Linear models II

Classification

Q6

What is the gradient of $f(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2$ with respect to the parameter vector, $\mathbf{w} = [w_0, w_1, w_2]^\top$? In other words, what is $\nabla_{\mathbf{w}} f$?

- ◯ A. $\nabla_{\mathbf{w}} f = [1, 2x]^\top$
- ◯ B. $\nabla_{\mathbf{w}} f = [x, x^2]^\top$
- ◯ C. $\nabla_{\mathbf{w}} f = [1, x, x^2]^\top$
- ◯ D. $\nabla_{\mathbf{w}} f = [0, 1, 2x]^\top$
- ◯ E. $\nabla_{\mathbf{w}} f = [w_0, w_1, w_2]^\top$
- ◯ F. $\nabla_{\mathbf{w}} f = 1 + x + x^2$
- ◯ G. $\nabla_{\mathbf{w}} f = w_1 + 2w_2 x$
- ◯ H. $\nabla_{\mathbf{w}} f = x_1 + 2w_2 x$
- ◯ I. $\nabla_{\mathbf{w}} f = w_0$
- ◯ J. $\nabla_{\mathbf{w}} f = 2w_2 x$

$$\nabla_{w} f = \begin{bmatrix} \frac{\partial f}{\partial w_0} \\ \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix}$$

Q7

If $\mathbf{w} = [1, 2]^\top$ and $\mathbf{x} = [2, 4]^\top$, then what is $\mathbf{w}^\top \mathbf{x}$?

- ◯ A. 8
- ◯ B. 9
- ◯ C. 10
- ◯ D. 25
- ◯ E. $\begin{bmatrix} 2 & 4 \\ 4 & 8 \end{bmatrix}$
- ◯ F. $\begin{bmatrix} 8 & 4 \\ 4 & 2 \end{bmatrix}$
- ◯ G. $\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$
- ◯ H. $\begin{bmatrix} 2 & 4 \\ 1 & 2 \end{bmatrix}$

$$\boldsymbol{w} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

$$\boldsymbol{w}^\top \boldsymbol{x} = \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \end{bmatrix} = 2 + 8 = 10$$

How can we…

model nonlinear relationships using linear models?

use linear models for classification?

choose the parameters to fit a linear classification model to training data?

Can we model nonlinear relationships?

Linear models are linear in the parameters

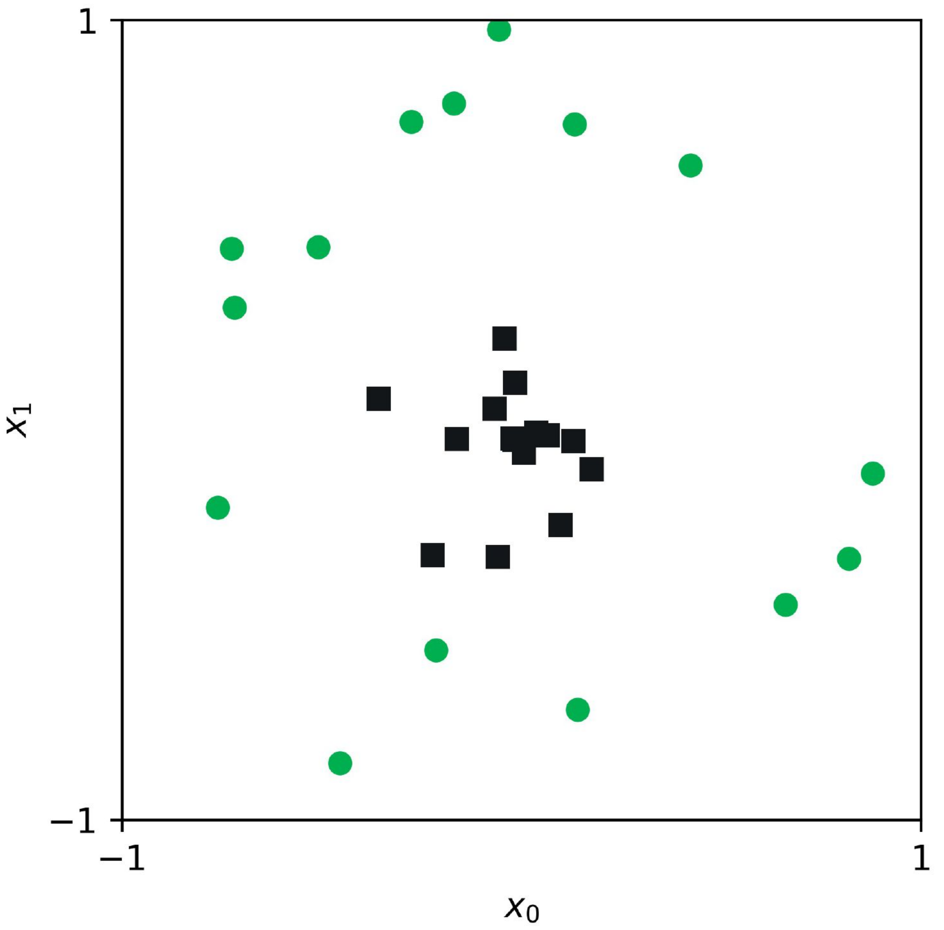
A linear combination is quantity where a set of terms are added together,
each multiplied by a constant (parameter) and adding the results

They can model **nonlinear relationships** between features and targets through **feature transformations**

Limitations of linear decision boundaries

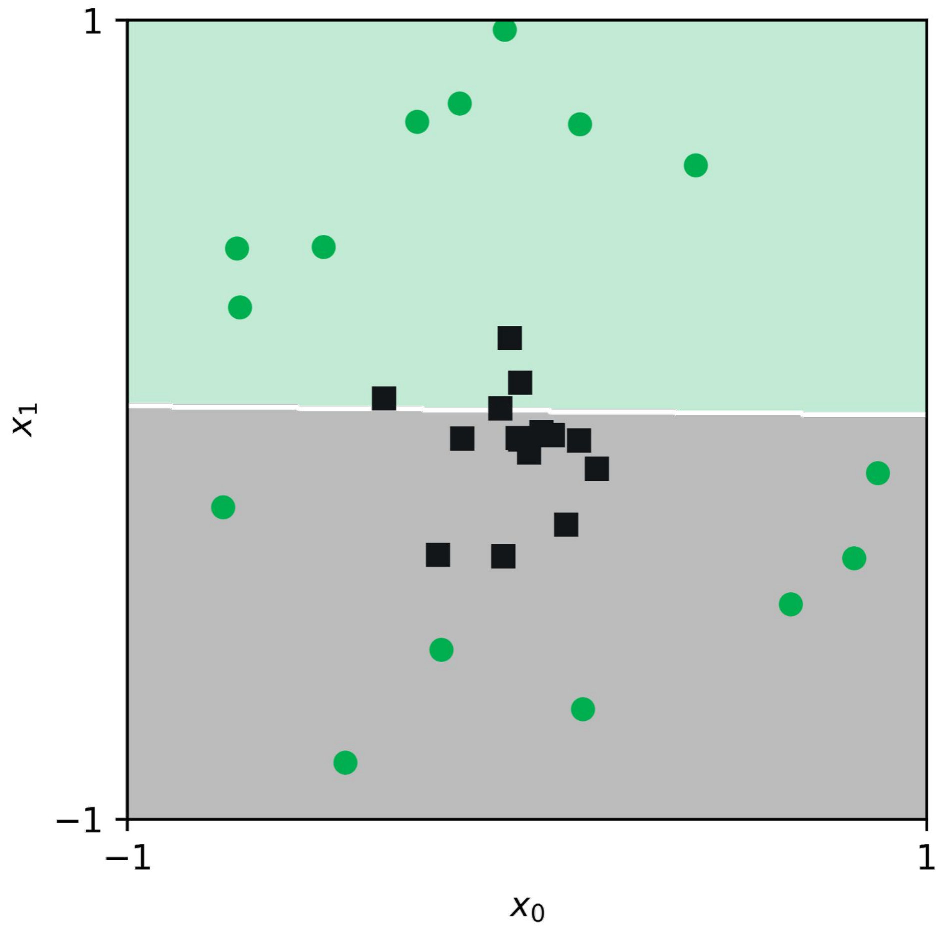
Original data

$\boldsymbol{x}$



Classify the features in this X-space

$$\hat{f}_{\boldsymbol{x}}(\boldsymbol{x}) = \begin{cases} 1 & \boldsymbol{w}^T\boldsymbol{x} > 0 \\ 0 & else \end{cases}$$



Transformations of features

Consider a digits example…

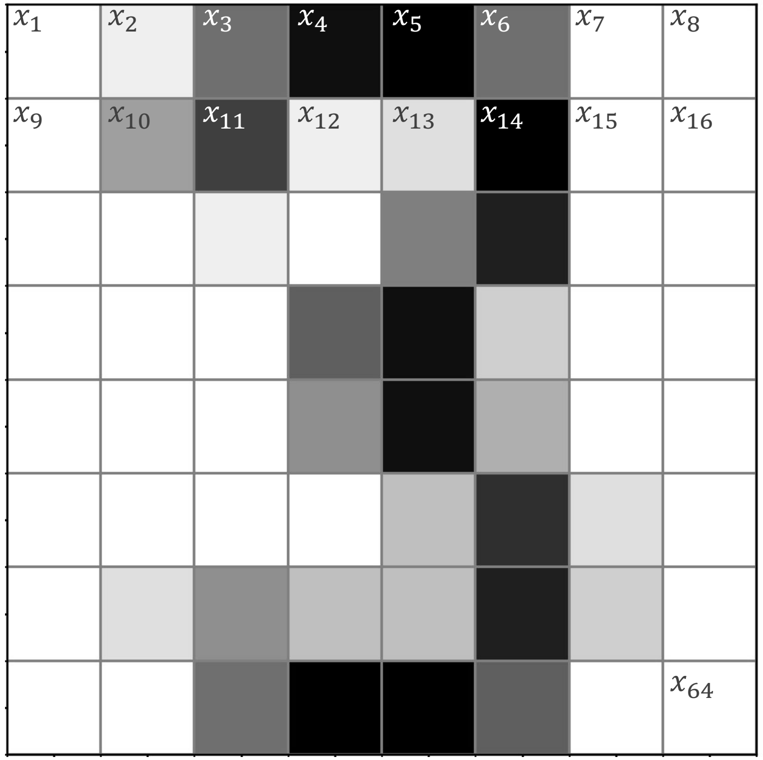
$$\boldsymbol{x} = [x_1, x_2, x_3, \dots, x_{64}]$$

