ECLT5810/SEEM5750
Logistic Regression for Classification

Reference: “Speech and Language Processing” Chapter 5.1-5.7

https://web.stanford.edu/~jurafsky/slp3/

Classification: definition

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

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

Binary Classification in Logistic Regression

- Given a series of input/output pairs:
  - $(x^{(i)}, y^{(i)})$
- For each observation $x^{(i)}$
  - We represent $x^{(i)}$ by a **feature vector** $[x_1, x_2, \ldots, x_n]$
  - We compute an output: a predicted class $\hat{y}^{(i)} \in \{0,1\}$

Features in logistic regression

- For feature x_i, weight w_i tells is how important is x_i
  - x_1 ="income_level is high/low": w_1 = +10
  - x_2 ="student is yes/no": w_2 = -2
  - x_3 ="spending_history is high/low": w_3 = +5

Logistic Regression for one observation x

- Input observation: vector $x = [x_1, x_2, \ldots, x_n]$
- Weights: one per feature: $W = [w_1, w_2, \ldots, w_n]$
  - Sometimes we call the weights $\theta = [\theta_1, \theta_2, \ldots, \theta_n]$
- Output: a predicted class $\hat{y} \in \{0,1\}$

(multinomial logistic regression: $\hat{y} \in \{0, 1, 2, 3, 4\}$)

How to do classification

- For each feature x_i, weight w_i tells us importance of x_i
- Also, the model has a bias term b
- We'll sum up all the weighted features and the bias

$$z = \sum_{i=1}^{n} w_i x_i + b$$

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

- If this sum is high, we say y=1; if low, then y=0

But we want a probabilistic classifier

- We need to formalize "sum is high".
- We'd like a principled classifier that gives us a probability
- We want a model that can tell us:

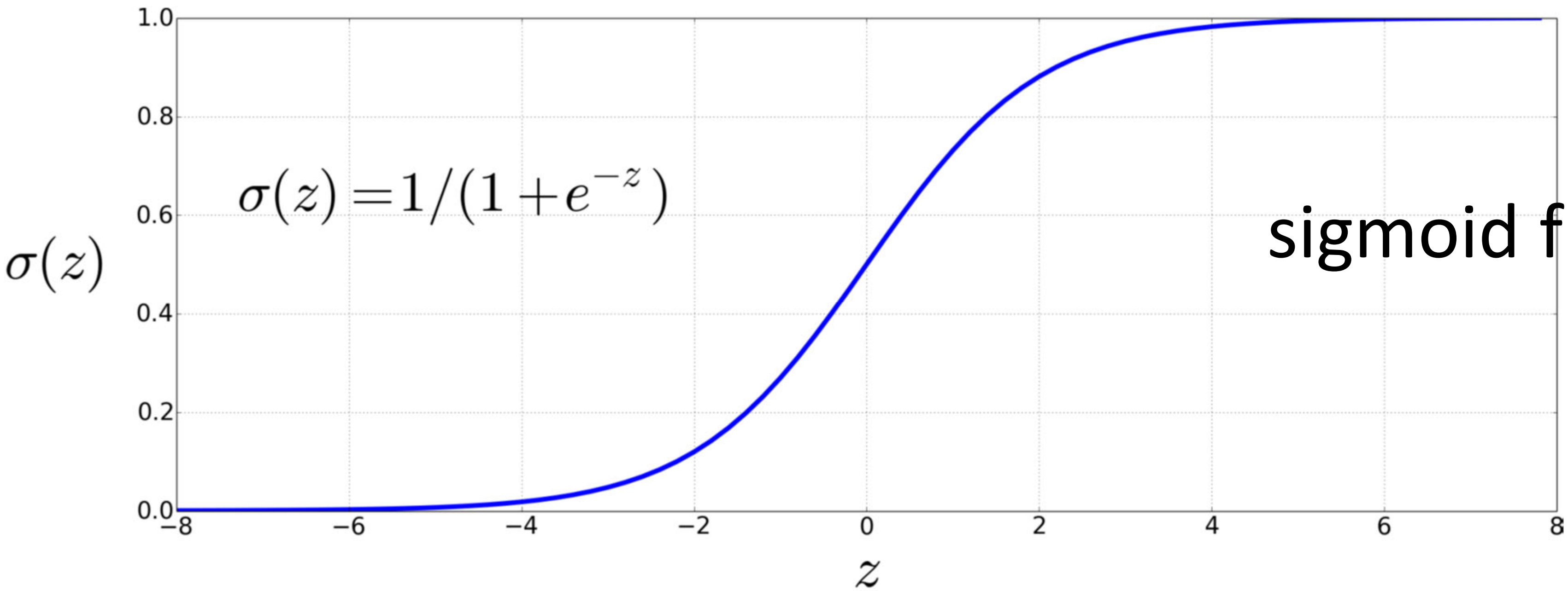
  $p(y=1|x;\theta)$

  $p(y=0|x;\theta)$

One issue

$z = w \cdot x + b$ isn't a probability, it's just a number!

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



sigmoid function

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

Idea of logistic regression

- We'll compute $w \cdot x + b$
- And then we'll pass it through the sigmoid function:

$$\sigma(w \cdot x + b)$$

- And we'll just treat it as a probability

Making probabilities with sigmoids

$$\begin{aligned}
P(y=1) &= \sigma(w \cdot x + b) \\
&= \frac{1}{1+\exp(-(w \cdot x + b))}
\end{aligned}$$

$$\begin{aligned}
P(y=0) &= 1-\sigma(w \cdot x + b) \\
&= 1-\frac{1}{1+\exp(-(w \cdot x + b))} \\
&= \frac{\exp(-(w \cdot x + b))}{1+\exp(-(w \cdot x + b))}
\end{aligned}$$

Interesting Property

$$
\begin{aligned}
P(y=0) &= 1-\sigma(w\cdot x+b) \\
&= 1-\frac{1}{1+\exp\left(-(w\cdot x+b)\right)} \\
&= \frac{\exp\left(-(w\cdot x+b)\right)}{1+\exp\left(-(w\cdot x+b)\right)} \quad = \quad \sigma(-(w\cdot x+b))
\end{aligned}
$$

