Department of Linguistics and Translation
香港城市大學
City University of Hong Kong

Computational Linguistics LT3233

Jixing Li

Lecture 7: Logistic Regression

Slides adapted from Dan Jurafsky

Lecture plan

- Naive Bayes and Laplace smoothing review
- Logistic regression
  - feature representation
  - classification function: **sigmoid**
  - loss function: **cross-entropy loss**
  - optimization algorithm: **gradient descent**
- Short break (15 mins)
- Hands-on exercises

Naive Bayes classifier

Bayes rule: $P(\text{male}|\text{卓琳}) = P(\text{卓琳}|\text{male})P(\text{male})$

$P(\text{female}) = \frac{7}{10}$ $P(\text{male}) = \frac{3}{10}$

卓琳= [卓, 琳] → **features**

$P(\text{卓琳}|\text{male}) \approx P(\text{卓}|\text{male})\, P(\text{琳}|\text{male})$

$$P(\text{卓}|\text{male}) = \frac{Count(\text{卓 in } male\ names)}{Count(all\ chatacters\ in\ male\ names)} = \frac{1}{6}$$

$$P(\text{琳}|\text{male}) = \frac{Count(\text{琳 in } male\ names)}{Count(all\ chatacters\ in\ male\ names)} = \frac{0}{6}$$

$$P(\text{卓琳}|\text{male}) = \frac{1}{6} \text{x} \frac{0}{6} = 0$$

- 歐承璋 M
- 李思穎 F
- 陳敏琪 F
- 廖倚琳 F
- 吴建瑞 M
- 馮紫晴 F
- 廖卓楠 M
- 徐婉晴 F
- 周咏楠 F
- 馬卓妍 F
- 袁卓琳 ?

Laplace (Add-1) smoothing

$$P(卓|\text{male}) = \frac{Count(卓 \text{ in } male\ names) + 1}{Count(all\ chatacters\ in\ male\ names) + Count(V)} = \frac{2}{6+17}$$

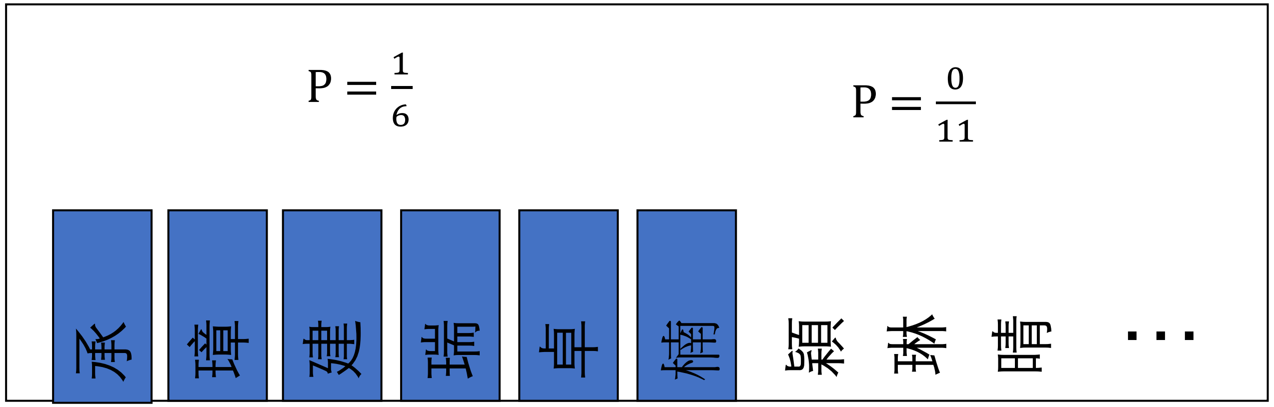
$$P(琳|\text{male}) = \frac{Count(琳 \text{ in } male\ names) + 1}{Count(all\ chatacters\ in\ male\ names) + Count(V)} = \frac{1}{6+17}$$