We could **design features** based on the original features. For example:

$$\boldsymbol{z} = \left[x_5 x_{11}, x_{14}^2, \frac{x_{64}}{x_{14}}\right]$$

Which can be written simply as variables in a new feature space:

$$\boldsymbol{z} = [z_1, z_2, z_3]$$

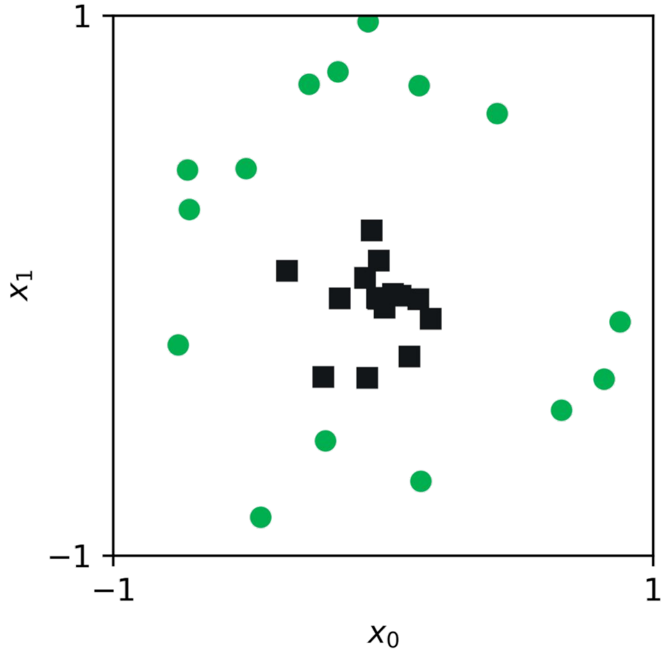


Source: Abu-Mostafa, Learning from Data, Caltech

1

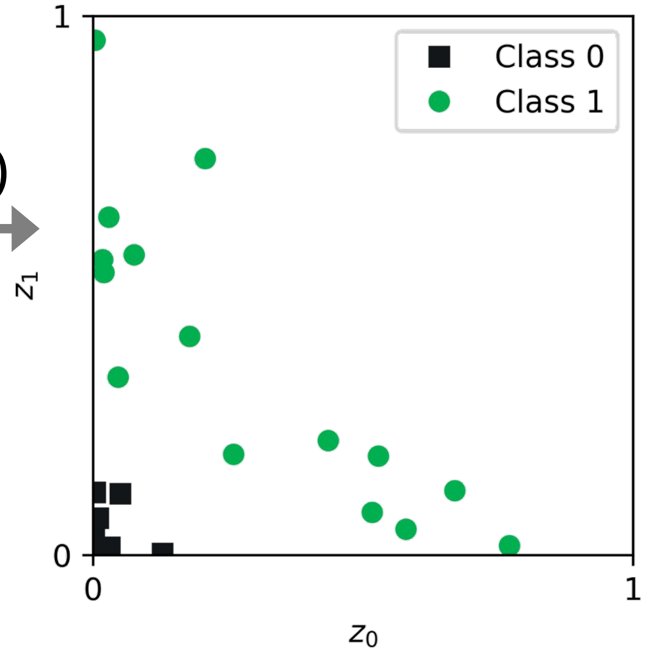
Original data

$\boldsymbol{x}$



transform the data

$\boldsymbol{z} = \Phi(\boldsymbol{x})$



2

This example transform is quadratic

$$z_i = \Phi(x_i) = x_i^2$$

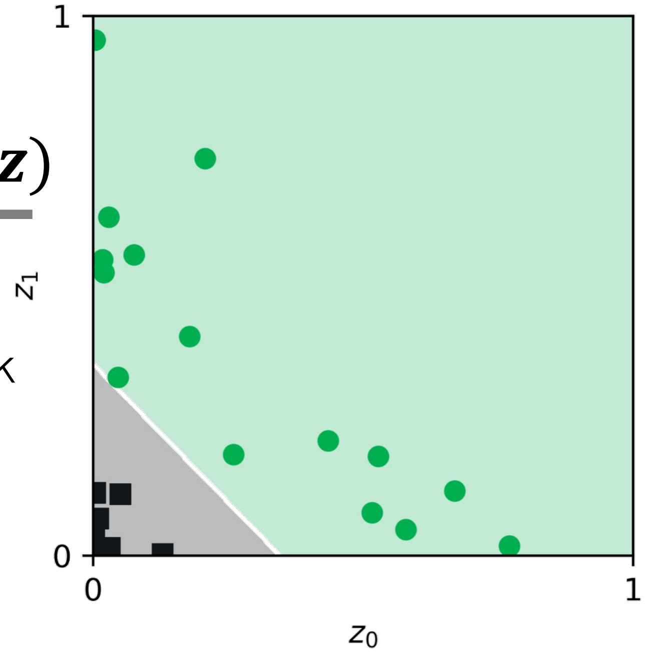
$$z_0 = x_0^2$$

$$z_1 = x_1^2$$

Classify the features in this Z-space

$$\hat{f}_z(\boldsymbol{z}) = \begin{cases} 1 & \boldsymbol{w}^T\boldsymbol{z} > 0 \\ 0 & else \end{cases}$$