Therefore, the sigmoid function has the property:

$$1-\sigma(x)=\sigma(-x)$$

Turning a probability into a classifier

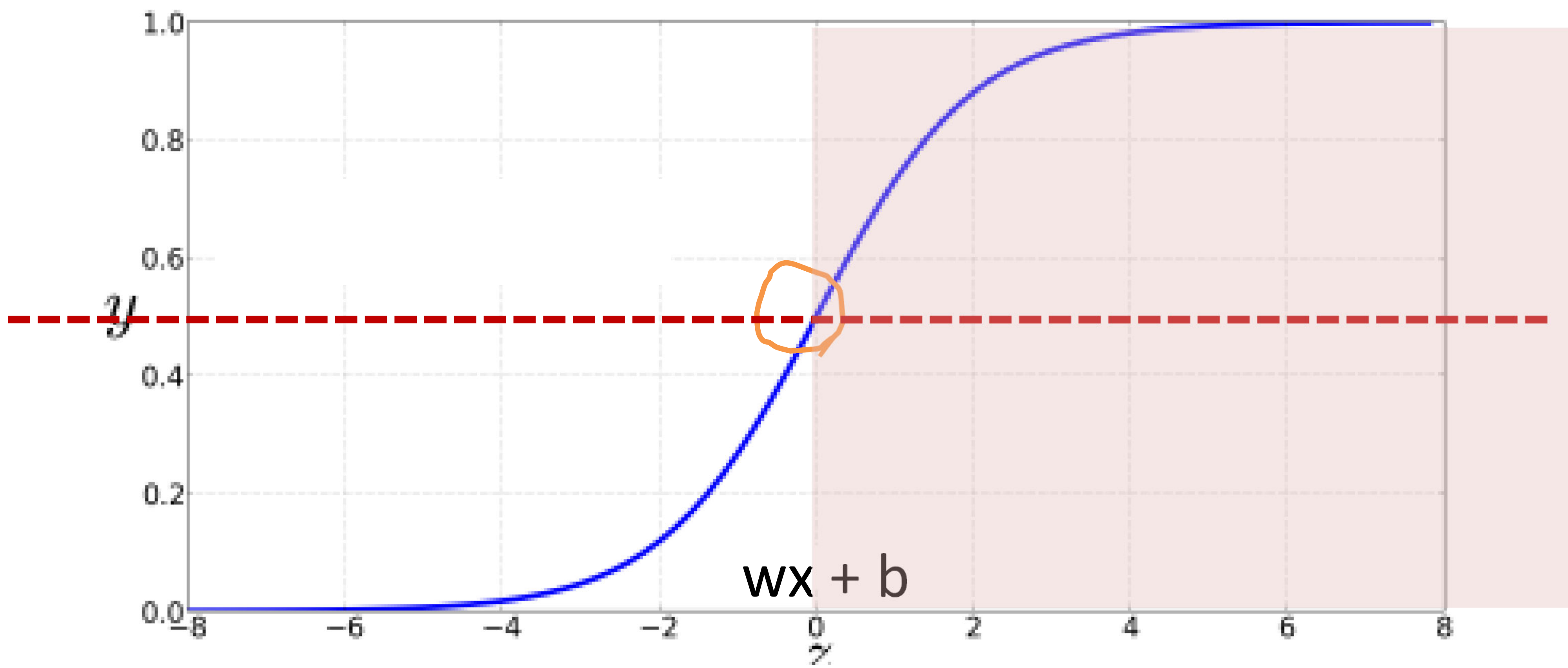
$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

0.5 here is called the **decision boundary**

The probabilistic classifier

$$P(y=1) = \sigma(w \cdot x + b)$$
$$= \frac{1}{1+e^{-(w \cdot x + b)}}$$

P(y=1)



