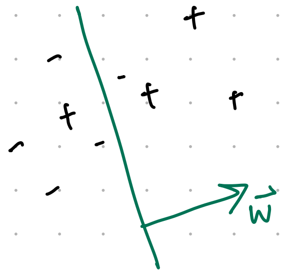
Logistic Regression

$$\boxed{P(y \mid \vec{x}) = \frac{1}{1 + e^{-y\vec{w}^T\vec{x}}} \quad \text{for } y \in \{+1, -1\}}$$



- By viewing $\vec{w}$ as parameter

$$\vec{w}_{MLE} = \underset{\vec{w}}{\operatorname{argmin}} \underbrace{\sum_{i=1}^{n} \log(1 + e^{-y_i \vec{w}^T \vec{x}_i})}_{l(\vec{w}) = \sum_{i=1}^{n} L(x_i, y_i, \vec{w})}$$

- Claim that $l(\vec{w})$ (logistic loss function) is continuous ✓, differentiable ✓, and convex ✓.

- The term $\vec{w}^T\vec{x}_i$ is the prediction of $\vec{x}_i$

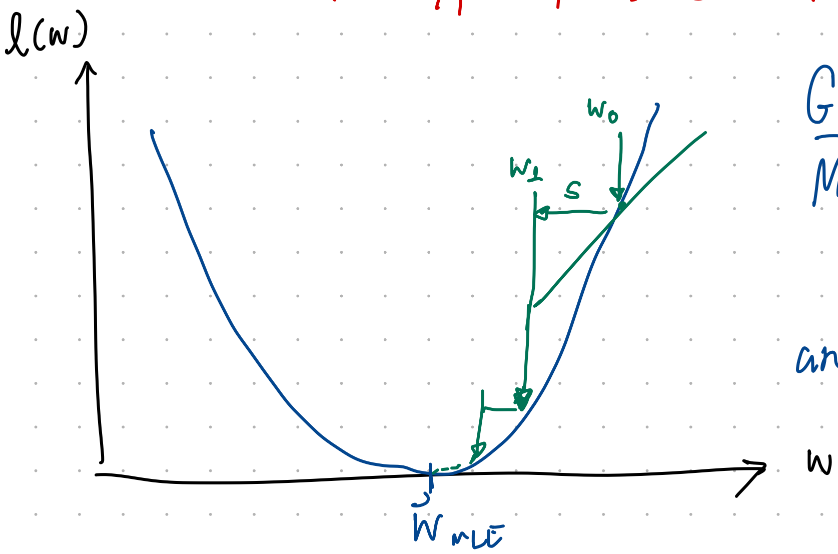
$$h(\vec{x}_i) = \operatorname{sign}(\vec{w}^T\vec{x})$$

- If the prediction is correct, then $h(\vec{x}_i) = y_i$

$$\Rightarrow -y_i \vec{w}^T\vec{x}_i < 0 \Rightarrow L(x_i, y_i, \vec{w}) \to 0$$

- If the prediction is wrong, then $h(\vec{x}_i) \neq y_i$

$$\Rightarrow -y_i \vec{w}^T\vec{x}_i > 0 \Rightarrow L(x_i, y_i, \vec{w}) \to -y_i \vec{w}^T\vec{x}_i$$



Gradient Descent

Make a progress with step

$$\vec{s} = -\alpha \nabla l(\vec{w})$$

and repeat the process until converged

We need to ensure that $l(w_{i+1}) > l(w_i)$ at each step i

Taylor Expansion

- If $\|s\|_2$ is small (meaning $\vec{w}+\vec{s}$ is close to $\vec{w}$), then the following holds

$$\boxed{\ell(\vec{w}+\vec{s}) \approx \ell(\vec{w}) + \nabla \ell(\vec{w})^T \vec{s}}$$

- With $\vec{s} = -\alpha \nabla \ell(\vec{w})$, where $\alpha > 0$, we have that