A new **representation** of our data



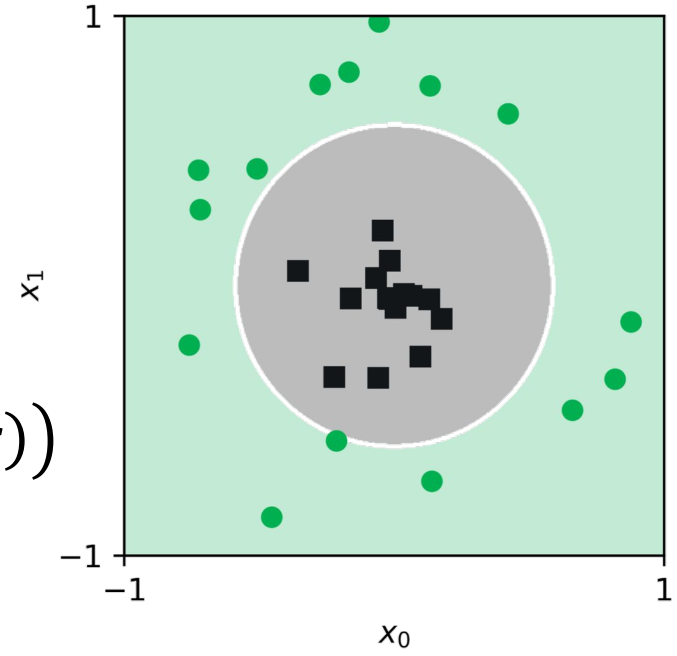
3

$\boldsymbol{x} = \Phi^{-1}(\boldsymbol{z})$

transform the data back

$$x_0 = z_0^{1/2}$$

$$x_1 = z_1^{1/2}$$



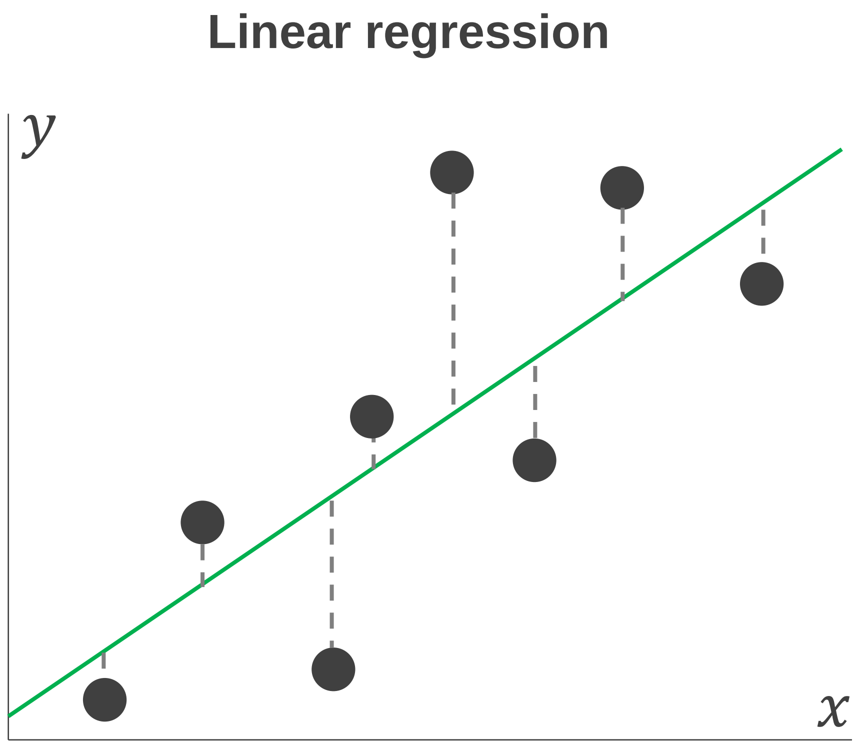
Predictions in the original X-space

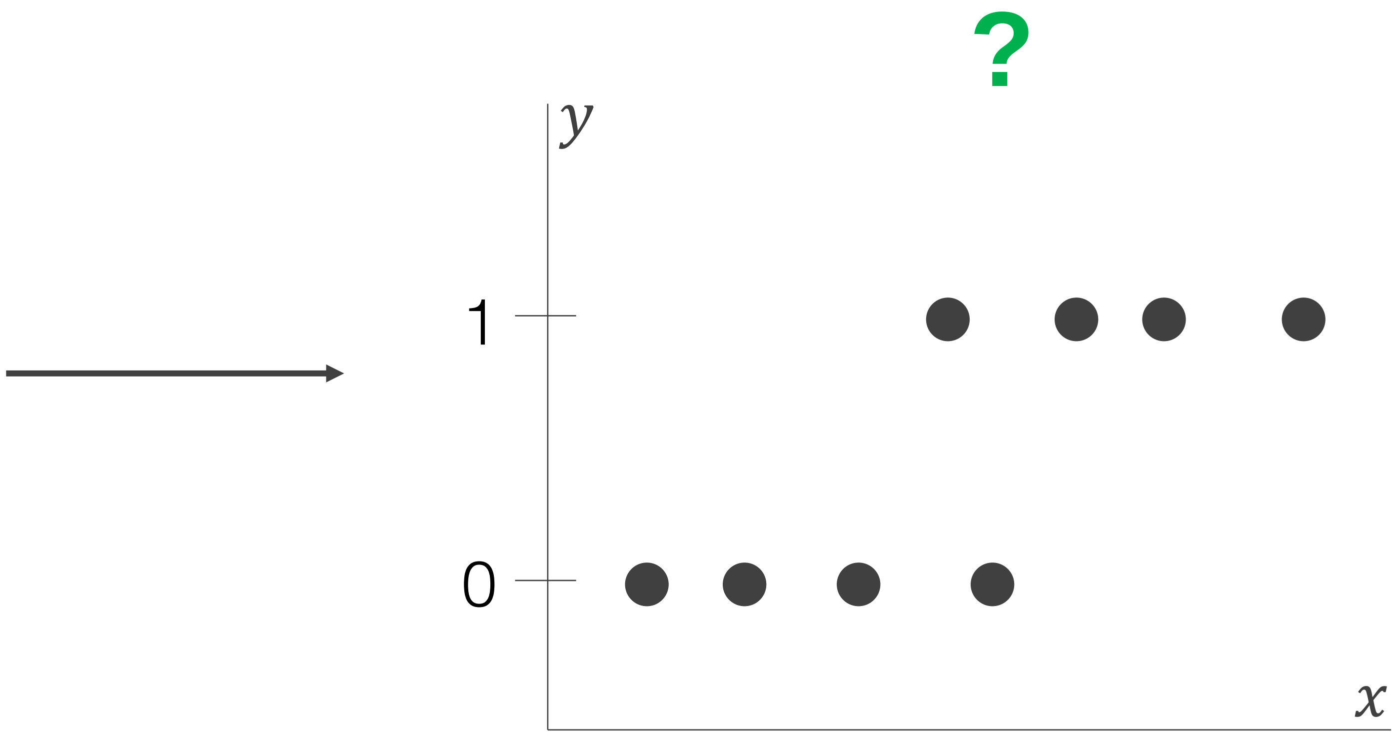
$$\hat{f}(\boldsymbol{x}) = \hat{f}_z\big(\Phi(\boldsymbol{x})\big)$$

4

So how do we use linear models for classification?

How do we fit linear models for classification?





Moving from regression to classification

Regression

$$y = \sum_{i=0}^{p} w_i x_i$$

Classification
(perceptron)

$$y = \ sign\left(\sum_{i=0}^{p} w_i x_i\right)$$

$$y = \begin{cases} 1 & \sum_{i=0}^{p} w_i x_i > 0 \\ -1 & else \end{cases}$$

where

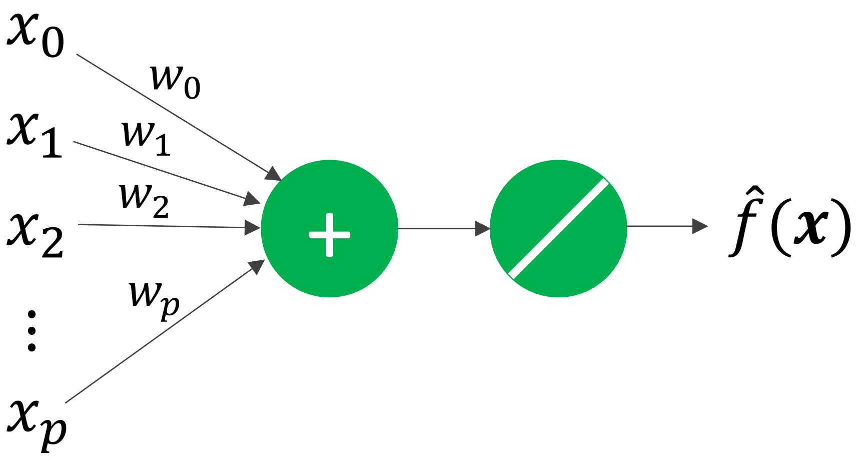
$$sign(x) = \begin{cases} 1 & x > 0 \\ -1 & else \end{cases}$$

Source: Abu-Mostafa, Learning from Data, Caltech

Moving from regression to classification

Linear Regression

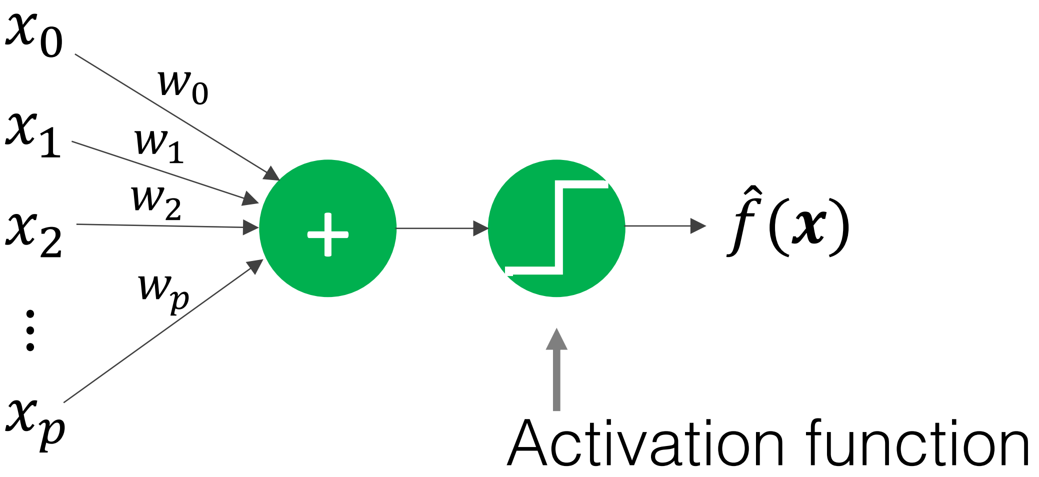
$$\hat{f}(\boldsymbol{x}) = \sum_{i=0}^{p} w_i x_i$$



Linear Classification

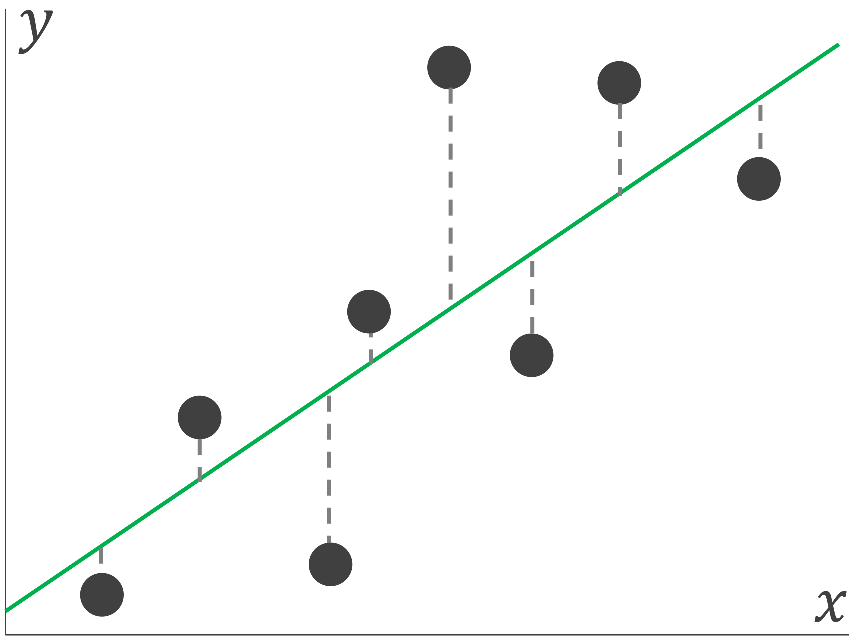
(perceptron)