Turning a probability into a classifier

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

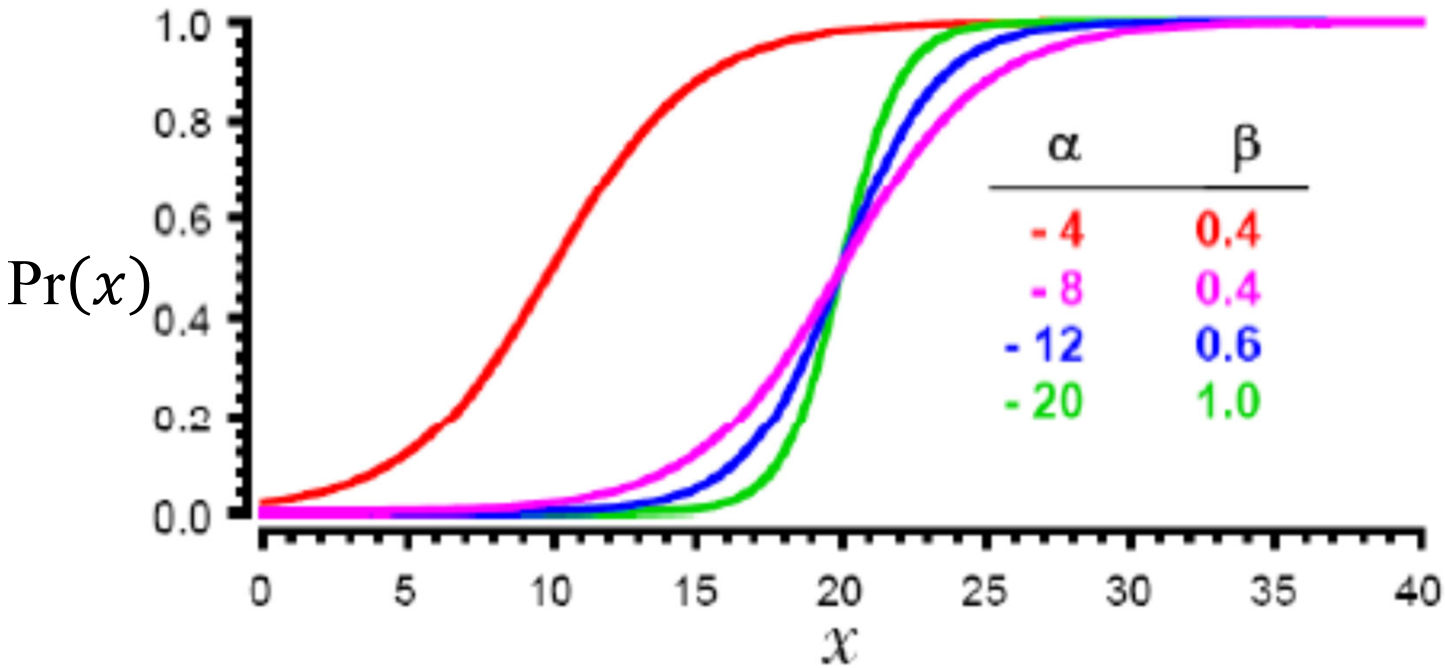
if $w \cdot x + b > 0$

if $w \cdot x + b \leq 0$

Logistic Regression
Shape of sigmoid curve

- Consider 1-dimensional x

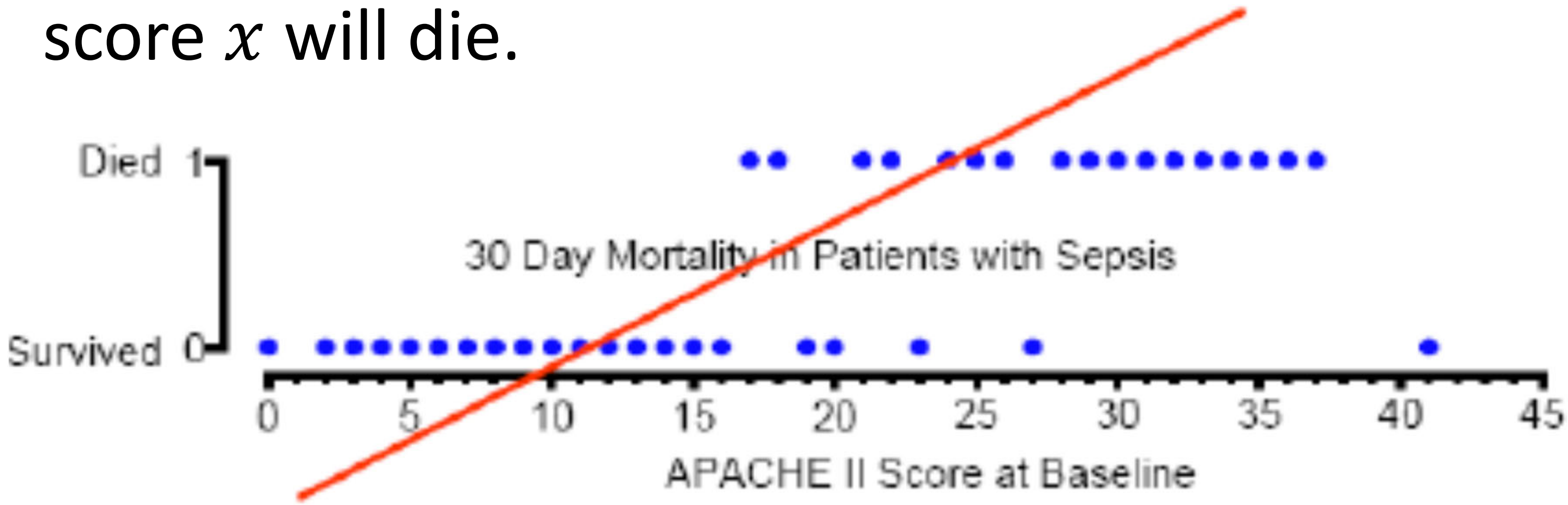
$$\Pr(x) = \frac{1}{1+\exp\left(-(\alpha+\beta x)\right)}$$



Logistic Regression

An Example of One-dimension

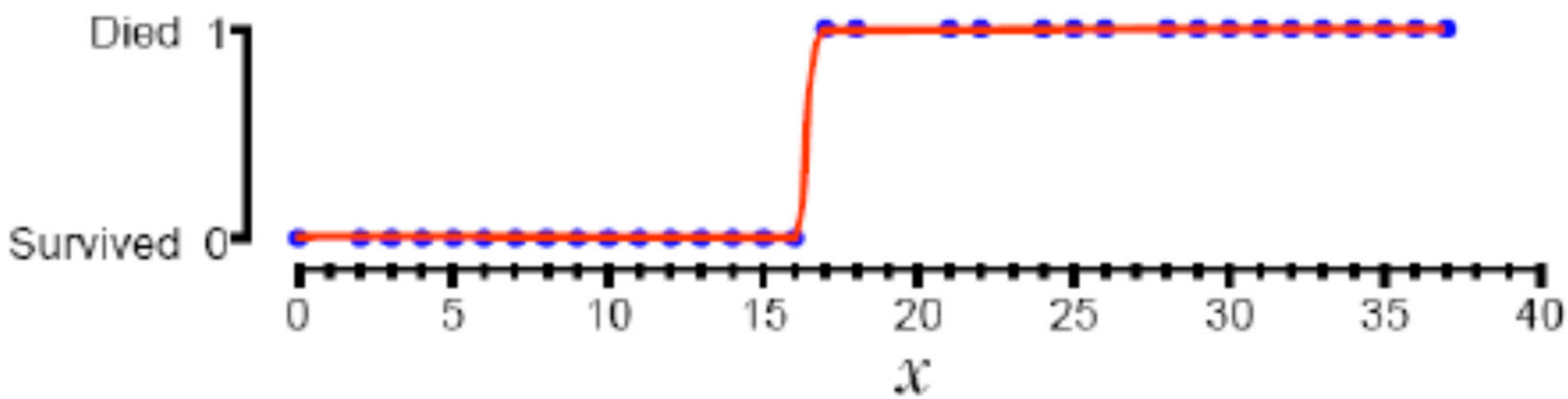
- We wish to predict death from baseline APACHE II score of patients.
- Let $\Pr(x)$ be the probability that a patient with score x will die.



- Note that linear regression would not work well since it could produce probabilities less than 0 or greater than 1

Logistic Regression
An Example of One-dimension

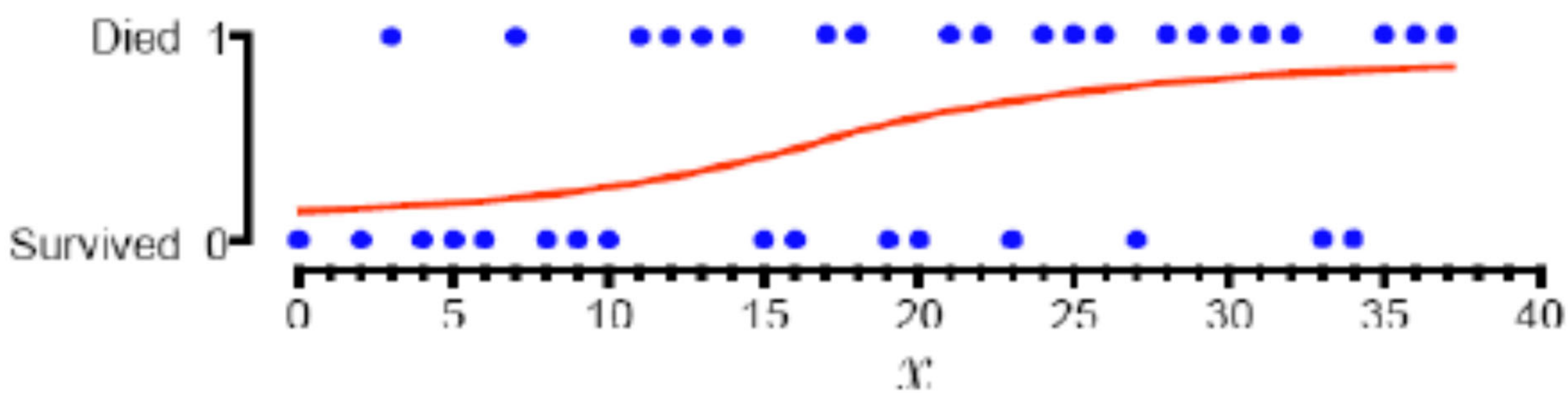
- Data that has a sharp survival cut off point between patients who live or die will lead to a large value of β



Logistic Regression

An Example of One-dimension

- One the other hand, if the data has a lengthy *transition* from survival to death, it will lead to a low value of β



Where did the W's come from? Learning W from data

- 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 set w and b to minimize the **distance** between our estimate $\hat{y}^{(i)}$ and the true $y^{(i)}$.
- We need a distance estimator: a **loss function** or a **cost function**
- We need an optimization algorithm to update w and b to minimize the loss.