$$P(卓琳|\text{male}) = \frac{2}{23} \text{ x } \frac{1}{23} = \frac{2}{529}$$

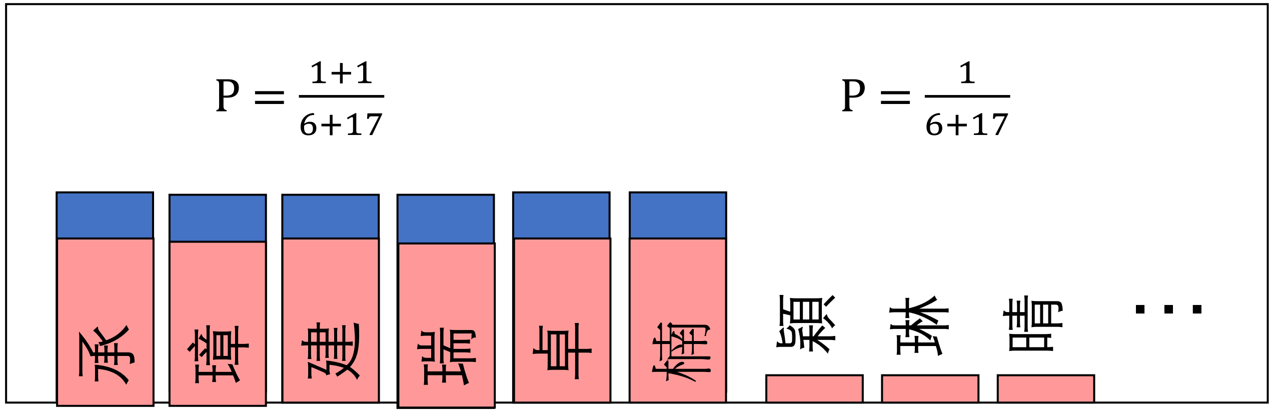
Why vocabulary size in the denominator?

- 歐承璋 M
- 李思穎 F
- 陳敏琪 F
- 廖倚琳 F
- 吴建瑞 M
- 馮紫晴 F
- 廖卓楠 M
- 徐婉晴 F
- 周咏楠 F
- 馬卓妍 F
- 袁卓琳 ?

Laplace (Add-1) smoothing



$$\frac{1}{6} \text{ x } 6 + \frac{0}{11} \text{ x } 11 = 1$$



$$\frac{2}{23} \text{ x } 6 + \frac{1}{23} \text{ x } 11 = 1$$

- 歐承璋 M
- 李思穎 F
- 陳敏琪 F
- 廖倚琳 F
- 吳建瑞 M
- 馮紫晴 F
- 廖卓楠 M
- 徐婉晴 F
- 周咏楠 F
- 馬卓妍 F
- 袁卓琳 ?

Logistic regression

The task of text classification

- *Input*:
  - a document x
  - a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$

- *Output*: a predicted class $\hat{y} \in C$

Naive Bayes: Compare P (male|卓琳) P (female|卓琳)
→ generative classifier

Logistic regression: P (male|卓琳)
→ discriminative classifier

Components of logistic regression

1. **feature representation** of the input. For each input observation $x^{(i)}$, a vector of features $[x_1, x_2, \ldots, x_n]$.

   $x^{(i)} = [$卓, 琳$]$

   $= [$卓, 琳, Cheuk, Lam, LLA$]$

2. **classification function** that computes $\hat{y}$, the estimated class: **sigmoid** functions

3. **objective function for learning**: **cross-entropy loss**

4. **algorithm for optimizing** the objective function: **gradient descent**

Features in logistic regression

Input vector: $x = [x_1, x_2, \ldots, x_n]$

[卓，琳, Cheuk, Lam, LLA]

Probability of these features in female names:

