Week 04 - Part 02

So far: We talked about Logistic Regression and linear classification for binary classes.

Today: Multi-class Logistic Regression

Multi-class Logistic Regression

- Label: $y \in \{1, 2, \dots, c\}$
- Hypothesis Set: Let $\Omega = \{\underline{w}_{(1)}, \underline{w}_{(2)}, \dots, \underline{w}_{(c)}\}$ be the weight vectors for c classes.
  - Hypothesize that

$$\mathbb{P}[y_n = i \mid \underline{x}_n] = \frac{e^{\underline{w}^T_{(i)} \underline{x}_n}}{\sum_{j=1}^{c} e^{\underline{w}^T_{(j)} \underline{x}_n}} \quad , \text{ for } i \in \{1, \dots, c\}$$

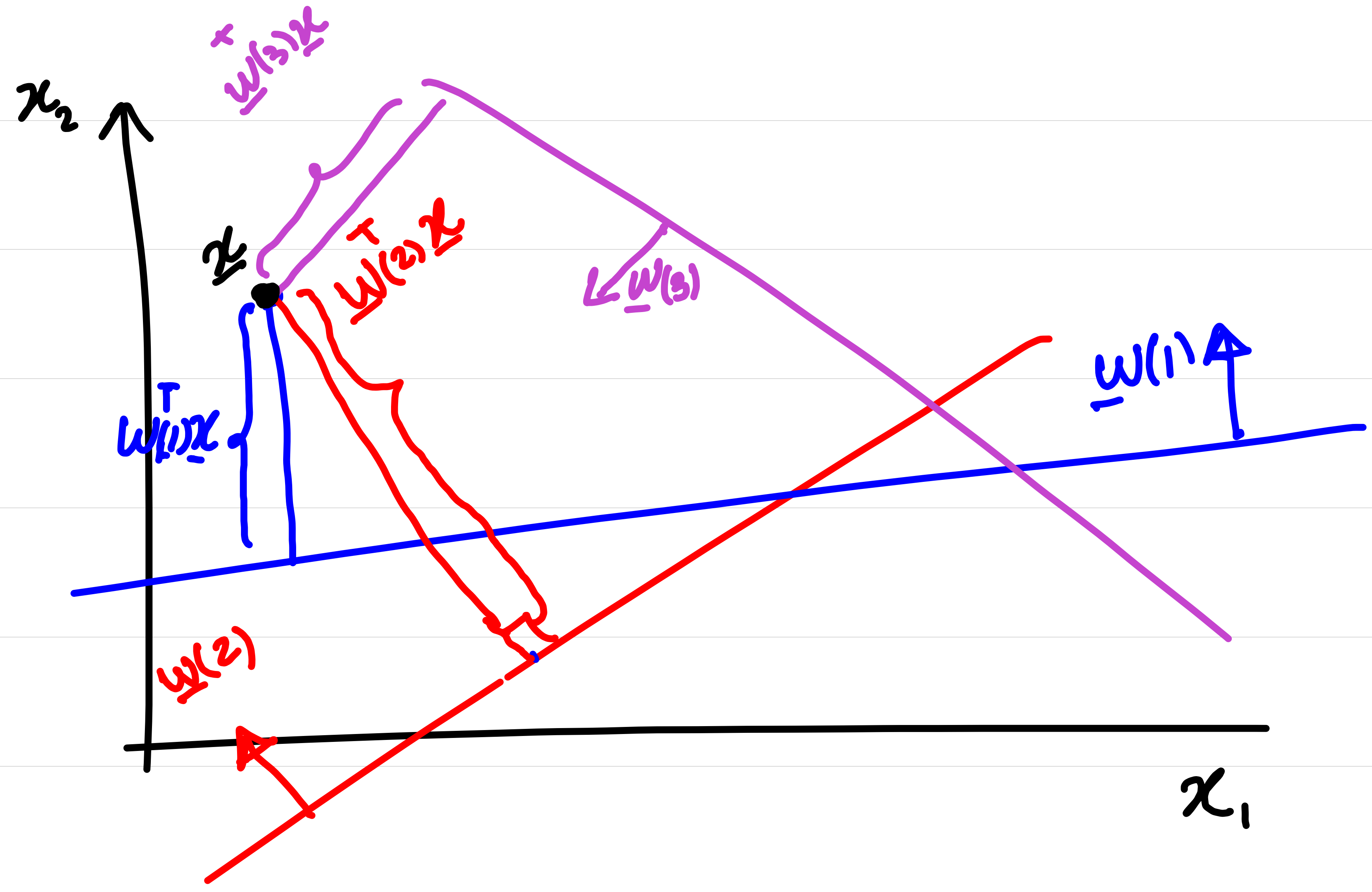
"Softmax function"