Learning components

- A loss function:
    - **cross-entropy loss**
- An optimization algorithm:
    - **stochastic gradient descent**

The distance between $\hat{y}$ and y

- We want to know how far is the classifier output:

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

- from the true output:

  y [= either 0 or 1]

- We'll call this difference loss function:

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

Deriving cross-entropy loss for a single observation x

- Consider the probability of the correct label in the training data (also called **likelihood** function) $p(y|x)$
- Recall that $\hat{y}$ denotes the classifier output. There are only 2 discrete outcomes, i.e. 0 or 1.
- We wish to express that if the correct label $y = 1$, the expression is $\hat{y}$. If the correct label $y = 0$, the expression is $1 - \hat{y}$

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

- The goal is to find the parameters, i.e. w and b, that can maximize the likelihood function

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

Deriving cross-entropy loss for a single observation x

Recall that the goal is to maximize the likelihood function

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

Now take the log of both sides (mathematically handy)

$$\text{Maximize:} \quad \log p(y|x) = \log\left[\hat{y}^y (1-\hat{y})^{1-y}\right]$$
$$= y \log \hat{y} + (1-y) \log(1-\hat{y})$$

whatever values maximize log p(y|x) will also maximize p(y|x)

Deriving cross-entropy loss for a single observation x

$$\text{Maximize:} \quad \log p(y|x) = \log\left[\hat{y}^{y}(1-\hat{y})^{1-y}\right]$$
$$= y\log\hat{y} + (1-y)\log(1-\hat{y})$$

- Now flip sign to turn this into a loss: something to minimize
- **Negative log likelihood loss** or **Cross-entropy loss** (because is formula for cross-entropy (y, $\hat{y}$))

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

- Or, plugging in the definition of $\hat{y}$:

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

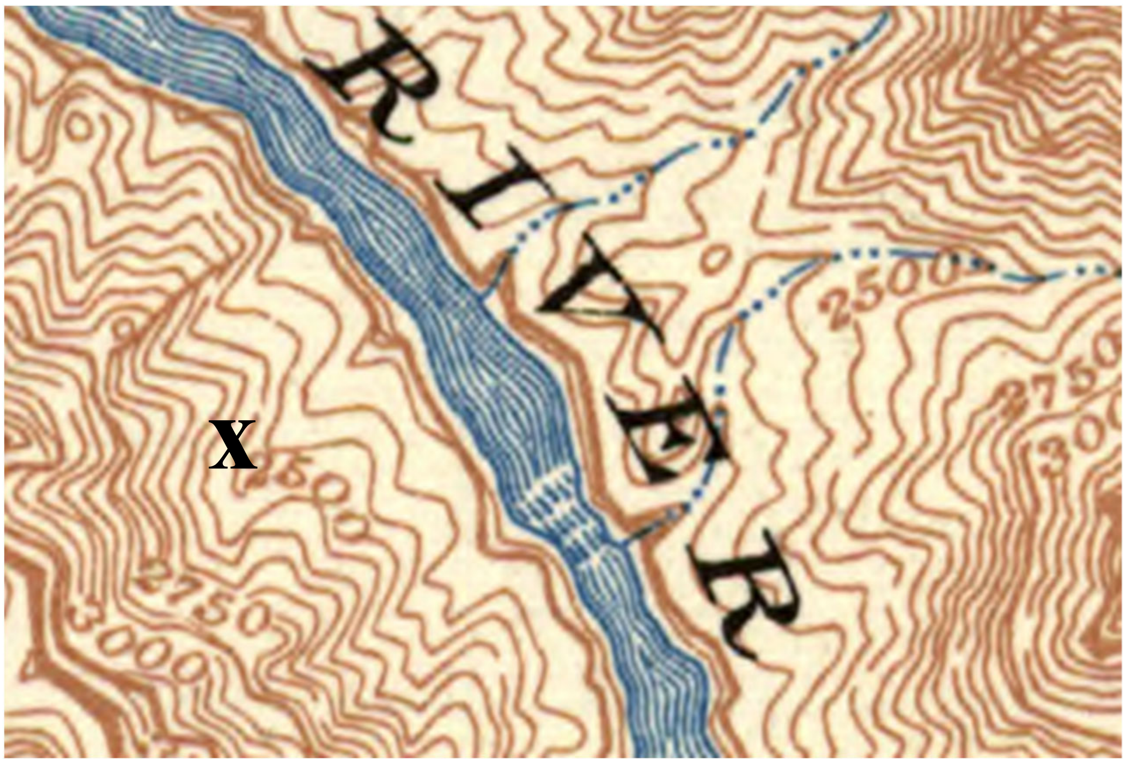
Our goal: minimize the loss

- Let's make explicit that the loss function is parameterized by weights θ=(w,b)
- And we'll represent $\hat{y}$ as f (x; θ) to make the dependence on θ more obvious
- We want the weights that minimize the loss, averaged over all examples:

