Week 2 Note

2.1 Gradient descent(GD)

2.1.1 Introduction

- The gradient vector at a point x points in the direction of greatest increase of the function f: each element of the gredient shows how fast $f(x)$ is changing

> - Example:
>
> $$f(\vec{x}) = f(x_1, x_2) = 6x_1^2 + 4x_2^2 - 4x_1x_2$$
>
> - The gradient vector is
>
> $$\triangledown f(x_1, x_2) = \begin{pmatrix} 12x_1 - 4x_2 \\ 8x_2 - 4x_1 \end{pmatrix}$$

2.1.2 Optimisation

- Optimisation algorithm
  1. Start with a point w(initial guess)
  2. Find a direction d to move on
  3. Determine how far (η) to move along d
  4. Update: $w = w + \eta d$

2.1.3 Minimisation

- it is an iterative algorithm, starting from $\vec{w}^{(0)}$ and producing a new $\vec{w}^{(t+1)}$ at each iteration as:

$$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta_t \triangledown C(\vec{w}^{(t)})$$

where $t = 0, 1, ..., T$

> $\eta_t > 0$ is the learning rate or step size

2.1.4 Choosing a step size

- Choosing a step size
  - If step size is too large - algorithm may never converge
  - If step size is too small - convergence may be very slow

2.1.5 GD for least squares regression

- Least squares regression
  - For least square regression, let's recall:

$$C(\vec{w}) = \frac{1}{2n}(\underbrace{\vec{w}^T X^T X \vec{w}}_{quadratic} - \underbrace{2\vec{w}^T X^T y}_{linear} + \underbrace{\vec{y}^T \vec{y}}_{constant})$$

  - The gredient is computed as:

$$\nabla C(\vec{w}) = \frac{1}{n}(X^T X \vec{w} - X^T \vec{y})$$

  - GD updates $\vec{w}^{(t)}$ by

$$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta \nabla C(\vec{w}^{(t)})$$

$$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \frac{\eta}{n}(X^T X \vec{w}^{(t)} - X^T \vec{y})$$

2.2 Stochastic gradient descent(SGD)

2.2.1 Introduction

- Replace the computationally expensive term $\nabla C(\vec{w}^{(t)})$ by a stochastic gradient computed on a random example

2.2.2 Algorithm

- Alogirithm
  1. Intialise the weights $\vec{w}^{(0)}$
  2. For $t = 0, 1, ..., T$
  - Draw i_t from $1, 2, ..., n$ with equal probability
  - Compute stochastic gradient $\nabla C_i(\vec{w}^{(0)})$ and update

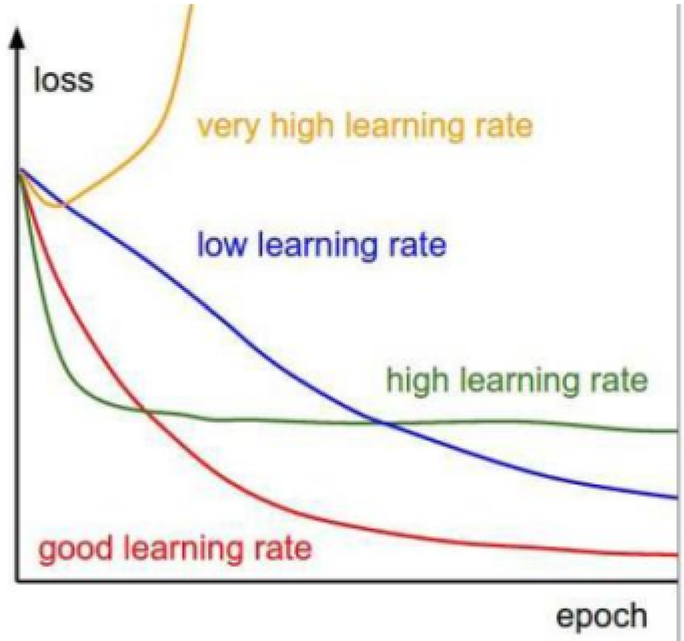
$$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta_t \nabla C_i(\vec{w}^{(t)})$$

2.2.3 SGD vs GD

- GD requires more computations per iteration but makes a good progress per iteration
  - It needs few iterations to get a good solution
- SGD requires less computations per iteration but makes less update per iteration
  - Therefore, it needs more iterations to get a good solution
- GD and SGD cannot always dominate the other.
  - If we want high accuracy and n is small, then **GD** is better
  - If we want moderate accuracy and n is large, then **SGD** is better