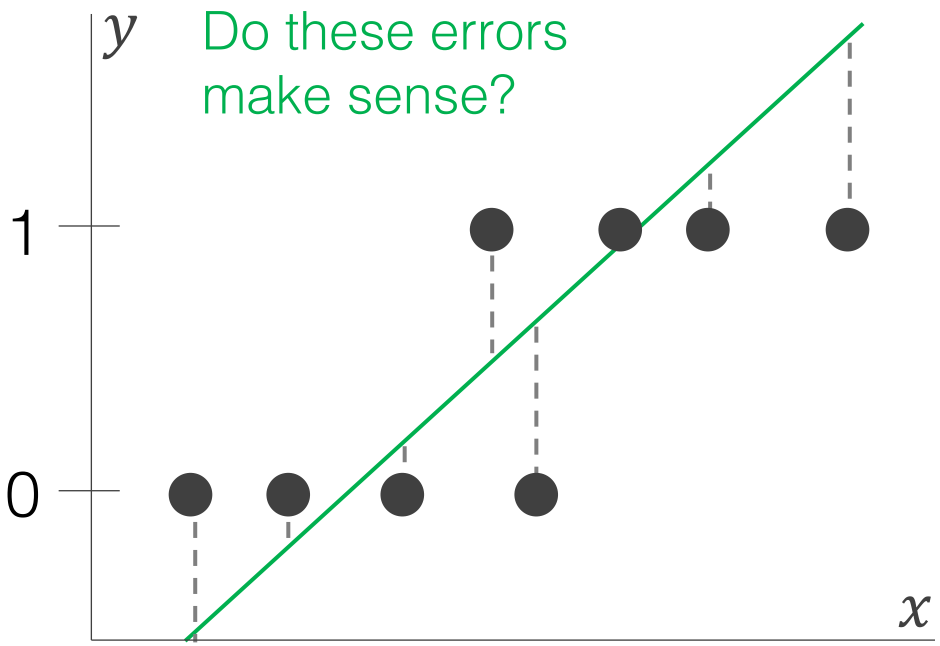
$$\hat{f}(\boldsymbol{x}) = sign\left(\sum_{i=0}^{p} w_i x_i\right)$$



Source: Abu-Mostafa, Learning from Data, Caltech

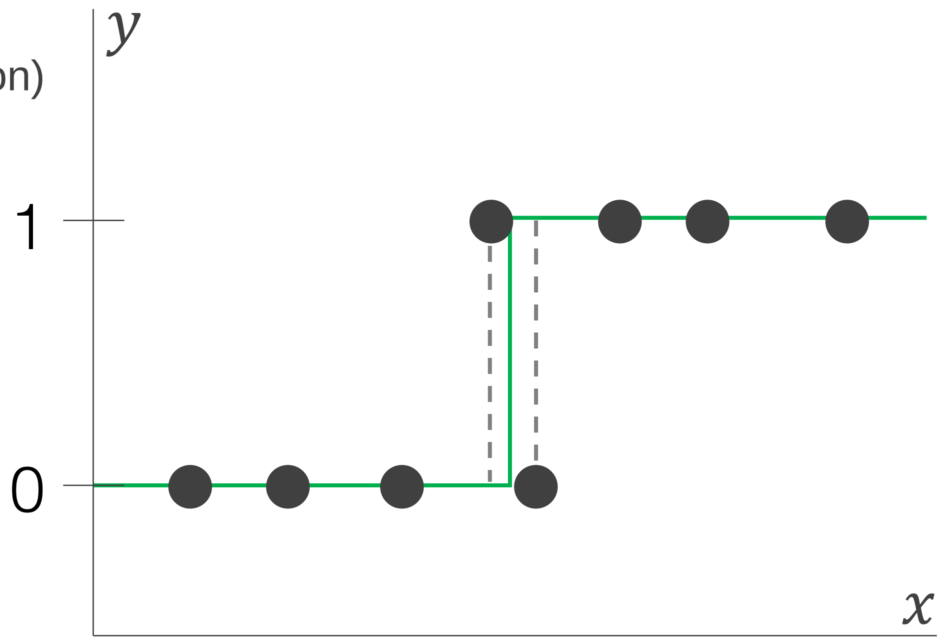
Linear regression
(linear activation)

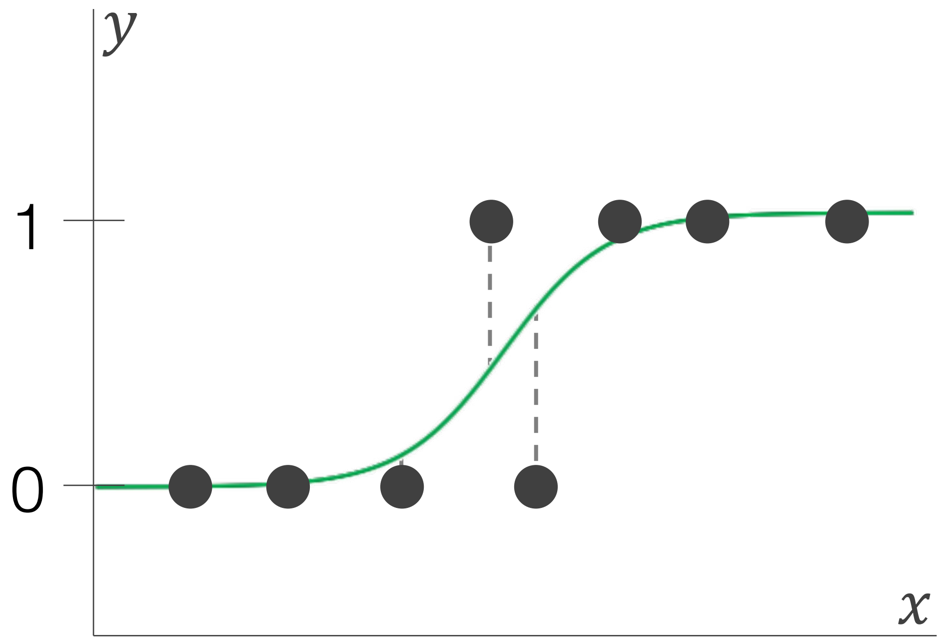




Linear regression
applied to a classification problem
(linear activation)

Perceptron
(sign activation)





Logistic regression
(sigmoid activation)