→ $x = [0.5, 0.7, 0.5, 0.6, 0.8]$

Weights: one per feature: $w = [w_1, w_2, \ldots, w_n]$

→ $w = [0.1, 0.8, -0.1, 0.2, 0.7]$

Prediction: $z = w \cdot x + b$

$$z = w_1{*}x_1 + w_2{*}x_2 + w_3{*}x_3 + w_4{*}x_4 + w_5{*}x_5 + b$$

$$= 0.25 + 0.56 + (-0.05) + 0.12 + 0.56 + 0.3$$

$$= 1.74$$

Transform prediction into probability

$$z = w \cdot x + b$$

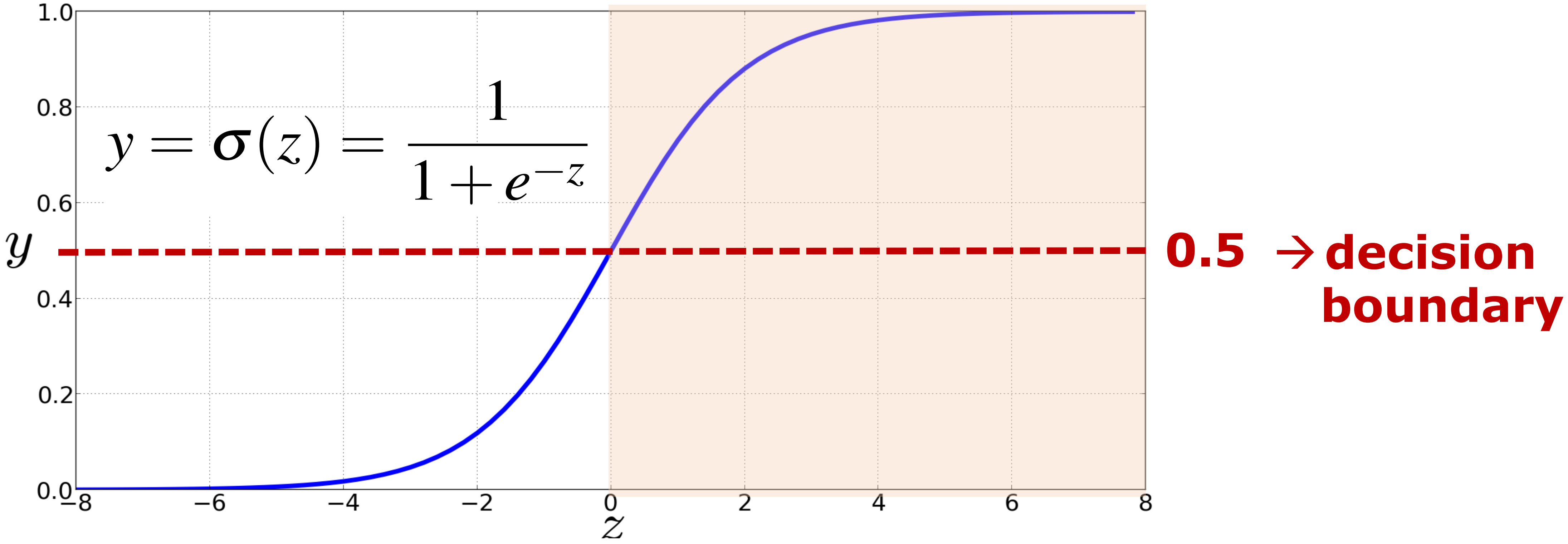
z is a number, But we We'd like a classifier that gives us a probability, just like Naive Bayes did

Solution: use a function of z that goes from 0 to 1

$$y = \sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+\exp(-z)}$$

→ **the sigmoid function**

The sigmoid function



$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

$\text{if } w \cdot x + b > 0$

$\text{if } w \cdot x + b \leq 0$

Example

[卓，琳, Cheuk, Lam, LLA]

x = [0.5, 0.7, 0.5, 0.6, 0.8]

w = [0.1, 0.8, -0.1, 0.2, 0.7]

$z = w \cdot x + b$

$= w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + w_4 * x_4 + w_5 * x_5 + b$

$= 0.25 + 0.56 + (-0.05) + 0.12 + 0.56 + 0.3$

$= 1.74$

$\hat{y} = \sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-1.74}} = 0.85 > 0.5 \rightarrow$ female

How to calculate weights?