2.2.4 Effect of learning rates

- If we choose a low learning rate, then SGD would converge very slowly
- If we choose a large learning rate, then SGD would not go further as we run more and more iterations
- If we choose a huge learning rate, then SGD would become unstable
- A typical choice is $\eta_t = \frac{c}{\sqrt{t}}$, where c is a parameter needed to tune



2.3 Minibatch SGD

2.3.1 Introduction

- Randomly select a batch of indices: $B_t \subseteq \{1, 2, ..., n\}$ and update the model

$$\vec{w}^{(t)} = \vec{w}^{(t)} - \frac{\eta_t}{b} \sum_{i \in B_t} \nabla C_i(\vec{w}^{(t)})$$

  where b is the batch size

  > If $b = 1$, it is cler that minibatch SGD is SGD

2.3.2 Algorithm

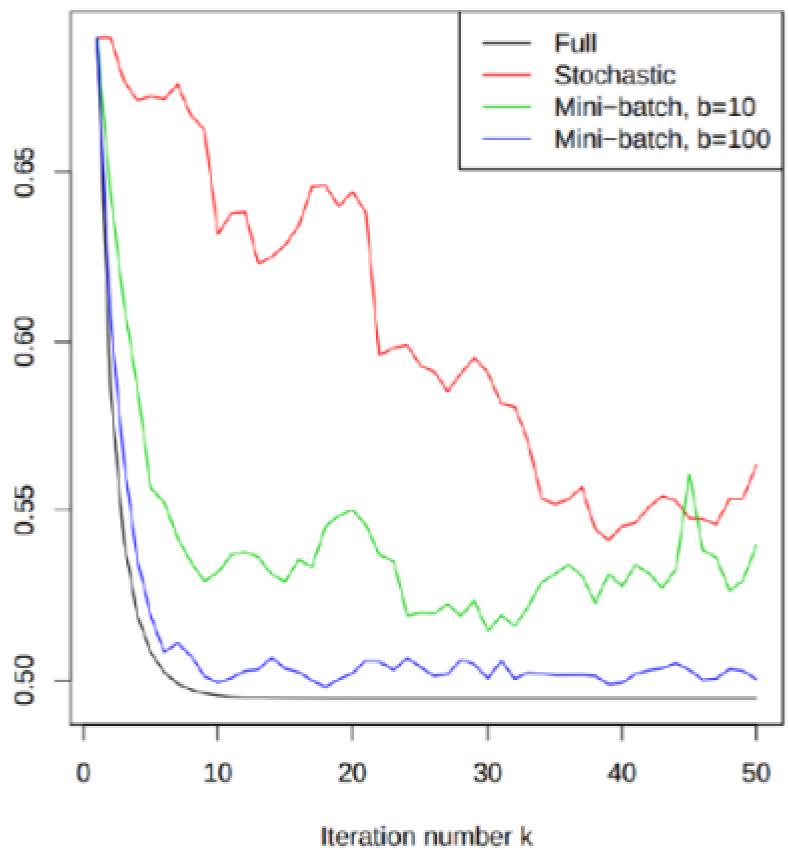
- Let $\{\eta_t\}$ be a sequence of step sizes
- Algorithm
  1. Initialise the weights $\vec{w}^{(0)}$
  2. For $t = 0, 1, ..., T$
  - Randomly select a batch $B_t \subseteq \{1, 2, ..., n\}$ of size b
  - Compute stochastic gradient $\nabla C_i(\vec{w}^{(t)})$ with $i \in B_t$ and update

$$\vec{w}^{(t)} = \vec{w}^{(t)} - \frac{\eta_t}{b}\sum_{i\in B_t}\nabla C_i(\vec{w}^{(t)})$$

2.3.3 minibatch selection

- There are two ways to sample the minibatch B_t
  - sampling with replacement
  - sampling without replacement

2.3.4 Minibatch SGD vs SGD vs GD



2.4 Linear classification

2.4.1 Introduction

- Suppose we have

$$D = \{(\vec{x}^1, y^1), (\vec{x}^2, y^2), \ldots, (\vec{x}^n, y^n)\}$$

and

$$y^i \in \{-1, +1\}$$

- To build a linear model to separate posite examples from negative examples

2.4.2 0-1 loss

$$L(\hat{y}, y) = II[\hat{y} \neq y] = \begin{cases} 1 & if\ \hat{y} \neq y \\ 0 & otherwise \end{cases}$$

- The behaviour of a model on D can be measured by:

$$C(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} II[sgn(\vec{w}^T \vec{x}^i) \neq y^i]$$