Sigmoid function

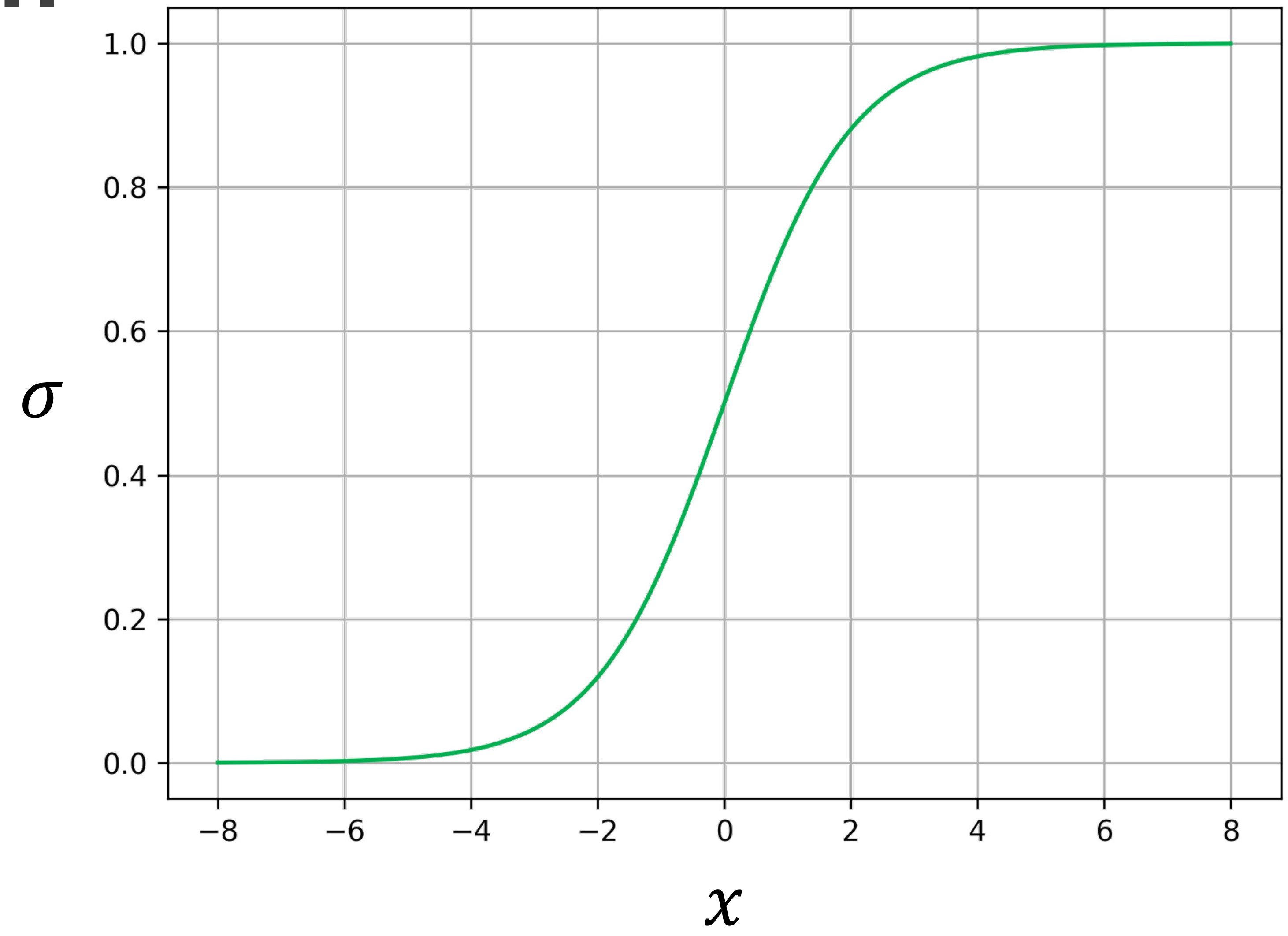
Definition

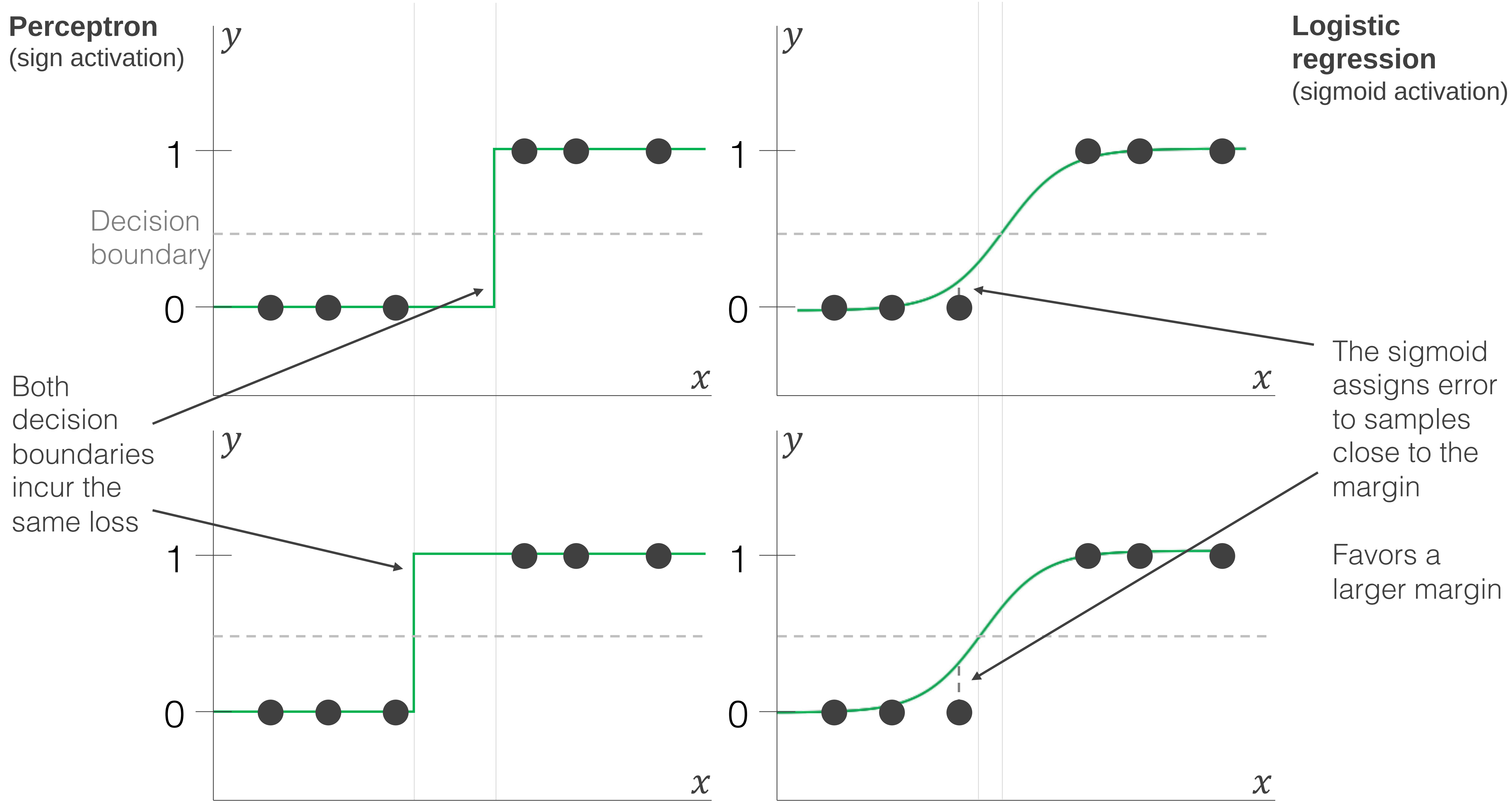
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Useful properties

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

$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))$$

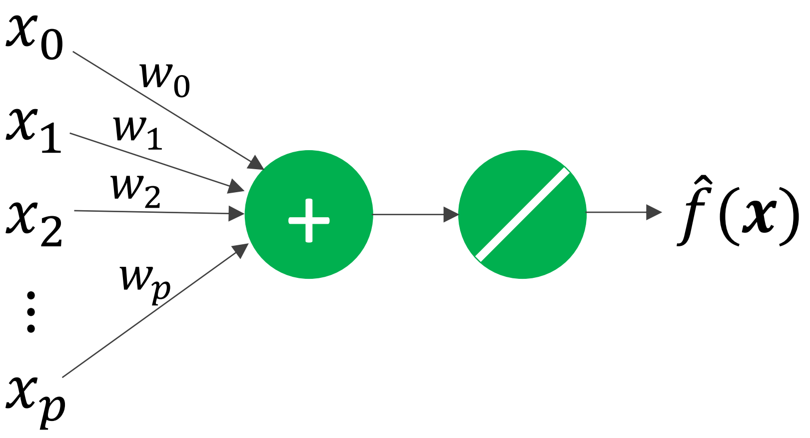



Perceptron
(sign activation)
Logistic regression
(sigmoid activation)
y
x
1
0
Decision boundary
Both decision boundaries incur the same loss
The sigmoid assigns error to samples close to the margin
Favors a larger margin

Moving from regression to classification

Linear Regression

$$\hat{f}(\boldsymbol{x}) = \sum_{i=0}^{p} w_i x_i$$

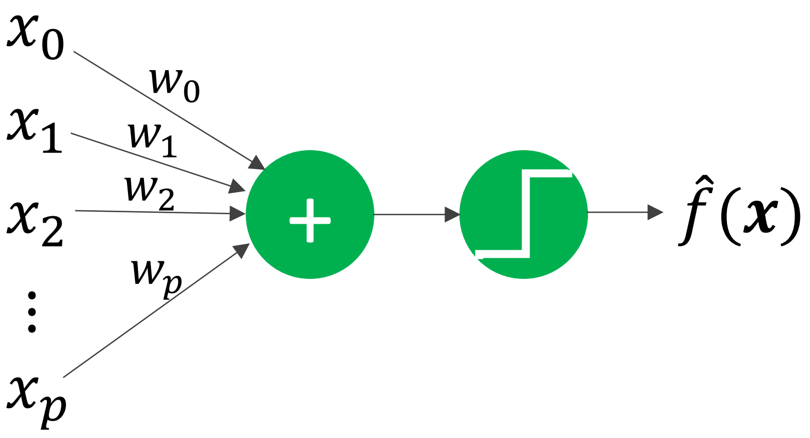


Linear Classification

Perceptron

$$\hat{f}(\boldsymbol{x}) = sign\left(\sum_{i=0}^{p} w_i x_i\right)$$

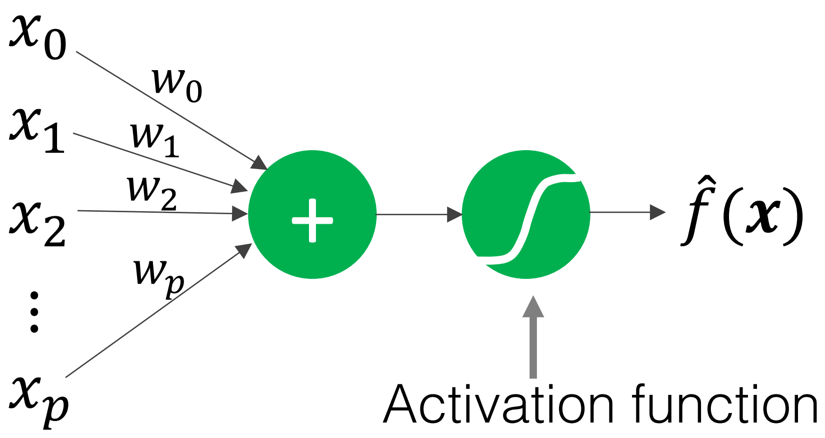
$$sign(x) = \begin{cases} 1 & x > 0 \\ -1 & \text{else} \end{cases}$$



Logistic Regression

$$\hat{f}(\boldsymbol{x}) = \sigma\left(\sum_{i=0}^{p} w_i x_i\right)$$

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



Source: Abu-Mostafa, Learning from Data, Caltech

We fit our model to training data

1. Choose a **hypothesis set of models** to train

2. Identify a **cost function** to measure the model fit to the training data

3. **Optimize** model **parameters** to minimize cost

For linear regression the steps were (i.e. OLS):

a. Calculate the gradient of the cost function
b. Set the gradient to zero
c. Solve for the model parameters

When this approach is not an option, we often use **gradient descent**

For classification we COULD try the same cost function as regression

Assume the cost function is **mean square error**

$$C(\boldsymbol{w}) \triangleq E_{in}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\boldsymbol{x}_n, \boldsymbol{w}) - y_n\right)^2$$

$$\hat{f}(\boldsymbol{x}_n, \boldsymbol{w}) = \sigma(\boldsymbol{w}^T \boldsymbol{x}_n)$$

Plug in our model

$$C(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(\sigma(\boldsymbol{w}^T \boldsymbol{x}_n) - y_n)^2$$

Calculate the gradient

$$\nabla_w C(\boldsymbol{w}) = \frac{2}{N}\sum_{n=1}^{N}[\sigma(\boldsymbol{w}^T \boldsymbol{x}_n) - y_n]\sigma(\boldsymbol{w}^T \boldsymbol{x}_n)[\boldsymbol{1} - \sigma(\boldsymbol{w}^T \boldsymbol{x}_n)]\boldsymbol{x}_n$$