Supervised classification:

We know the correct label y (either 0 or 1) for each x.

But what the system produces is an estimate, $\hat{y}$

We want to know how far is the classifier output:

$$\hat{y} = \sigma(w \cdot x + b)$$

from the true output:

$$y = \text{either 0 or 1}$$

We'll call this difference the loss:

$$L(\hat{y}, y) = \text{how much } \hat{y} \text{ differs from the true } y$$

Cross-entropy loss

Goal: **maximize** the probability of the correct label $p(y|x)$

Since there are only 2 outcomes (0 or 1), we can express the probability $p(y|x)$ from our classifier as:

$$p(y|x) = \hat{y}^y (1-\hat{y})^{1-y}$$

if y=1, this simplifies to $\hat{y}$
if y=0, this simplifies to 1- $\hat{y}$

Now take the log of both sides:

$$\begin{aligned} \log p(y|x) &= \log \left[\hat{y}^y (1-\hat{y})^{1-y}\right] \\ &= y \log \hat{y} + (1-y) \log(1-\hat{y}) \end{aligned}$$

Now flip sign to turn this into a loss: Something to **minimize**

$$L_{\mathrm{CE}}(\hat{y}, y) = -\log p(y|x) = -[y \log \hat{y} + (1-y) \log(1-\hat{y})]$$

cross-entropy loss: negative log likelihood loss

Example

[卓，琳, Cheuk, Lam, LLA]

x = [0.5, 0.7, 0.5, 0.6, 0.8]

w = [0.1, 0.8, -0.1, 0.2, 0.7]

b = 0.5

$\hat{y} = \sigma(w \cdot x + b) = 0.85$

if 卓琳 is female: y = 1:

$L_{CE}(\hat{y}, y)$ = -(ylog $\hat{y}$ +(1-y)log(1- $\hat{y}$)) $= -\log(0.85) = 0.16$

if 卓琳 is male: y = 0:

$L_{CE}(\hat{y}, y)$ = -(ylog $\hat{y}$ +(1-y)log(1- $\hat{y}$)) $= -\log(1 - 0.85) = 1.9$

→ The loss is greater when the prediction is wrong

Minimize the loss

Let's make explicit that the loss function is parameterized by weights $\theta=(w,b)$