> The surrogate of $C(\vec{w})$ which is easy to minimise

2.4.3 margin-based loss

- Margin
  - The margin of a model $\vec{w}$ on an example$(\vec{x}, y)$is defined as $y\vec{w}^T\vec{x}$
- A model with a `positive` margin means a `correct` prediction
- A model with a `negative` margin means an `incorrect` prediction

> $\hat{y} = sgn(\vec{w}^T x)$
> $\bar{y} = \vec{w}^T x$

> Margin(边距)，正确分类的情况下，距离决策边界越远的数据预测的越准确
>
> This further motivateds a model with large margin: a large margin means the model is robust in making a correct prediction

- Loss function of the form:

$$L(\hat{y}, y) = g(y\hat{y})$$

  where g is decreasing

- minimising L means maximising the margin

  - Maximising the margin means getting a model with good performance

2.4.4 Surrogate loss

- We mainly consider

$$g(t) = \frac{1}{2}(max0, 1 - t)^2 = \begin{cases} \theta & if\ t \geq 1 \\ \frac{1}{2}(1 - t)^2 & otherwise \end{cases}$$

- The loss function becomes

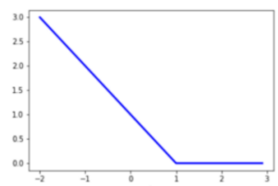
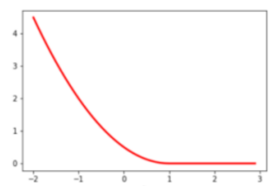
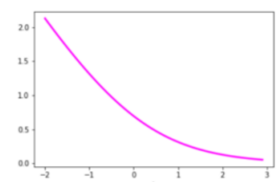
$$L(\hat{y}, y) = \frac{1}{2}(max\{0, 1 - y\hat{y}\})^2 = \frac{1}{2}(max\{0, 1 - y\vec{w}^T\vec{x}\})^2$$

- The behaviour on D is quantified by

$$C(\vec{w}) = \frac{1}{2n}\sum_{i=1}^{n}(max\{0, 1 - y^i\vec{w}^T\vec{x}\})^2$$

- We further get

$$\nabla C_i(\vec{w}) = \begin{cases} \theta & if\ y^i\vec{w}^T\vec{x}^i \geq 1 \\ (\vec{w}^T\vec{x}^i - y^i)\vec{x}^i & otherwise \end{cases}$$

> - There are some other choices, including:
>   - $g(t) = max\{0, 1 - t\}$
>   - $\min\limits_{\vec{w}} \frac{1}{n}\sum_{i=1}^{n}(max\{0, 1 - y^i\vec{w}^T\vec{x}^i\})$
>   - $g(t) = \log(1 + \exp(-t))$
>   - $\min\limits_{\vec{w}} \frac{1}{n}\sum_{i=1}^{n}\log(1 + \exp(-y^i\vec{w}^T\vec{x}^i))$
>
> 
>
> $g(t) = \max\{0, 1 - t\}$
>
> 
>
> $g(t) = \frac{1}{2}(\max\{0, 1 - t\})^2$
>
> 
>
> $g(t) = \log(1 + \exp(-t))$

2.4.5 SGD for linear classification

- Consider linear classification with the hinge loss

$$C_i(\vec{w}) = \max\{0, 1 - y^i\vec{w}^T\vec{x}^i\}$$

- The formula for SGD update:

$$\vec{w}^{(t+1)} = \begin{cases} \vec{w}^{(0)} & if\ y^{i_t}\vec{w}^{(t)^T}\vec{x}^{i_t} \geq 1 \\ \vec{w}^{(0)} + \eta_t y^{i_t}\vec{x}^{i_t} & otherwise \end{cases}$$

- Consider regularisation in the loss function

$$C_i(\vec{w}) = \frac{1}{2}(\max\{0, 1 - y^i\vec{w}^T\vec{x}^i\})^2 + \frac{1}{2}\lambda||\vec{w}||_2^2$$

- The formula for SGD update:

$$\vec{w}^{(t+1)} = \begin{cases} \vec{w}^{(t)} - \lambda\vec{w}^{(t)} & if\ y^{i_t}\vec{w}^{(t)^T}\vec{x}^{i_t} \geq 1 \\ \vec{w}^{(t)} - \lambda\vec{w}^{(t)} + \eta_t(1 - y^{i_t}\vec{w}^{(t)^T}\vec{x}^{i_t})y^{i_t}\vec{x}^{i_t} & otherwise \end{cases}$$