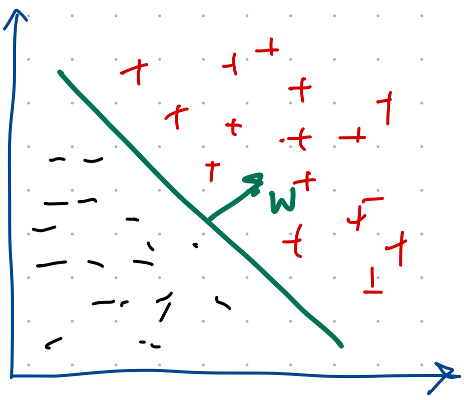
$$\begin{aligned}
\ell(\vec{w}+\vec{s}) &\approx \ell(\vec{w}) + \nabla \ell(\vec{w})^T \left(-\alpha \nabla \ell(\vec{w})\right) \\
&= \ell(\vec{w}) + \left(-\alpha \left(\nabla \ell(\vec{w})^T\right) \nabla \ell(\vec{w})\right) \\
&= \ell(\vec{w}) + (-\underset{>0}{\alpha} \underbrace{\|\nabla \ell(\vec{w})\|_2}_{>}) \\
&< \ell(\vec{w})
\end{aligned}$$

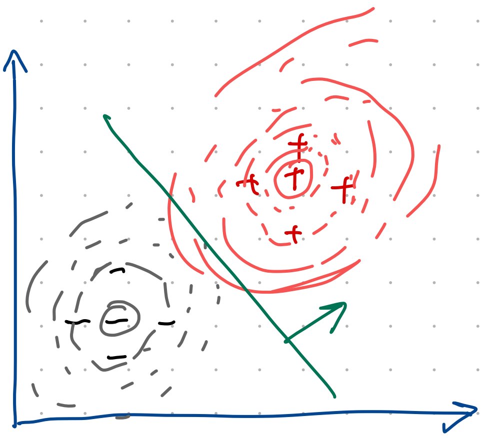
What's to pick up

- Logistic loss function $\ell(\vec{w}) = \sum_{i=1}^{n} \log(1 + e^{-y_i \vec{w}^T \vec{x}_i})$ also measures how well the linear classifier performs on our data. Ideally, we want $\ell(\vec{w}) \rightarrow 0$.
- The Gradient Descent is a hill climbing algorithm that finds an optimal point of several loss function
- Logistic regression is an ML algorithm for binary classification that gives a linear classifier (discrimative counterpart of naive Bayes)
- No assumption about modeling distributions made for Logistic regression

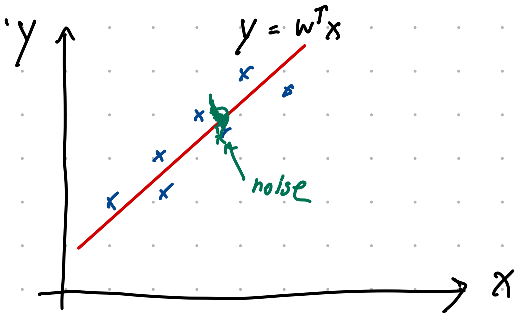
Naive Bayes vs Logistic Regression

- With little data and if the modeling distribution is appropiate, Naive Bayes tends to outperform Logistic Regression
- As data sets becomes larges, Logistic Regression often outperforms Naive Bayes, which suffers from the fact that the modeling assumptions made for $P(x|y)$ are probably not exactly corret





Linear Regression



- Assume $y_i \in \mathbb{R}$
- Assume $y_i = \underbrace{\vec{w}^T \vec{x}_i}_{\text{line}} + \underbrace{\varepsilon_i}_{\text{noise}}$,

where $\varepsilon_i \sim N(0, \sigma^2)$

$$\Longleftrightarrow \quad y_i \sim N(\vec{w}^T \vec{x}_i, \sigma^2) \Rightarrow P(y_i | \vec{x}_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\vec{w}^T \vec{x}_i - y_i)^2}{2\sigma^2}}$$

The goal is to estimate the vector $\vec{w}$ that defines the slope.