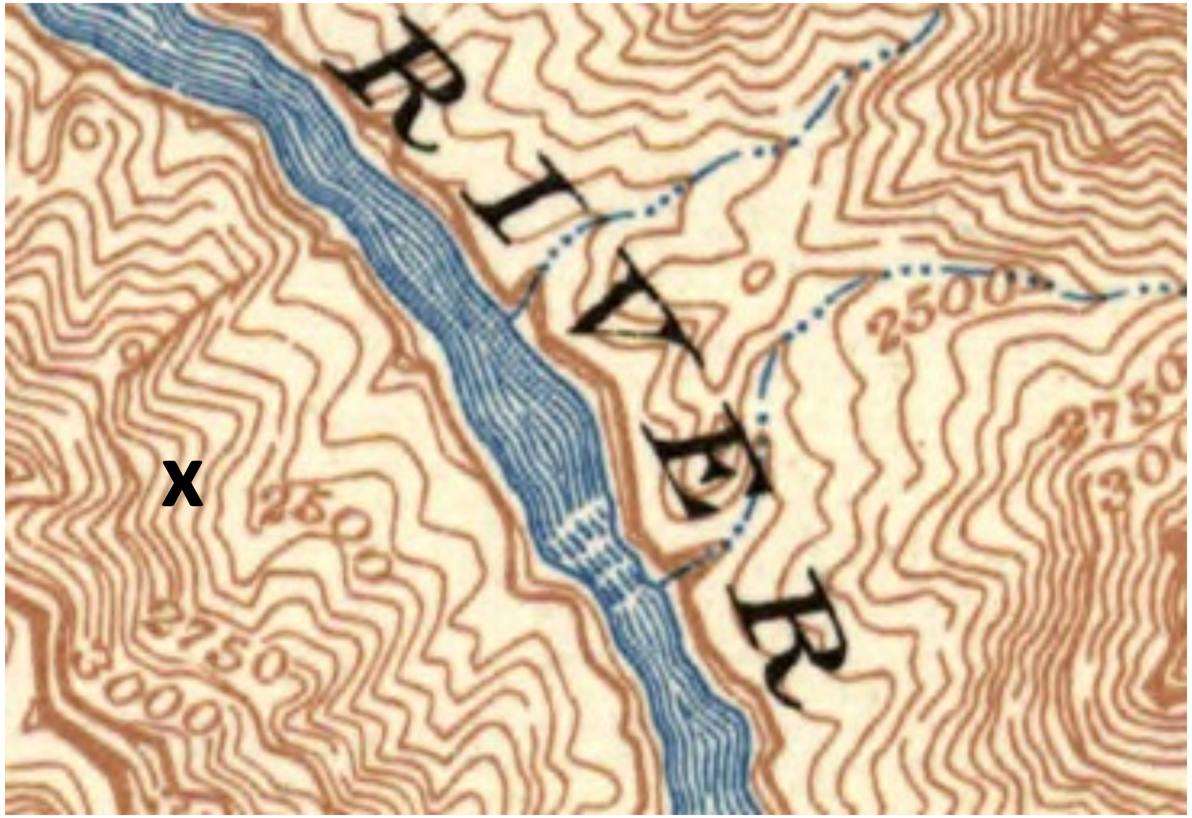
And we'll represent $\hat{y}$ as $f(x;\theta)$ to make the dependence on θ more obvious

We want the weights that minimize the loss, averaged over all examples:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{\mathrm{CE}}(f(x^{(i)};\theta), y^{(i)})$$

Gradient descent

How do I get to the bottom of this river canyon?