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

Intuition of 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

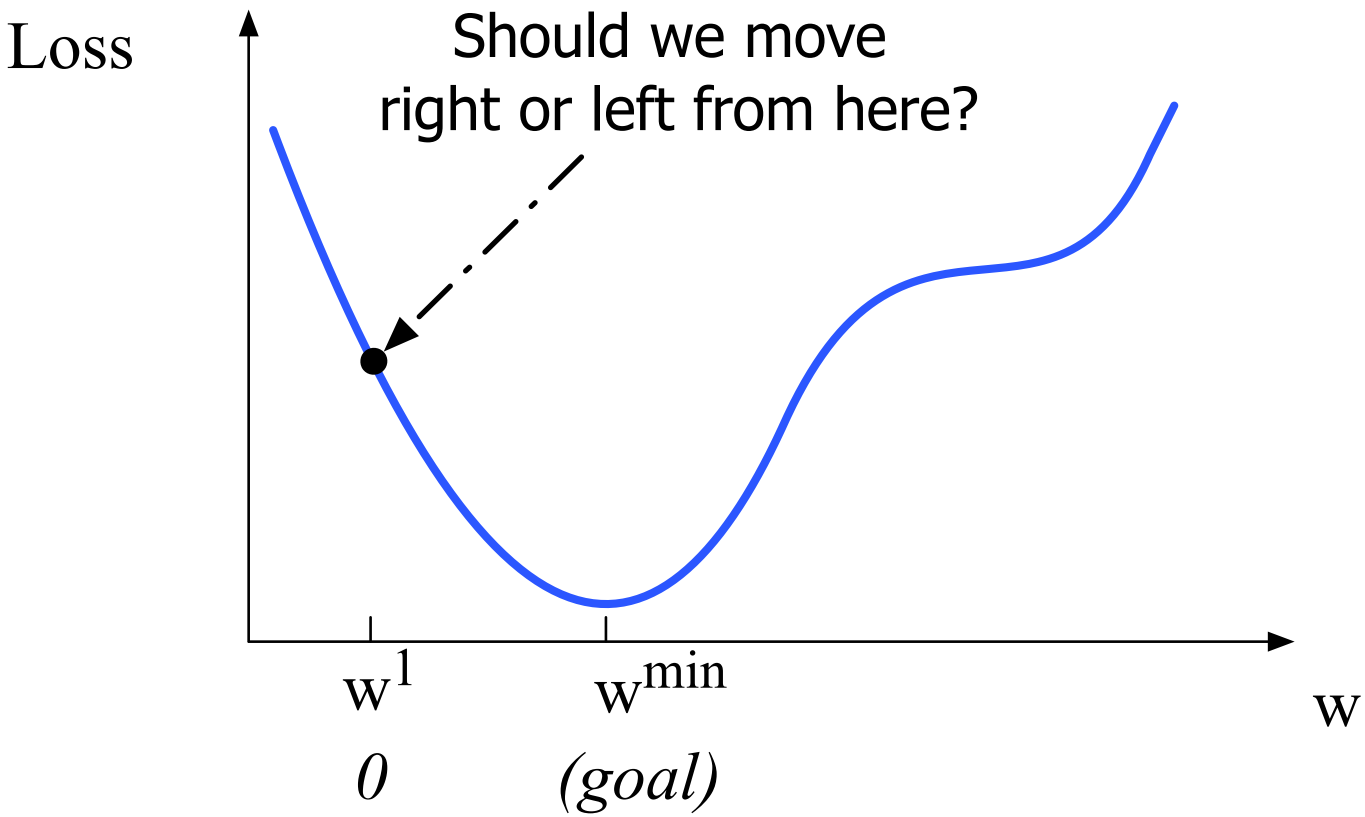
Our goal: minimize the loss

- For logistic regression, loss function is **convex**
- A convex function has just one minimum
- Gradient descent starting from any point is guaranteed to find the minimum
  - (Loss for neural networks is non-convex)

Let's first visualize for a single scalar w

Q: Given current w, should we make it bigger or smaller?

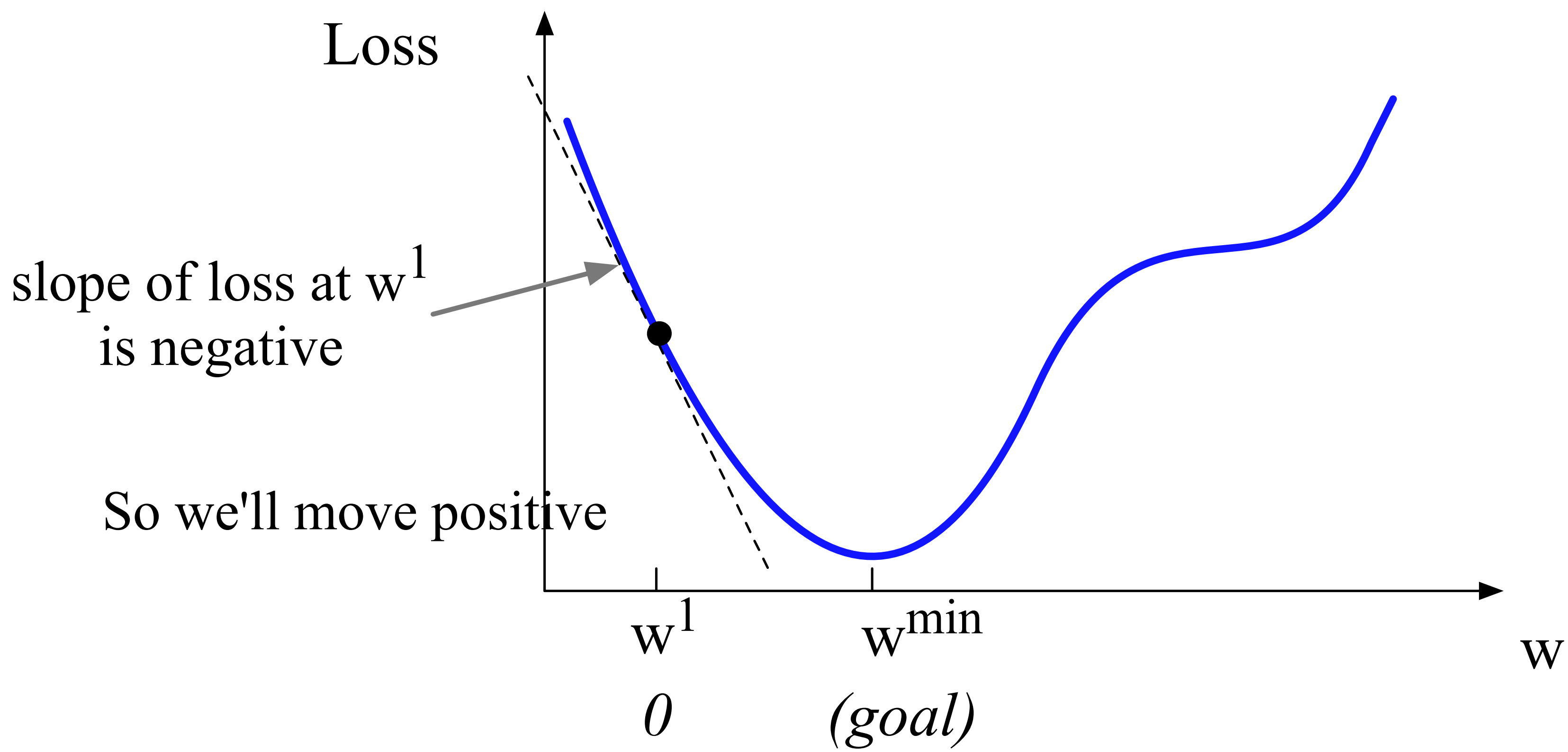
A: Move w in the reverse direction from the slope of the function



Let's first visualize for a single scalar w

Q: Given current w, should we make it bigger or smaller?