Estimate $\vec{w}$ with MLE

$$\vec{w}_{MLE} = \underset{\vec{w}}{\arg\max} \prod_{i=1}^{d} P(y_i \mid \vec{x}_i ; \vec{w})$$

$$= \underset{\vec{w}}{\arg\max} \sum_{i=1}^{d} \log(P(y_i \mid \vec{x}_i ; \vec{w}))$$

$$= \underset{\vec{w}}{\arg\max} \sum_{i=1}^{d} \left[\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \log\left(e^{-\frac{(\vec{w}^T\vec{x}_i - y_i)^2}{2\sigma^2}}\right) \right]$$

$$= \underset{\vec{w}}{\arg\max} -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (\vec{w}^T \vec{x}_i - y_i)^2$$

$$= \underset{\vec{w}}{\arg\min} \underbrace{\frac{1}{n} \sum_{i=1}^{n} (\vec{w}^T \vec{x}_i - y_i)^2}_{l(\vec{w})}$$

** Remark: Here, we are minimizing the square loss function $l(\vec{w})$

Ways to find $\vec{w}_{MLE}$: ① "Gradient Descent" since the square loss is continuous, differentiable, and convex (parabola)

② Compute via a closed-form solution of the square loss function

---

Closed-form solution of $\vec{w}_{MLE}$

$$\vec{w}_{MLE} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \vec{y}, \text{ where } \mathbf{X} = \begin{bmatrix} \leftarrow \vec{x}_1^T \rightarrow \\ \leftarrow \vec{x}_2^T \rightarrow \\ \vdots \\ \leftarrow x_n^T \rightarrow \end{bmatrix} \text{ and } \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$\mathbf{X}\vec{w} - \vec{y} = \begin{bmatrix} — \vec{x}_1^T — \\ — \vec{x}_2^T — \\ \vdots \\ — \vec{x}_n^T — \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$= \begin{bmatrix} \vec{w}^T\vec{x}_1 \\ \vec{w}^T\vec{x}_2 \\ \vdots \\ \vec{w}^T\vec{x}_n \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \vec{w}^T\vec{x}_1 - y_1 \\ \vec{w}^T\vec{x}_2 - y_2 \\ \vdots \\ \vec{w}^T\vec{x}_n - y_n \end{bmatrix}$$

$$l(\vec{w}) = (\mathbf{X}\vec{w} - \vec{y})^T(\mathbf{X}\vec{w} - \vec{y}) = \sum_{i=1}^{n}(\vec{w}^T\vec{x}_i - y_i)^2$$

$$\nabla_{\vec{w}} l(\vec{w}) = \nabla_{\vec{w}}(\mathbf{X}\vec{w} - \vec{y})^T(\mathbf{X}\vec{w} - \vec{y})$$

$$= \nabla_{\vec{w}}(w^T\mathbf{X}^T\mathbf{X}\vec{w} - y^T\mathbf{X}\vec{w} - x^Tw^T\vec{y} + \cancel{\vec{y}^T\vec{y}})$$

$$= 2X^TX\vec{w} - y^T\mathbf{X} - \mathbf{x}^Ty$$

$$\boxed{\nabla_{\vec{w}} l(\vec{w}) = 2X^TX\vec{w} - 2\mathbf{X}^Ty}$$

To find $\vec{w}_{MLE}$, $\nabla_{\vec{w}} l(\vec{w}) \overset{set}{=} 0$

$$\Rightarrow \cancel{2}X^TX\vec{w}_{MLE} = \cancel{2}X^Ty$$

$$\Rightarrow X^TX\vec{w}_{MLE} = X^Ty$$

$$\boxed{\vec{w}_{MLE} = (X^TX)^{-1}X^Ty}$$ ← Closed-form solution (complexity $O(d^3)$)

*** If d is large, the solution is not possible to compute.