$$= \hat{P}_{\Omega}(i \mid \underline{x}_n)$$

- Error Criterion:

$$e_n(\Omega) = -\log \hat{P}_{\Omega}(y_n \mid \underline{x}_n) = -\underline{w}^T_{(y_n)} \underline{x}_n + \log \sum_{j=1}^{c} e^{\underline{w}^T_{(j)} \underline{x}_n}$$

E.g.: $d=2$, $C=3$



- $$\hat{P}_\Omega(1|\underline{x}) = \frac{e^{\underline{w}^T(1)\underline{x}}}{e^{\underline{w}^T(1)\underline{x}} + e^{\underline{w}^T(2)\underline{x}} + e^{\underline{w}^T(3)\underline{x}}}$$

- $$\hat{P}_\Omega(2|\underline{x}) = \frac{e^{\underline{w}^T(2)\underline{x}}}{e^{\underline{w}^T(1)\underline{x}} + e^{\underline{w}^T(2)\underline{x}} + e^{\underline{w}^T(3)\underline{x}}}$$

- $$\hat{P}_\Omega(3|\underline{x}) = \frac{e^{\underline{w}^T(3)\underline{x}}}{e^{\underline{w}^T(1)\underline{x}} + e^{\underline{w}^T(2)\underline{x}} + e^{\underline{w}^T(3)\underline{x}}}$$

- $\underline{x}$ is furthest away from line given by $\underline{w}(2)$. Hence, $\hat{P}_\Omega(2|\underline{x}) > \hat{P}_\Omega(1|\underline{x})$ and $\hat{P}_\Omega(2|\underline{x}) > \hat{P}_\Omega(3|\underline{x})$

To minimize $E_{in}(\Omega)$, we need to use GD. Hence, we must find the gradient.

$$\nabla_{\Omega}\, e_n(\Omega) = \begin{bmatrix} \dfrac{\partial e_n}{\partial w_0(1)}(\Omega) \\ \dfrac{\partial e_n}{\partial w_1(1)}(\Omega) \\ \vdots \\ \dfrac{\partial e_n}{\partial w_d(1)}(\Omega) \\ \vdots \\ \dfrac{\partial e_n}{\partial w_0(c)}(\Omega) \\ \vdots \\ \dfrac{\partial e_n}{\partial w_d(c)}(\Omega) \end{bmatrix} = \begin{bmatrix} \nabla_{\underline{w}(1)} e_n(\Omega) \\ \vdots \\ \nabla_{\underline{w}(c)} e_n(\Omega) \end{bmatrix}$$

SGD update rule:

$$\Omega_{t+1} = \Omega_t - \varepsilon_t \nabla_{\Omega_t} e_n(\Omega) \Longrightarrow \begin{bmatrix} w_{t+1}(1) \\ \vdots \\ w_{t+1}(c) \end{bmatrix} = \begin{bmatrix} \underline{w}_t(1) \\ \vdots \\ \underline{w}_t(c) \end{bmatrix} - \varepsilon_t \begin{bmatrix} \nabla_{\underline{w}(1)} e_n(\Omega) \\ \vdots \\ \nabla_{\underline{w}(c)} e_n(\Omega) \end{bmatrix}$$

SGD update:

In each iteration t :

Set $\Omega_t = \{ \underline{w}_t(1), \dots, \underline{w}_t(c) \}$

Sample a datapoint $n \sim \text{Uniform}(\{1, \dots, N\})$

For $i = 1, 2, \dots, c$

$$\underline{g}_t^{(i)} = \nabla_{\underline{w}_t(i)} e_n(\Omega)$$

$$\underline{V}_t(i) = -\underline{g}_t(i)$$

$$\underline{w}_{t+1}(i) = \underline{w}_t(i) + \varepsilon_t \underline{V}_t(i)$$

