problem is to find a separating hyperplane

$$w^\top x = 0$$

to classify data with $+1$ and data with -1. Basically, we predict the label of x as $+1$ if $w^\top x \geq 0$ and -1 if not.

The goal is to find a separating hyperplane $w^\top x = 0$ such that the distance between two consecutive hyperplanes $w^\top x = 0$ and $w^\top x = 1$ is maximized. Here, the distance is given by $1/\|w\|_2$. Then the problem can be formulated as

$$\begin{aligned} \text{minimize} \quad & \|w\|_2 \\ \text{subject to} \quad & y_i(w^\top x_i) \geq 0, \ i = 1, \ldots, n. \end{aligned}$$

If this problem is feasible, then $x \to \text{sign}(w^\top x)$ is a valid classifier for the data set.

What if the data set is not entirely separable? What if no hyperplane separates the data without an error? In such cases, we force separation via a penalty term, instead of imposing hard constraints. The number of misclassifications can be used as penalty. Namely,

$$\sum_{i=1}^{n} 1(y_i \neq \text{sign}(w^\top x_i)).$$

However, this is not convex. Instead, we apply the **hinge loss**, which is an upper bound on the number of misclassifications, given by

$$\sum_{i=1}^{n} \max\{1 - y_i(w^\top x_i), \ 0\}.$$

Then we solve

$$\min_{w} \quad \lambda\|w\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} \max\{1 - y_i(w^\top x_i), \ 0\}$$

where λ determines the trade-off between the margin size and the penalty. To run SGD, we may consider

$$h(w, (x, y)) = \max\Big\{1 - y(w^\top x), \ 0\Big\} + \lambda\|w\|_2^2.$$

Then a subgradient of $h(w,(x,y))$ is given by

$$g = \begin{cases} 2\lambda w - yx, & \text{if } y(w^\top x) < 1, \\ 2\lambda w, & \text{otherwise.} \end{cases}$$

Based on this, SGD works as follows. At each iteration t, given the current weight vector w_t, we sample a data point (x_t, y_t) based on which we deduce the next point with

$$w_{t+1} = \begin{cases} (1 - 2\eta_t\lambda)w_t + \eta_t y_t x_t, & \text{if } y(w^\top x) < 1, \\ (1 - 2\eta_t\lambda)w_t, & \text{otherwise.} \end{cases}$$

This algorithm is referred to as the support vector machine.

6 Logistic Regression

In this section, we consider a different approach for the binary classification problem. The data points $(x_1, y_1), \ldots, (x_n, y_n)$ have labels $y_i \in \{0, 1\}$. Instead of a linear classifier as in the perceptron algorithm and the support vector machine, we model the conditional probability distribution $\mathbb{P}(y \mid x)$ using a logistic function as follows. We assume that for some weight vector w,

$$\mathbb{P}(y \mid x) = g_w(x) = \frac{1}{1 + e^{-w^\top x}}.$$

If $w^\top x \to \infty$, then we have $\mathbb{P}(y = 1 \mid x) = 1$. $w^\top x \to -\infty$, then we have $\mathbb{P}(y = 1 \mid x) = 0$. Given the n data points $(x_1, y_1), \ldots, (x_n, y_n)$, the likelihood is given by

$$\prod_{i=1}^{n} g_w(x_i)^{y_i} \cdot (1 - g_w(x_i))^{1-y_i},$$

and the log-likelihood is given by

$$\begin{aligned} \log\left(\prod_{i=1}^{n} g_w(x_i)^{y_i} \cdot (1 - g_w(x_i))^{1-y_i}\right) &= \sum_{i=1}^{n} \left(y_i \log(g_w(x_i) + (1 - y_i)\log(1 - g_w(x_i))\right) \\ &= \sum_{i=1}^{n} \left(y_i(w^\top x_i) + \log\frac{e^{-w^\top x_i}}{1 + e^{-w^\top x_i}}\right). \end{aligned}$$

Then w maximizing the likelihood can be computed by

$$\operatorname*{argmax}_w \left\{\sum_{i=1}^{n} \left(y_i(w^\top x_i) + \log\frac{e^{-w^\top x_i}}{1 + e^{-w^\top x_i}}\right)\right\} = -\operatorname*{argmin}_w \left\{-\sum_{i=1}^{n} \left(y_i(w^\top x_i) + \log\frac{e^{-w^\top x_i}}{1 + e^{-w^\top x_i}}\right)\right\}.$$

Therefore, it is equivalent to consider the loss function

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} h(w, (x_i, y_i))$$

where

$$h(w,(x,y)) = -y(w^\top x) - \log\frac{e^{-w^\top x}}{1 + e^{-w^\top x}}.$$

Note that

$$\nabla_w h(w,(x,y)) = -yx + x + \nabla_w \log\left(1 + e^{-w^\top x}\right) = \left(\frac{1}{1+e^{-w^\top x}} - y\right) x = (g_w(x) - y)\, x.$$

Then SGD minimizing the loss function f would proceed with

$$w_{t+1} = w_t - \eta_t (g_{w_t}(x_t) - y_t) x_t$$

where w_t is the current coefficient vector and (x_t, y_t) is the data point obtained for iteration t.