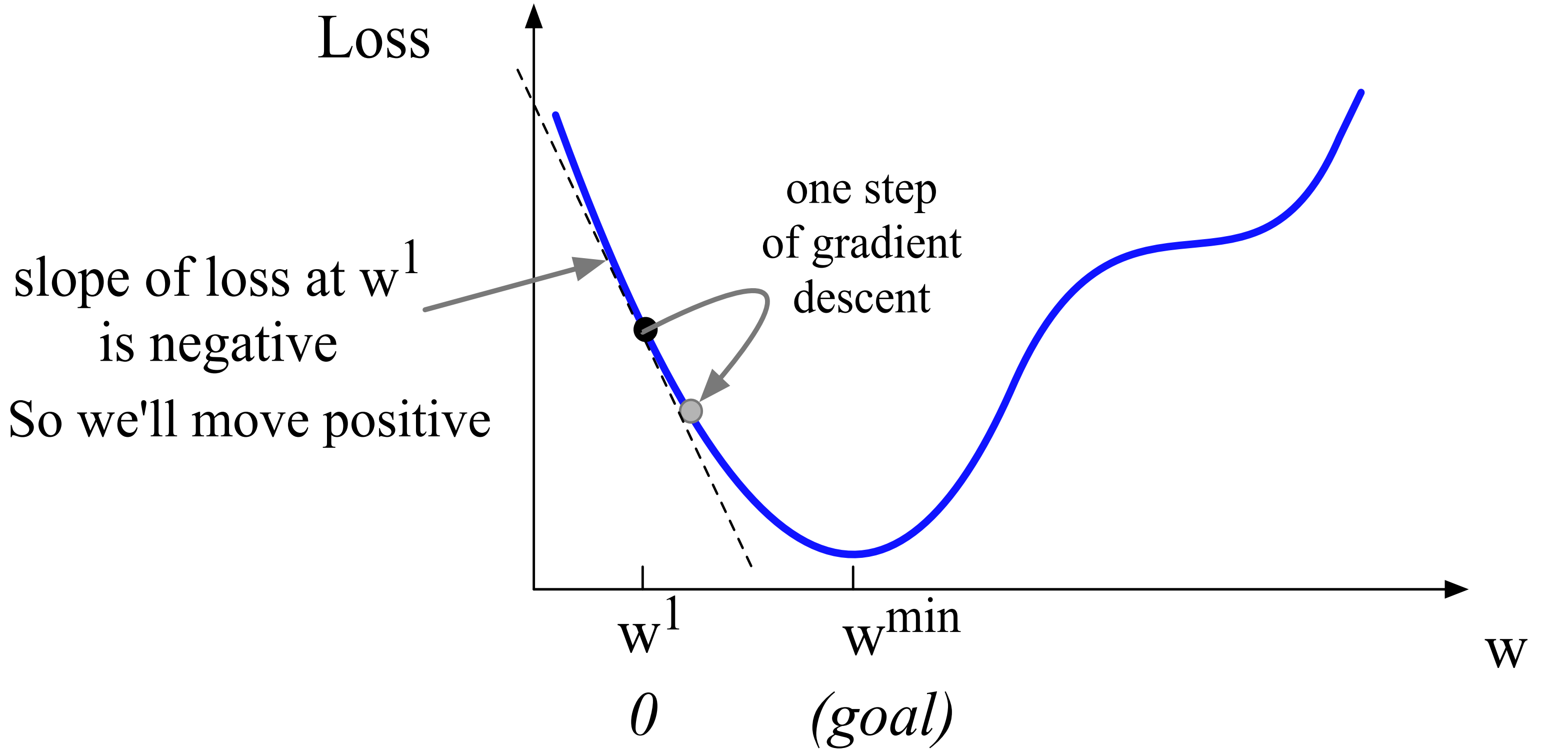
Look around me 360°

Find the direction of steepest slope down

Go that way

Gradient descent for a single scaler

Minimize loss: Given the current w, Move w in the reverse direction from the slope of the function



The **gradient** of a function of many variables is a vector pointing in the direction of the greatest increase in a function.

Gradient descent: Find the gradient of the loss function at the current point and move in the opposite direction.

Gradient descent

The new weight w^{t+1} is the old weight w^t minus the value of the gradient weighted by a learning rate η

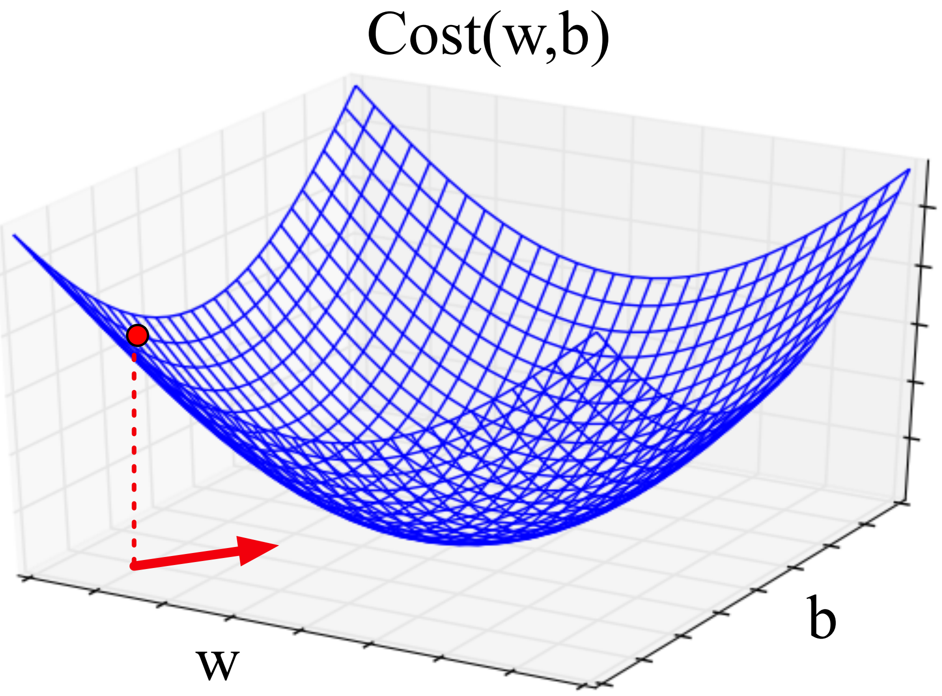
$$w^{t+1} = w^t - \boxed{\eta} \boxed{\frac{d}{dw} L(f(x;w), y)}$$