Computing $\nabla_{\underline{w}(i)}\, e_n(\Omega_t)$

$$e_n(\Omega_t) = -\log \hat{P}_{\Omega_t}(y_n \mid \underline{x}_n) = -\log \frac{e^{\underline{w}^T(y_n)\,\underline{x}_n}}{\sum_{j=1}^{C} e^{\underline{w}^T(j)\,\underline{x}_n}}$$

$$= -\underline{w}^T(y_n)\,\underline{x}_n + \log\left(\sum_{j=1}^{C} e^{\underline{w}^T(j)\,\underline{x}_n}\right)$$

For $i = y_n$,

$$\nabla_{\underline{w}_t(i)}\, e_n(\Omega_t) = \nabla_{\underline{w}_t(y_n)}\left(-\underline{w}^T(y_n)\,\underline{x}_n + \log\left(\sum_{j=1}^{C} e^{\underline{w}^T(j)\,\underline{x}_n}\right)\right)$$

$$= -\underline{x}_n + \frac{e^{\underline{w}^T(y_n)\,x_n}}{\sum_{j=1}^{C} e^{\underline{w}^T(j)\,\underline{x}_n}}\,\underline{x}_n$$

□ For $i \neq y_n$,

$$\nabla_{\underline{w}(i)} e_n(\Omega_t) = \nabla_{\underline{w}(i)} \left(-\underline{w}^T(y_n)\,\underline{x}_n + \log\left(\sum_{j=1}^{c} e^{\underline{w}^T(j)\,\underline{x}_n} \right) \right)$$

$$= \frac{e^{\underline{w}^T(i)\,\underline{x}_n}}{\sum_{j=1}^{c} e^{\underline{w}^T(j)\,\underline{x}_n}}\,\underline{x}_n$$

Softmax Logistic Regression for Binary Classification (C=2)

- What is the relation between Softmax Logistic regression for C=2 and binary logistic regression that we studied last week.

$$\hat{P}_\Omega(1|\underline{x}_n) = \frac{e^{\underline{w}^T(1)\underline{x}}}{e^{\underline{w}^T(1)\underline{x}} + e^{\underline{w}^T(2)\underline{x}_n}} = \frac{e^{(\underline{w}^T(1)-\underline{w}^T(2))\underline{x}}}{1+e^{(\underline{w}^T(1)-\underline{w}^T(2))\underline{x}}}$$

$$\hat{P}_\Omega(2|\underline{x}_n) = \frac{e^{\underline{w}^T(2)\underline{x}_n}}{e^{\underline{w}^T(1)\underline{x}_n} + e^{\underline{w}^T(2)\underline{x}_n}} = 1 - \hat{P}_\Omega(1|\underline{x}_n)$$

It is Logistic Regression with $\underline{w} = \underline{w}(1) - \underline{w}(2)$

Can we use GD/SGD for Linear Regression? Yes, you can

- $E_{in}(\underline{w}) = \frac{1}{N}\sum_{n=1}^{N} e_n(\underline{w}) = \frac{1}{N}\sum_{n=1}^{N} (\underline{w}^T\underline{x}_n - y_n)^2$.

- $\underline{w}_{ls}$, the optimal Solution.
  - $\underline{w}_{ls} = (X^T X)^{-1} X^T \underline{y}$
  - Complexity : $O(Nd^2 + d^3)$

- $E_{in}(\underline{w})$ is Convex. So, with GD/SGD, as $t \to \infty$, $\underline{w}_t$ Converges to the optimal Solution, i.e., $\underline{w}_{ls}$.
  - $\nabla_{\underline{w}} e_n(\underline{w}) = 2(\underline{w}^T \underline{x}_n - y_n)\underline{x}_n$
  - Time Complexity of SGD, Full-GD, Min_batch GD:
    - SGD: $O(d)$
    - Full GD: $O(Nd)$
    - Mini GD: $O(Md)$

■ Benefits of GD over Least-squares method:

1- Better Complexity

2- Most often, we are not interested in finding the exact optimal Solution.

↳ We only care about test error, not E_{in} (train error)

↳ If we have noisy data, the optimal solution leads to overfitting.

- In practice, we run GD for a few iterations,
- then, we do Validation and stop when the validation error starts deteriorating