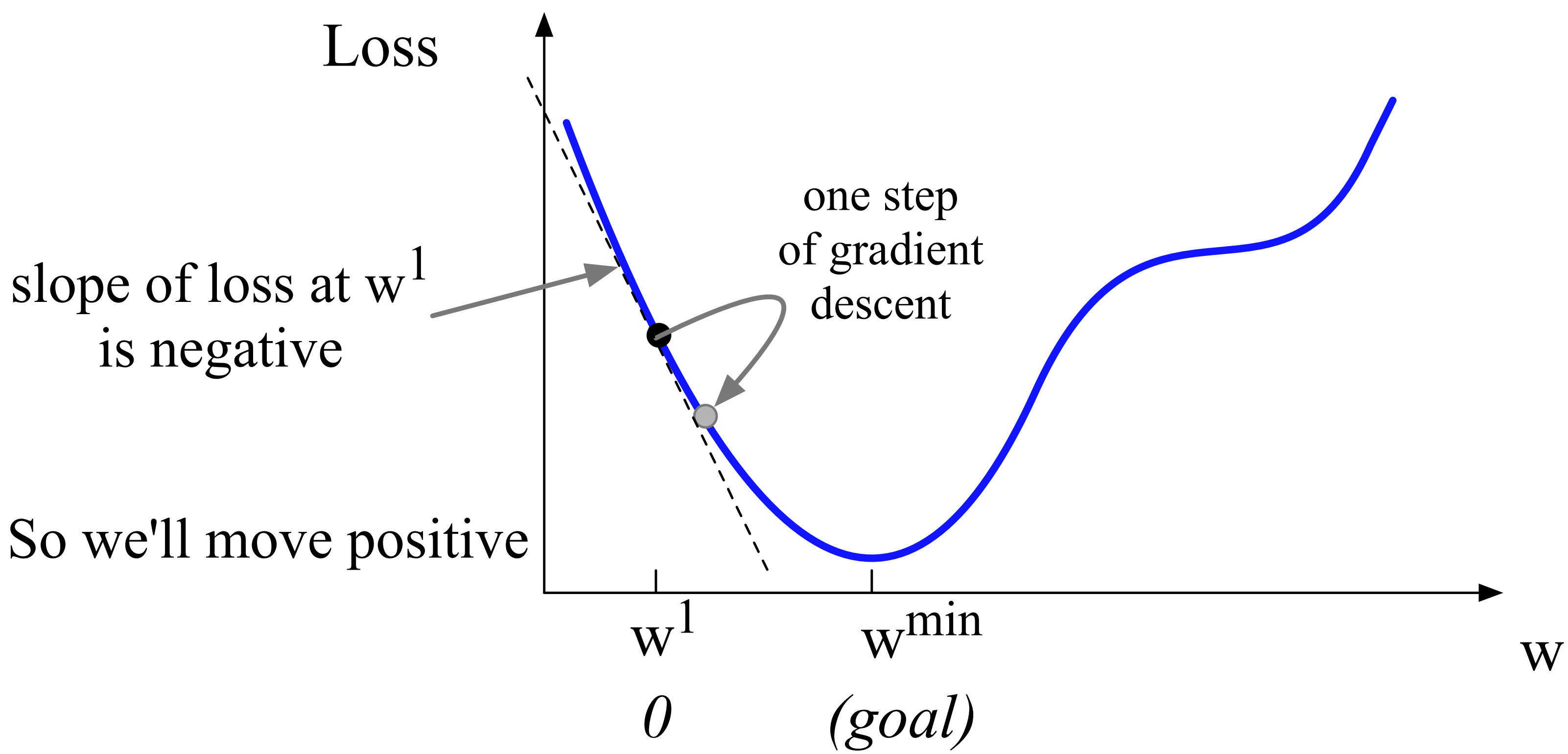
A: Move w in the reverse direction from the slope of the function



Let's first visualize for a single scalar w

Q: Given current w, should we make it bigger or smaller?

A: Move w in the reverse direction from the slope of the function



Gradients

- 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.

How much do we move in that direction ?

- The value of the gradient (slope in our example) $\frac{d}{dw}L(f(x;w),y)$ weighted by a **learning rate** η
- Higher learning rate means move *w* faster

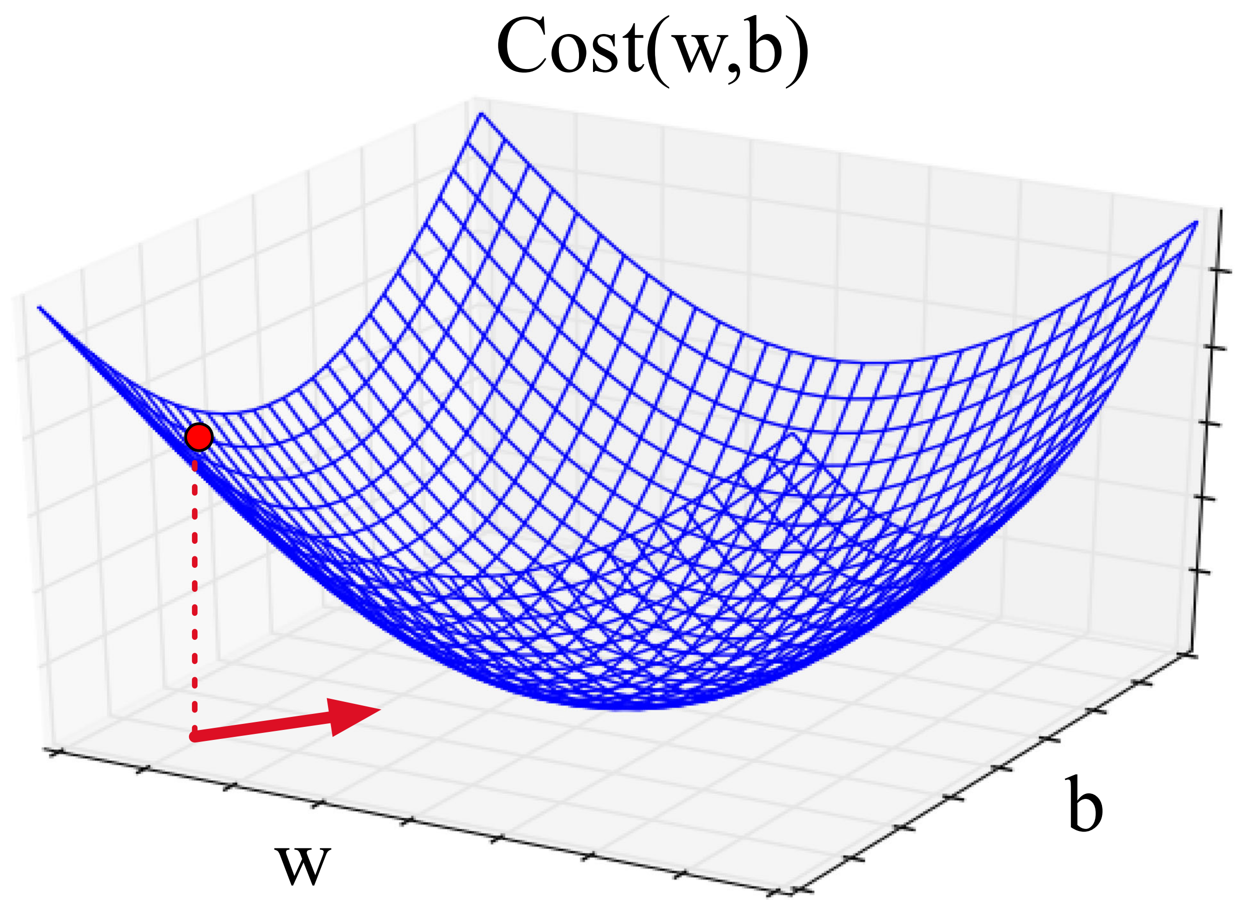
$$w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x;w),y)$$

Now let's consider N dimensions

- We want to know where in the N-dimensional space (of the *N* parameters that make up θ) we should move.
- The gradient is just such a vector; it expresses the directional components of the sharpest slope along each of the *N* dimensions.