Set the gradient to zero and minimize to solve for $\boldsymbol{w}$

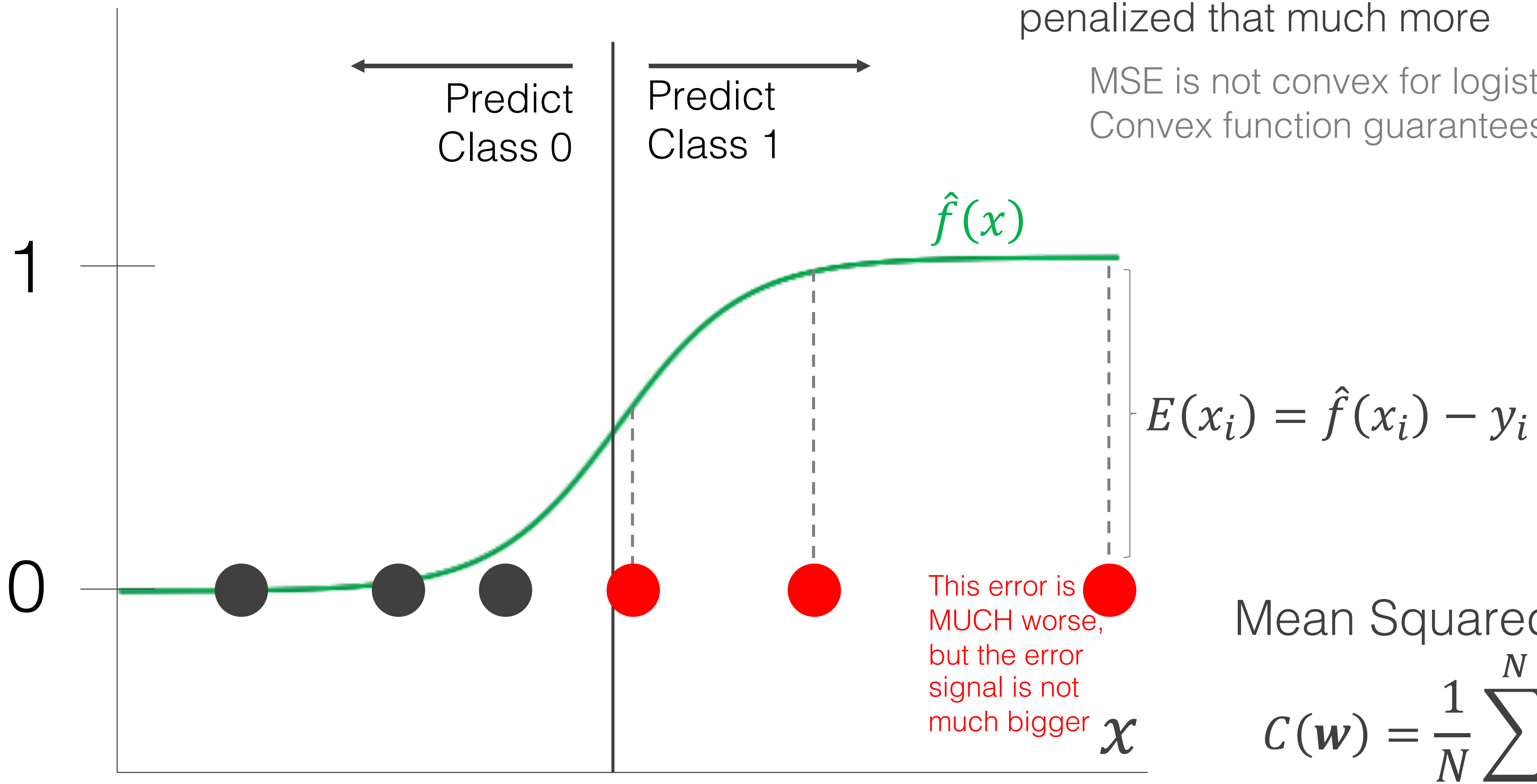
$$\nabla_w C(\boldsymbol{w}) = \boldsymbol{0}$$

But does MSE make sense for classification?

MSE for classification

Intuition: With a mean squared error cost function, bigger classification mistakes are not penalized that much more

MSE is not convex for logistic regression
Convex function guarantees a global minimum



Mean Squared Error Cost:

$$C(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\mathbf{x}_n, \mathbf{w}) - y_n\right)^2$$

We need a different cost function for logistic regression...

Is there a better cost function we could use for classification problems…?

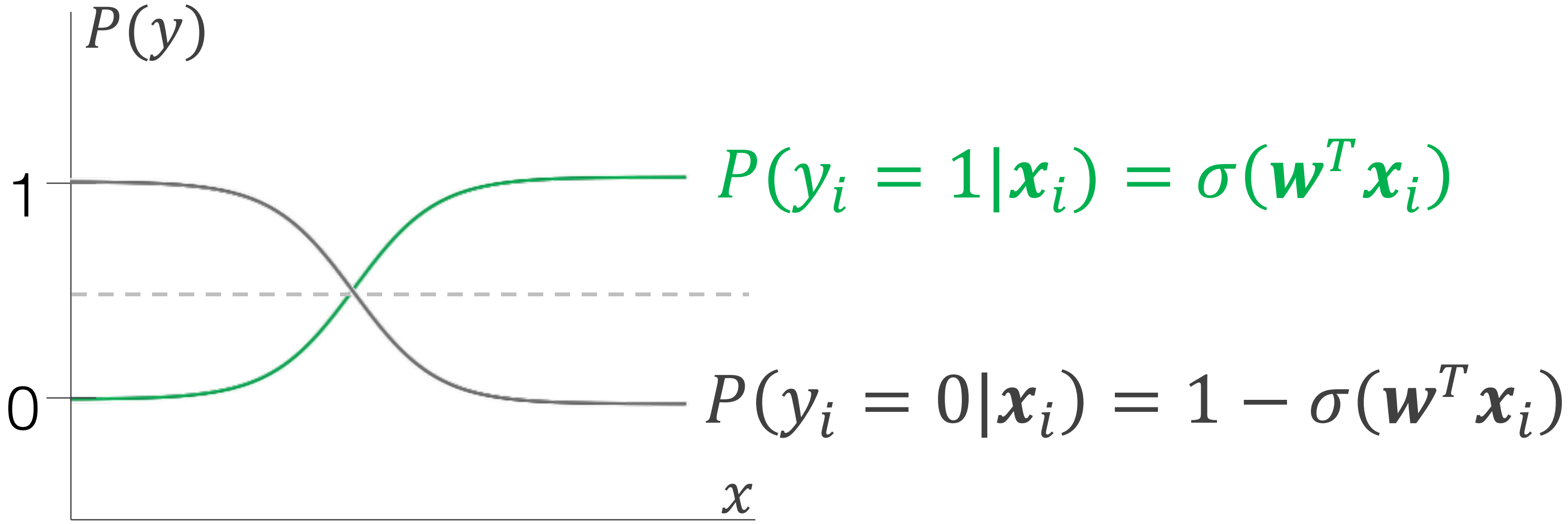
Another interpretation of logistic regression

Our model: $\hat{y} = \hat{f}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T \boldsymbol{x})$

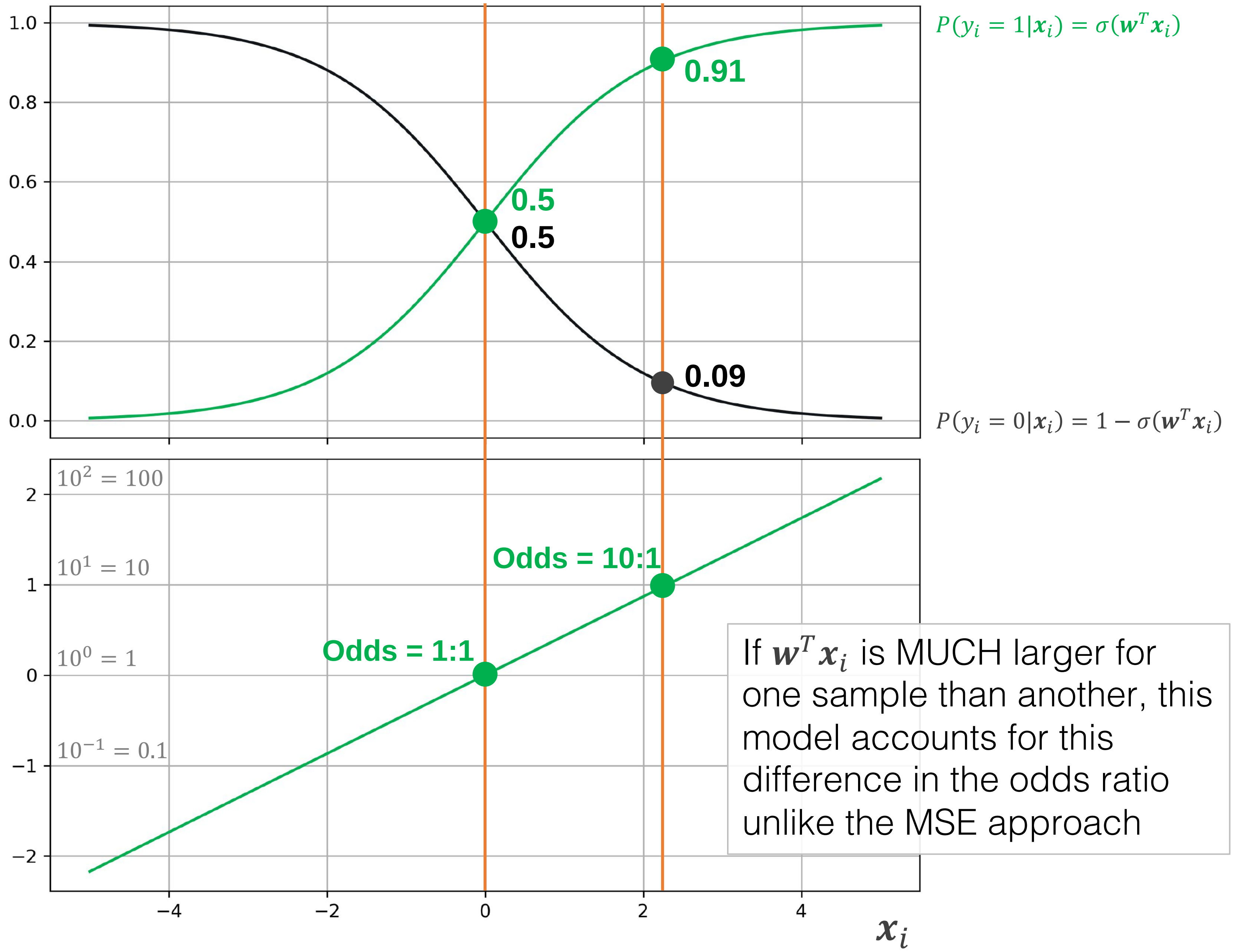
$$\sigma(\boldsymbol{w}^T \boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{w}^T \boldsymbol{x}}}$$

