# Week04 - Part 01

## Before we start, Some announcements:

1 _ Unfortunately, we cannot change the tutorial times.

2 _ WS solutions will be posted weekly (on the Wednesday of the tutorials). This gives you time to work on the WS on your own and test/improve your understanding

3 _ A list of useful documents to review probability, linear algebra, and matrix Calculus is posted on Course page.

4. You MUST write your own solution to each week's WS before checking the solutions and going to tutorials. This is the only way to learn. You do not need to submit your solutions.

# Week 04 - Part 01

**Last week Recap:** Logistic Regression & Gradient descent

- We saw the activation function and log-loss function.
- We saw the Maximum likelihood and Cross-Entropy Minimization interperetation of minimizing log-loss.
- We don't have a closed form Solution for minimizing the log-loss function
- Hence, We used gradient descent to numerically find the minimizer

$$\underline{g}_t = \nabla f(\underline{x}_t)$$

$$\underline{v}_t = -\underline{g}_t$$

$$\underline{x}_{t+1} = \underline{x}_t + \varepsilon_t \underline{v}_t$$

# A Quick Recall Before We Move On

**Recall:** Let $f: \mathbb{R}^n \to \mathbb{R}$ and $h: \mathbb{R} \to \mathbb{R}$. Then

$$\nabla_{\underline{x}}\, h\big(f(\underline{x})\big) = h'\big(f(\underline{x})\big)\ \nabla_{\underline{x}} f(\underline{x})$$

**E.g.:** $\nabla_{\underline{x}} \left(1+\log(\underline{a}^T\underline{x})\right)^2 = 2\left(1+\log(\underline{a}^T\underline{x})\right) \nabla_{\underline{x}}\left(1+\log(\underline{a}^T\underline{x})\right)$

$$= 2\left(1+\log(\underline{a}^T\underline{x})\right) \nabla_{\underline{x}} \log(\underline{a}^T\underline{x}) = 2\left(1+\log(\underline{a}^T\underline{x})\right) \frac{1}{\underline{a}^T\underline{x}} \nabla_{\underline{x}}\, \underline{a}^T\underline{x}$$

$$= \frac{2\left(1+\log(\underline{a}^T\underline{x})\right)}{\underline{a}^T\underline{x}}\ \underline{a}$$

**Homework:** Prove the chain rule.

# Using Gradient Descent for Logistic Regression

- Logistic Regression in_sample Error:

$$E_{in}(\underline{w}) = \frac{1}{N}\sum_{n=1}^{N} e_n(\underline{w}) = \frac{1}{N}\sum_{n=1}^{N} \log\left(1+e^{-y_n \underline{w}^T \underline{x}_n}\right)$$

- $$\nabla_{\underline{w}_t} E_{in}(\underline{w}_t) = \frac{1}{N}\sum_{n=1}^{N} \nabla_{\underline{w}_t} e_n(\underline{w}_t) = \frac{1}{N}\sum_{n=1}^{N} \nabla_{\underline{w}_t} \log\left(1+e^{-y_n \underline{w}_t^T \underline{x}_n}\right)$$

$$= \frac{1}{N}\sum_{n=1}^{N} \frac{1}{1+e^{-y_n \underline{w}_t^T x_n}} \nabla_{\underline{w}_t}\left(1+e^{-y_n \underline{w}_t^T \underline{x}_n}\right)$$

$$= \frac{1}{N}\sum_{n=1}^{N} \frac{1}{1+e^{-y_n \underline{w}_t^T x_n}} e^{-y_n \underline{w}_t^T x_n} \cdot \left(-y_n \underline{x}_n\right)$$

$$= \frac{1}{N}\sum_{n=1}^{N} \frac{-y_n \underline{x}_n}{1+e^{y_n \underline{w}_t^T \underline{x}_n}}$$

- Thus, Gradient Descent would update $\underline{w}_t$ by

$$\underline{g}_t = \nabla E_{in}(\underline{w}_t) = \frac{1}{N}\sum_{n=1}^{N} \nabla e_n(\underline{w}_t)$$

Average per sample error gradient.

$$\underline{v}_t = -\underline{g}_t$$

$$\underline{w}_{t+1} = \underline{w}_t + \varepsilon_t \underline{v}_t$$

- Good News:
  - It can be shown that $\frac{1}{N}\sum_{n=1}^{N} \log(1+e^{-y_n \underline{w}^T \underline{x}_n})$ is convex.
  - Hence, GD converges to the globally minimum point.
- Bad News:
  - To run one iteration of Gradient Descent, we need to find $\nabla_{\underline{w}_t} e_n(\underline{w}_t)$ for all $n \in \{1, 2, \ldots, N\}$ points.
  - This is costly, specially when $N$ is large (big data).
- In practice, we overcome this issue by using

Stochastic gradient descent (SGD)

# Stochastic Gradient Descent (SGD)

**Gradient Descent (GD):** $\underline{w}_{t+1} = \underline{w}_t - \varepsilon_t \nabla_{\underline{w}_t} E_{in}(\underline{w}_t)$

The averge gradient of the points $\frac{1}{N}\sum_{n=1}^{N} \nabla e_n(\underline{w}_t)$

**Stochastic Gradient Descent (SGD):** In each iteration,

- choose uniformly at random a datapoint $(\underline{x}_n, y_n)$
- $g_t = \nabla_{\underline{w}_t} e_n(\underline{w}_t)$ ← use the gradient of the $n_{th}$ sample to update $\underline{x}_t$
- $\underline{V}_t = -\underline{g}_t$
- $\underline{w}_{t+1} = \underline{w}_t + \varepsilon_t \underline{V}_t$

■ But does it work?