Imagine 2 dimensions, w and b

- Visualizing the gradient vector at the red point
- It has two dimensions shown in the x-y plane



Real gradients

- Are much longer; lots and lots of weights
- For each dimension w_i the gradient component *i* tells us the slope with respect to that variable.
  - "How much would a small change in w_i influence the total loss function *L*?"
  - We express the slope as a partial derivative ∂ of the loss ∂w_i
- The gradient is then defined as a vector of these partials.

The gradient

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

$$\nabla L(f(x;\theta),y) = \begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x;\theta),y) \\ \frac{\partial}{\partial w_2} L(f(x;\theta),y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x;\theta),y) \\ \frac{\partial}{\partial b} L(f(x;\theta),y) \end{bmatrix}$$

The final equation for updating θ based on the gradient is thus

$$\theta^{t+1} = \theta^{t} - \eta \nabla L(f(x;\theta),y)$$

What are these partial derivatives for logistic regression?

The loss function

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

The elegant derivative of this function (see book chapter 5.8 for derivation)

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

Algorithm

function STOCHASTIC GRADIENT DESCENT($L()$, $f()$, x, y) **returns** θ

    # where: L is the loss function

    #   f is a function parameterized by θ

    #   x is the set of training inputs $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

    #   y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots, y^{(m)}$

$\theta \leftarrow 0$

repeat til done   # see caption

  For each training tuple $(x^{(i)}, y^{(i)})$ (in random order)

    1. Optional (for reporting):   # How are we doing on this tuple?

       Compute $\hat{y}^{(i)} = f(x^{(i)};\theta)$ # What is our estimated output $\hat{y}$?

       Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$) from the true output $y^{(i)}$?

    2. $g \leftarrow \nabla_\theta L(f(x^{(i)};\theta), y^{(i)})$ # How should we move θ to maximize loss?

    3. $\theta \leftarrow \theta - \eta\, g$ # Go the other way instead

return θ

Hyperparameters

- The learning rate η is a **hyperparameter**
  - too high: the learner will take big steps and overshoot
  - too low: the learner will take too long
- Hyperparameters:
  - Briefly, a special kind of parameter for an ML model
  - Instead of being learned by algorithm from supervision (like regular parameters), they are chosen by algorithm designer.

Working through an example

- One step of gradient descent
- A mini-sentiment example, where the true y=1 (positive)
- Two features with values:

$x_1 = 3$

$x_2 = 2$

Assume 3 parameters (2 weights and 1 bias) in Θ^0 are zero:

$w_1 = w_2 = b = 0$

$\eta = 0.1$

Example of gradient descent

$w_1 = w_2 = b = 0;$
$x_1 = 3; \quad x_2 = 2$

- Update step for update θ is:

$$\theta^{t+1} = \theta^{t} - \eta \nabla L(f(x;\theta), y)$$

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

- Gradient vector has 3 dimensions:

$$\nabla_{w,b} = \begin{bmatrix} \frac{\partial L_{\mathrm{CE}}(\hat{y},y)}{\partial w_1} \\ \frac{\partial L_{\mathrm{CE}}(\hat{y},y)}{\partial w_2} \\ \frac{\partial L_{\mathrm{CE}}(\hat{y},y)}{\partial b} \end{bmatrix}$$

Example of gradient descent

- Update step for update θ is:

$w_1 = w_2 = b = 0;$
$x_1 = 3; \quad x_2 = 2$

$$\theta^{t+1} = \theta^{t} - \eta \nabla L(f(x;\theta), y)$$

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

- Gradient vector has 3 dimensions:

$$\nabla_{w,b} = \begin{bmatrix} \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial w_1} \\ \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial w_2} \\ \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial b} \end{bmatrix} = \begin{bmatrix} (\sigma(w \cdot x + b) - y)x_1 \\ (\sigma(w \cdot x + b) - y)x_2 \\ \sigma(w \cdot x + b) - y \end{bmatrix}$$

Example of gradient descent

$w_1 = w_2 = b = 0;$
$x_1 = 3; \quad x_2 = 2$

- Update step for update θ is:

$$\theta^{t+1} = \theta^t - \eta \nabla L(f(x;\theta), y)$$

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

- Gradient vector has 3 dimensions:

$$\nabla_{w,b} = \begin{bmatrix} \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial w_1} \\ \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial w_2} \\ \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial b} \end{bmatrix} = \begin{bmatrix} (\sigma(w \cdot x + b) - y)x_1 \\ (\sigma(w \cdot x + b) - y)x_2 \\ \sigma(w \cdot x + b) - y \end{bmatrix} = \begin{bmatrix} (\sigma(0) - 1)x_1 \\ (\sigma(0) - 1)x_2 \\ \sigma(0) - 1 \end{bmatrix} = \begin{bmatrix} -0.5x_1 \\ -0.5x_2 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix}$$

Example of gradient descent

$$\nabla_{w,b} = \begin{bmatrix} \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial w_1} \\ \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial w_2} \\ \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial b} \end{bmatrix} = \begin{bmatrix} (\sigma(w \cdot x + b) - y)x_1 \\ (\sigma(w \cdot x + b) - y)x_2 \\ \sigma(w \cdot x + b) - y \end{bmatrix} = \begin{bmatrix} (\sigma(0) - 1)x_1 \\ (\sigma(0) - 1)x_2 \\ \sigma(0) - 1 \end{bmatrix} = \begin{bmatrix} -0.5x_1 \\ -0.5x_2 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix}$$

Now that we have a gradient, we compute the new parameter vector θ^1 by moving θ^0 in the opposite direction from the gradient:

$$\theta^{t+1} = \theta^t - \eta \nabla L(f(x;\theta), y)$$

η = 0.1;

$$\theta^1 =$$

Example of gradient descent

$$\nabla_{w,b} = \begin{bmatrix} \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial w_1} \\ \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial w_2} \\ \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial b} \end{bmatrix} = \begin{bmatrix} (\sigma(w \cdot x + b) - y)x_1 \\ (\sigma(w \cdot x + b) - y)x_2 \\ \sigma(w \cdot x + b) - y \end{bmatrix} = \begin{bmatrix} (\sigma(0) - 1)x_1 \\ (\sigma(0) - 1)x_2 \\ \sigma(0) - 1 \end{bmatrix} = \begin{bmatrix} -0.5x_1 \\ -0.5x_2 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix}$$