Logistic regression models the

conditional probability that a sample belongs to a class



Interpretation of the odds ratio



Log-odds ratio: ratio of positive class probability to the negative class

$$\log \left[\frac{P(y_i = 1)}{P(y_i = 0)} \right]$$

If $\boldsymbol{w}^T \boldsymbol{x}_i$ is MUCH larger for one sample than another, this model accounts for this difference in the odds ratio unlike the MSE approach

The log-odds ratio is a linear function of the features

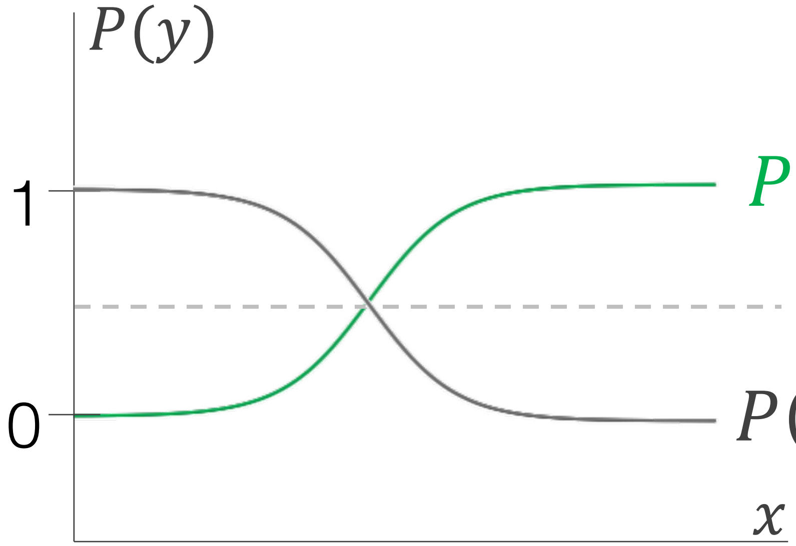
So how do we fit our model to the data in this case?

Our model: $\hat{y} = \hat{f}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T \boldsymbol{x})$

$$\sigma(\boldsymbol{w}^T \boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{w}^T \boldsymbol{x}}}$$

Logistic regression models the

conditional probability that a sample belongs to a class



$P(y_i = 1 | \boldsymbol{x}_i) = \sigma(\boldsymbol{w}^T \boldsymbol{x}_i)$

$P(y_i = 0 | \boldsymbol{x}_i) = 1 - \sigma(\boldsymbol{w}^T \boldsymbol{x}_i)$

Sidebar: Maximum Likelihood Estimation

We want to determine the underlying probability of the coin landing on “heads”; the coin could be biased.

We flip the coin 1,000 times

…in other words, we have $N = 1{,}000$ **independent** Bernoulli trials

Coin flips, binary outcomes

$$P(X = 1) = p$$
$$P(X = 0) = 1 - p$$

Goal: find the value of p that maximizes the likelihood function

Interpretation of likelihood: a function of a parameter we want optimize for, given our data: $L(p|x)$

Goal: find the value of p that maximizes the likelihood of our data

$$P(X=1)=p$$
$$P(X=0)=1-p$$

For a **single observation**, the likelihood is:

$$L(p|x_i)=P(x_i|p)=p^{x_i}(1-p)^{1-x_i}$$

For a **multiple independent observations**, the likelihood is:

For independent random events, the probability of both events is the product of their individual probabilities:
$P(A \text{ and } B)=P(A)P(B)$

$$L(p|\boldsymbol{x})=P(\boldsymbol{x}|p)=\prod_{i=1}^{N}P(x_i|p)$$

$$=p^{\sum_{i=1}^{N}x_i}(1-p)^{N-\sum_{i=1}^{N}x_i}$$

Goal: find the value of p that maximizes the likelihood of our data

$$L(p) = p^{\sum x_i}(1-p)^{N-\sum x_i}$$

Here, $L(p)$ is short for $L(p|\boldsymbol{x})$

Maximizing the likelihood is equivalent to maximizing the log-likelihood

$$\ln[L(p)] = \ln[p^{\sum x_i}(1-p)^{N-\sum x_i}]$$

$$\ln[L(p)] = \ln(p)\sum_{i=1}^{N} x_i + \ln(1-p)\left[N - \sum_{i=1}^{N} x_i\right]$$

To **maximize the likelihood**, we take the **derivative of this log likelihood and set it to zero**, then solve for p

Goal: find the value of p that maximizes the likelihood of our data

We take the derivative of this log likelihood and set it to zero, then solve for p

$$\ln[L(p)] = \ln(p) \sum_{i=1}^{N} x_i + \ln(1-p) \left[N - \sum_{i=1}^{N} x_i \right]$$

$$\frac{\partial \ln[L(p)]}{\partial p} = \frac{\sum_{i=1}^{N} x_i}{p} - \frac{N - \sum_{i=1}^{N} x_i}{1-p} = 0$$

This results in our estimate being the mean of our observations:

$$\hat{p} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

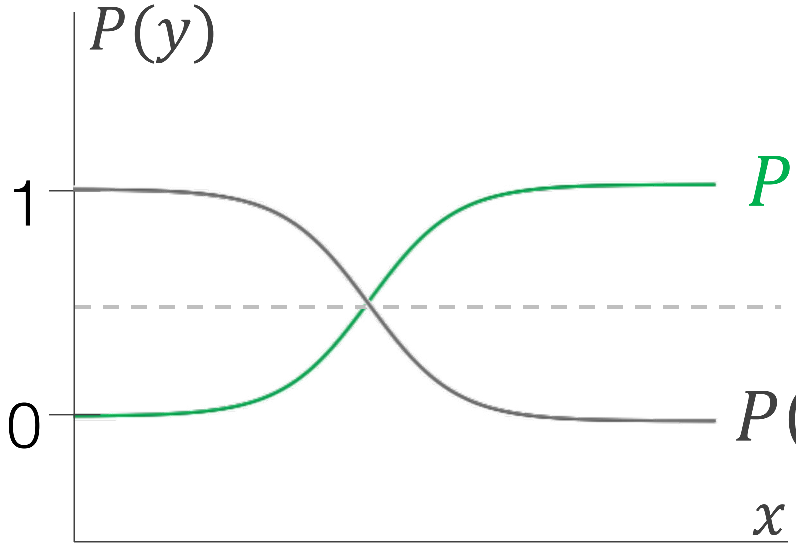
Applying this to logistic regression…

Our model: $\hat{y} = \hat{f}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x})$

$$\sigma(\boldsymbol{w}^T\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{w}^T\boldsymbol{x}}}$$

Logistic regression models the

conditional probability that a sample belongs to a class



$P(y_i = 1|\boldsymbol{x}_i) = \sigma(\boldsymbol{w}^T\boldsymbol{x}_i)$

$P(y_i = 0|\boldsymbol{x}_i) = 1 - \sigma(\boldsymbol{w}^T\boldsymbol{x}_i)$

Note these are both functions of our parameters, $\boldsymbol{w}$

The interpretation of the Likelihood

With class labels $y_1, y_2, \ldots, y_N$ and corresponding to $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$

The likelihood for **one observation**:

$$L(\boldsymbol{w}|y_i, \boldsymbol{x}_i) = P(y_i = 1|\boldsymbol{x}_i)^{y_i} P(y_i = 0|\boldsymbol{x}_i)^{1-y_i}$$

We're interested in the likelihood of the model as a function of the model parameters, $\boldsymbol{w}$. So $P(y_i|\boldsymbol{x}_i)$ is a function of $\boldsymbol{w}$.

$$L(\boldsymbol{w}) \triangleq P(\boldsymbol{y}|\boldsymbol{X})$$

The likelihood for **all observations**:

$$L(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{X}) = P(y_1, y_2, \ldots, y_N|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N) = \prod_{i=1}^{N} P(y_i|\boldsymbol{x}_i)$$

Source: Malik Magdon-Ismail, Learning from Data

The likelihood for all observations:

$$L(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{X}) = \prod_{i=1}^{N} P(y_i|\boldsymbol{x}_i) = \prod_{i=1}^{N} P(y_i = 1|\boldsymbol{x}_i)^{y_i} P(y_i = 0|\boldsymbol{x}_i)^{1-y_i}$$

Substituting:

$$P(y_i = 1|\boldsymbol{x}_i) = \sigma(\boldsymbol{w}^T \boldsymbol{x}_i)$$
$$P(y_i = 0|\boldsymbol{x}_i) = 1 - \sigma(\boldsymbol{w}^T \boldsymbol{x}_i)$$

$$= \prod_{i=1}^{N} \sigma(\boldsymbol{w}^T \boldsymbol{x}_i)^{y_i} [1 - \sigma(\boldsymbol{w}^T \boldsymbol{x}_i)]^{1-y_i}$$

We want to **MAXIMIZE the likelihood (minimize it's negative)**

We can take the **logarithm**, negate it to get our **cost function**, then minimize it (using the gradient)

A little algebra

$$L(\boldsymbol{w}|\boldsymbol{y},\boldsymbol{X}) = \prod_{i=1}^{N} \sigma(\boldsymbol{w}^T\boldsymbol{x}_i)^{y_i}[1-\sigma(\boldsymbol{w}^T\boldsymbol{x}_i)]^{1-y_i}$$

$$= \prod_{i=1}^{N} \hat{y}_i{}^{y_i}[1-\hat{y}_i]^{1-y_i}$$

assuming $\hat{y}_i \triangleq \sigma(\boldsymbol{w}^T\boldsymbol{x}_i)$

If we take the log of both sides:

$$\log L(\boldsymbol{w}|\boldsymbol{y},\boldsymbol{X}) = \log\left[\prod_{i=1}^{N} \hat{y}_i{}^{y_i}[1-\hat{y}_i]^{1-y_i}\right] = \sum_{i=1}^{N} \log(\hat{y}_i{}^{y_i}[1-\hat{y}_i]^{1-y_i})$$

$$= \sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)$$

Recall that
$\log(ab) = \log(a) + \log(b)$
$\log(a^b) = b\log(a)$

$$\log L(\boldsymbol{w}|\boldsymbol{y},\boldsymbol{X}) = \sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)$$