- It has been proven that for convex functions and under mild conditions, SGD will find the solution.
- Don't worry about the proof. Don't worry about the conditions.

## Note 1: $\underline{g}_t$ of SGD is an unbiased estimate of $\nabla E_{in}(\underline{w}_t)$

- Let's study the expected update direction derived by SGD in each step.

$$\mathbb{E}[\underline{g}_t] = \mathbb{P}[n=1]\, \nabla_{\underline{w}_t} e_1(\underline{w}_t) + \dots + \mathbb{P}[n=N]\, \nabla_{\underline{w}_t} e_N(\underline{w}_t)$$

$$= \frac{1}{N} \sum_{n=1}^{N} \nabla_{\underline{w}_t} e_n(\underline{w}_t) = \nabla_{\underline{w}_t} E_{in}(\underline{w}_t)$$

True gradient

- hence, by SGD, $\underline{g}_t$ is in fact an "unbiased estimator" of the true gradient $\nabla_{\underline{w}_t} E_{in}(\underline{w}_t)$
  - It is a noisy version of $\nabla E_{in}(\underline{w}_t)$, but it is unbiased because $\mathbb{E}[\underline{g}_t] = \nabla_{\underline{w}_t} E_{in}(\underline{w}_t)$

# Note 2: Full GD Complexity V.S. SGD Complexity

- Full GD ("batch GD"): Per iteration complexity is $O(Nd)$
  - When $N$ is large, it is costly.
- SGD: Per iteration Complexity is $O(d)$
  - Complexity of $N$ iteration of SGD = Complexit of 1 iteration of GD
    - But with SGD, in each iteration, we update based on some noisy estimate of the gradient.
    - Numerically, SGD outperforms full GD. That is because usually there are high redundancies in a large dataset.
      - So, you don't have to use all points to find the right direction.

## Note 3: "mini-batch" GD is full-batch GD combined with SGD

■ mini-batch GD: In each iteration:

- choose $M$ sample $r_1, r_2, \ldots, r_M \in \{1, \ldots, N\}$ uniformly at random
- $\underline{g}_t = \frac{1}{M} \sum_{i=1} \nabla e_{r_i}(\underline{w}_t)$
- $\underline{v}_t = -\underline{g}_t$
- $\underline{w}_{t+1} = \underline{w}_t + \varepsilon_t \underline{v}_t$

■ Benefits:

↳ ~~This~~ $g_t$ is a more reliable estimator of the true gradient.

↳ Per iteration Complexity: $\mathcal{O}(Md)$

↳ In practice, we use multi-core CPU/GPU's. So, we can find the Gradient for $M$ datapoints in parallel and get a better estimator of the update direction.

# PLA is An Extremer Version of Logistic Regression with SGD

- logistic regression with SGD updates:

$$\nabla e_n(\underline{w}_t) = \frac{-y_n \underline{x}_n}{1 + e^{y_n \underline{w}_t^T \underline{x}_n}}$$

and $$\underline{w}_{t+1} = \underline{w}_t + \varepsilon_t \frac{y_n \underline{x}_n}{1 + e^{y_n \underline{w}_t^T \underline{x}_n}}$$

- When we randomly pick some $\underline{x}_n$ which is misclassified
  - $y_n \underline{w}_t^T \underline{x}_n \leq 0 \Longrightarrow 0 < e^{y_n \underline{w}_t^T \underline{x}_n} \leq 1$
  - The SGD update would be $\underline{w}_{t+1} = \underline{w}_t + \frac{\varepsilon_t}{1+e^{y_n \underline{w}_t^T \underline{x}_n}} \cdot (y_n \underline{x}_n)$
  - In extreme case, $e^{y_n \underline{w}_t^T \underline{x}_n} \approx 0$ and the update rule would be

$$\underline{w}_{t+1} \approx \underline{w}_t + \varepsilon_t (y_n \underline{x}_n)$$ (Same as PlA if $\varepsilon_t = 1$.)

■ When we randomly pick some $\underline{x}_n$ which is correctly classified.

- $y_n \underline{w}_t^T \underline{x}_n > 0 \implies 1 < e^{y_n \underline{w}_t^T \underline{x}_n}$
- The SGD update would be $\underline{w}_{t+1} = \underline{w}_t + \dfrac{\varepsilon_t}{1 + e^{y_n \underline{w}_t^T \underline{x}_n}} \cdot (y_n \underline{x}_n)$
- In extrem case, $\dfrac{1}{1+e^{y_n \underline{w}_t^T \underline{x}_n}} \approx 0$ and the update rule would be

$$\underline{w}_{t+1} \approx \underline{w}_t$$

(Same as PLA as it does not update $\underline{w}$ for correctly classified points)