learning rate: Higher learning rate means move w faster
→ a **hyperparameter**
not learned by algorithm from supervision, but are chosen by algorithm designer.

gradient (a vector of the derivatives with respect to the weight w)

Gradient in N-dimensional space

The gradient expresses the directional components of the sharpest slope along each of the N dimensions. For each dimension w_i, we express the slope as a partial derivative ∂ of the loss ∂w_i



The derivative of

$$L_{CE}(\hat{y}, y) = -[y \log \sigma(w \cdot x + b) + (1-y) \log (1 - \sigma(w \cdot x + b))]$$

is:

$$\frac{\partial L_{CE}(\hat{y}, y)}{\partial w_j} = [\sigma(w \cdot x + b) - y] x_j$$

Example

[卓, 琳, Cheuk, Lam, LLA]
$x = [0.5, 0.7, 0.5, 0.6, 0.8]$

1. initialize w and b, set η
$w = [0, 0, 0, 0, 0], b = 0, \eta = 0.1$

2. compute $\hat{y}$
$\hat{y} = \sigma(w \cdot x + b) = 0.5$
3. compute the gradients for w and b
$Gw = (0.5 - y)x = -0.5x = [-0.25, -0.35, -0.25, -0.3, -0.4]$
$Gb = 0.5 - y = -0.5$
4. update w and b
$w_{t+1} = w_t - Gw = [0, 0, 0, 0, 0] - [-0.25, -0.35, -0.25, -0.3, -0.4]$
$= [0.25, 0.35, 0.25, 0.3, 0.4]$
$b_{t+1} = b_t - Gb = 0 - (-0.5) = 0.5$

Calculate gradient descent over all examples

[卓, 琳, Cheuk, Lam, LLA] $x_1 = [0.5, 0.7, 0.5, 0.6, 0.8]$
[承, 璋, Shing Cheung, LLA] $x_2 = [-0.6, -0.8, -0.1, -0.6, 0.8]$

1. initialize w and b, set η
$w = [0, 0, 0, 0, 0], b = 0, \eta = 0.1$

2. compute $\hat{y}$
$\hat{y}_1 = \sigma(w \cdot x + b) = 0.5, \hat{y}_2 = \sigma(w \cdot x + b) = 0.5$

3. compute the gradients for w and b
$Gw = \frac{1}{2}((0.5 - y)x_1 + (0.5 - y)x_2) = \frac{1}{2}(-0.5x_1 - 0.5x_2) = [0.025, 0.025, -0.1, 0, -0.2]$
$Gb = \frac{1}{2}((0.5 - y_1) + (0.5 - y_2)) = 0$

4. update w and b
$w_{t+1} = w_t - Gw = [0, 0, 0, 0, 0] - [0.025, 0.025, -0.1, 0, -0.2]$
$= [-0.025, -0.025, 0.1, 0, 0.2]$
$b_{t+1} = b_t - Gb = 0$

Batch gradient descent

Update weights using the gradient that averaged over all examples:

$$\frac{1}{m}\sum_{i=1}^{m}(\widehat{y^{i}}-y^{i})X_j^i$$

To do

- Do HW6
- Optional reading: **SLP** Ch5