Now that we have a gradient, we compute the new parameter vector θ^1 by moving θ^0 in the opposite direction from the gradient:

$$\theta^{t+1} = \theta^t - \eta \nabla L(f(x;\theta), y)$$

$\eta = 0.1$;

$$\theta^1 = \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} - \eta \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix} = \begin{bmatrix} .15 \\ .1 \\ .05 \end{bmatrix}$$

Note that enough negative examples would eventually make w_2 negative

Mini-batch training

- Stochastic gradient descent chooses a single random example at a time.
- That can result in choppy movements
- More common to compute gradient over batches of training instances.
- **Batch training**: entire dataset
- **Mini-batch training**: *m* examples (512, or 1024)

Overfitting

- A model that perfectly match the training data has a problem.
- It will also **overfit** to the data, modeling noise
  - A random feature value that perfectly predicts *y* (it happens to only occur in one class) will get a very high weight.
  - Failing to generalize to a test set without this feature value.
- A good model should be able to **generalize**

Overfitting

- Models that are too powerful can **overfit** the data
- Fitting the details of the training data so exactly that the model doesn't generalize well to the test set
  - How to avoid overfitting?
  - Regularization in logistic regression

Regularization

- A solution for overfitting
- Add a regularization term *R*(θ) to the loss function (for now written as maximizing logprob rather than minimizing loss)

$$\hat{\theta} \;=\; \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) - \alpha R(\theta)$$

where α is a hyper-parameter

- Idea: choose an *R*(θ) that penalizes large weights
  – fitting the data well with lots of big weights not as good as fitting the data a little less well, with small weights

L1 Regularization (= lasso regression)

- The sum of the (absolute value of the) weights
- Named after the **L1 norm** $||W||_1$, = sum of the absolute values of the weights, = **Manhattan distance**

$$R(\theta) \;=\; ||\theta||_1 = \sum_{i=1}^{n} |\theta_i|$$

- L1 regularized objective function:

$$\hat{\theta} \;=\; \operatorname*{argmax}_{\theta} \left[\sum_{1=i}^{m} \log P(y^{(i)}|x^{(i)}) \right] - \alpha \sum_{j=1}^{n} |\theta_j|$$

Logistic Regression
Example

- The subset of the Coronary Risk-Factor Study (CORIS) baseline survey, carried out in three rural areas of the Western Cape, South Africa
- Aim: establish the intensity of ischemic heart disease risk factors in that high-incidence region
- Response variable (class attribute) is the presence or absence of myocardial infraction (MI) at the time of survey
- 160 cases in data set, sample of 302 controls

Logistic Regression Example

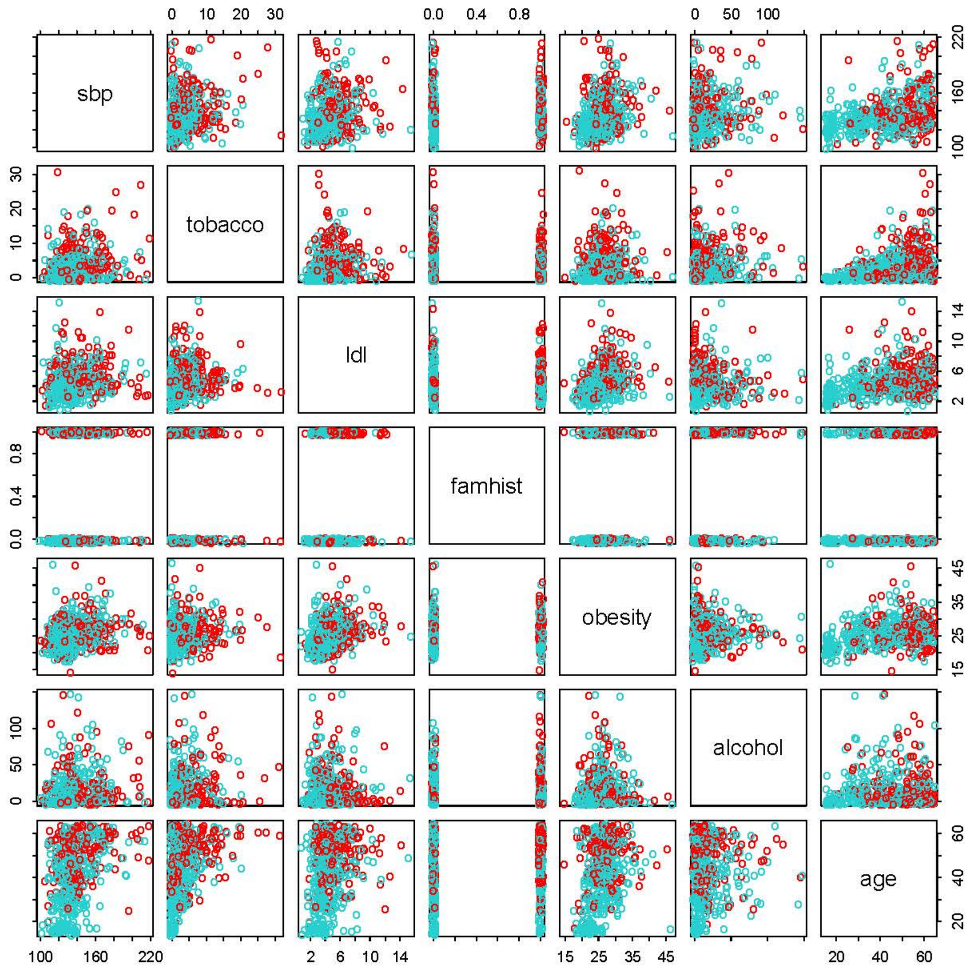


FIGURE 4.12. *A scatterplot matrix of the South African heart disease data. Each plot shows a pair of risk factors, and the cases and controls are color coded (red is a case). The variable* family history of heart disease (`famhist`) *is binary (yes or no).*

Logistic Regression

Example

- Fit a logistic-regression model by maximum likelihood, giving the results shown in the next slide
  - z scores for each coefficients in the model (coefficients divided by their standard errors)

Logistic Regression
Example

- Results from a logistic regression fit to the South African heart disease data:

	Coefficient	Std. Error	z Score
(Intercept)	-4.130	0.964	-4.285
sbp	0.006	0.006	1.023
tobacco	0.080	0.026	3.034
ldl	0.185	0.057	3.219
famhist	0.939	0.225	4.178
obesity	-0.035	0.029	-1.187
alcohol	0.001	0.004	0.136
age	0.043	0.010	4.184

Logistic Regression
Example

- z scores greater than approximately 2 in absolute value is significant at the 5% level
- Some surprises in the table of coefficients
  - sbp and obesity appear to be not significant
- *On their own*, both sbp and obesity are significant, with positive sign
- Presence of many other correlated variables → no longer needed (can even get a negative sign)