We can define our
cost function: $C(\boldsymbol{w}) = -\log L(\boldsymbol{w}|\boldsymbol{y},\boldsymbol{X})$

$$\boxed{C(\boldsymbol{w}) = -\left[\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]}$$

This can be normalized by dividing by N for interpreting the results as **mean cost per sample**

For logistic regression,

$$\hat{y}_i \triangleq \sigma(\boldsymbol{w}^T \boldsymbol{x}_i)$$

This is the cross entropy cost function

Cross Entropy

$$C(\boldsymbol{w}) = -\frac{1}{N}\left[\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right]$$

Per sample cost: $-y_i \log(\hat{y}_i) - (1 - y_i)\log(1 - \hat{y}_i)$

When $y_i = 0$

$$-\underset{0}{y_i} \log(\hat{y}_i) - (\underset{1}{1 - y_i})\log(1 - \hat{y}_i)$$

The cost is: $-\log(1 - \hat{y}_i)$

When $y_i = 1$

$$-\underset{1}{y_i} \log(\hat{y}_i) - (\underset{0}{1 - y_i})\log(1 - \hat{y}_i)$$

The cost is: $-\log(\hat{y}_i)$

Cross Entropy

$$C(\boldsymbol{w}) = -y_i \log(\hat{y}_i) - (1 - y_i)\log(1 - \hat{y}_i)$$

Single sample cost

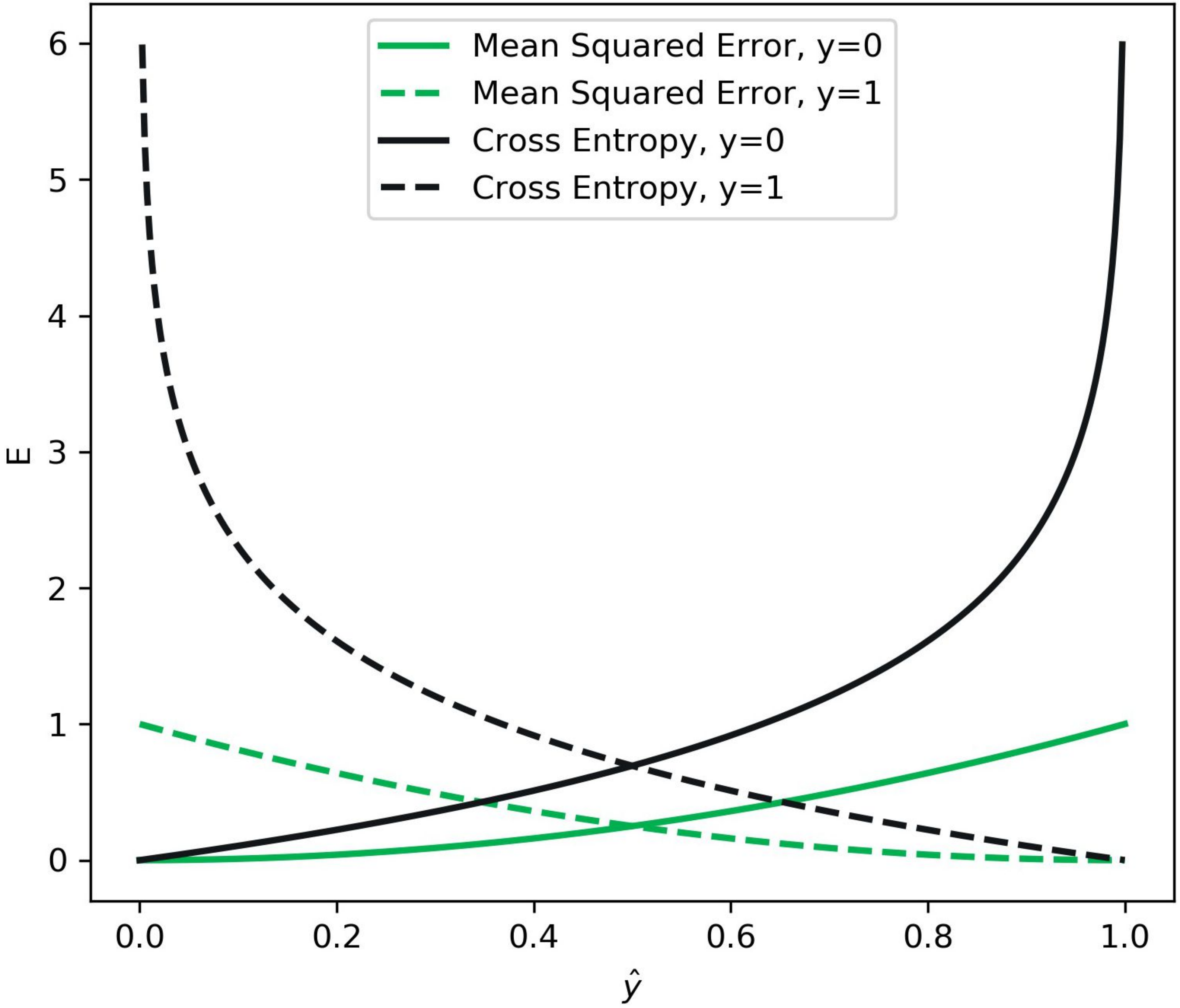
Progressively worse predictions →

Target label	CE Cost			
$y_i = 1$	$-\log(\hat{y}_i)$	$\hat{y}_i = 0.4$ $C(\boldsymbol{w}) = 0.91$	$\hat{y}_i = 0.1$ $C(\boldsymbol{w}) = 2.3$	$\hat{y}_i = 0.001$ $C(\boldsymbol{w}) = 6.9$
$y_i = 0$	$-\log(1 - \hat{y}_i)$	$\hat{y}_i = 0.6$ $C(\boldsymbol{w}) = 0.91$	$\hat{y}_i = 0.9$ $C(\boldsymbol{w}) = 2.3$	$\hat{y}_i = 0.999$ $C(\boldsymbol{w}) = 6.9$

Cross Entropy vs MSE

If a model is wrong, but is highly confident, it faces exponentially larger penalties with cross-entropy

Cross-entropy as a loss function provides a stronger error penalty for incorrect predictions



Logistic regression does not have a closed-form solution like linear regression did

We need a new approach to optimize the parameters…

Gradient descent

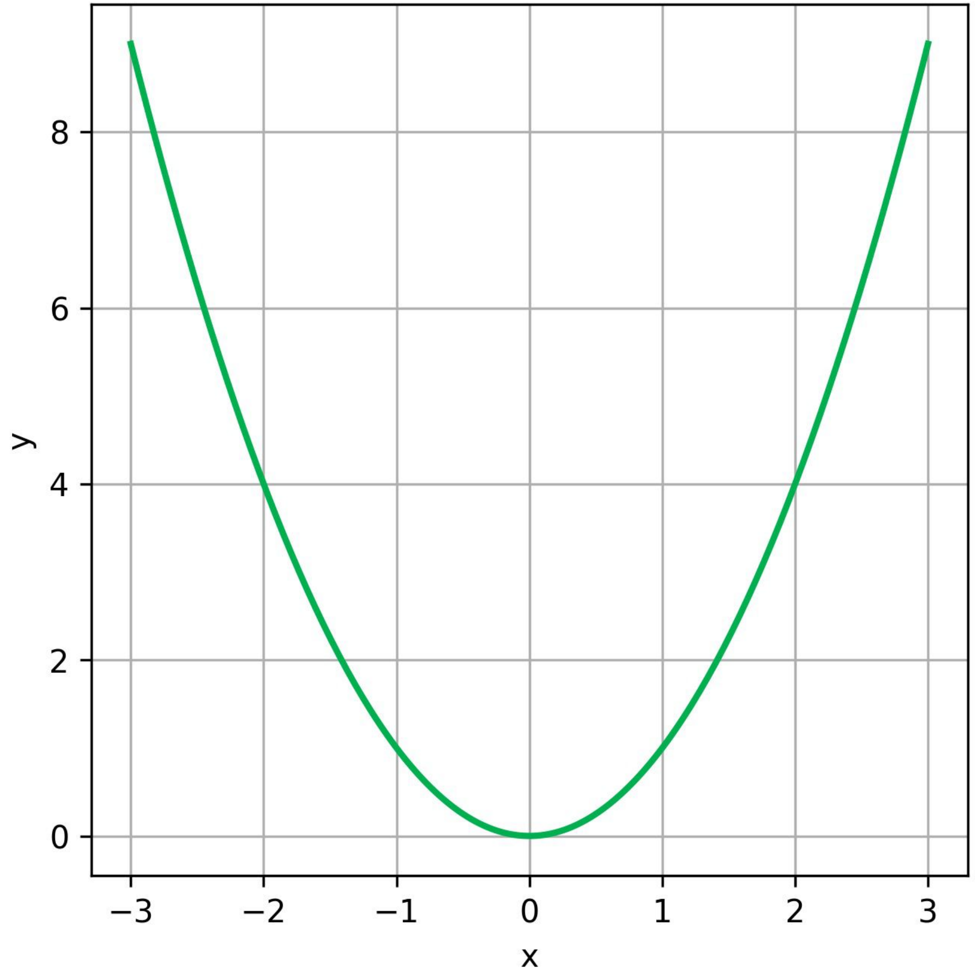
Minimize $y = x^2$

We start at an initial point and want to "roll" down to the minimum

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + \eta \boldsymbol{v}$$

Learning rate

Direction to move in



Gradient descent

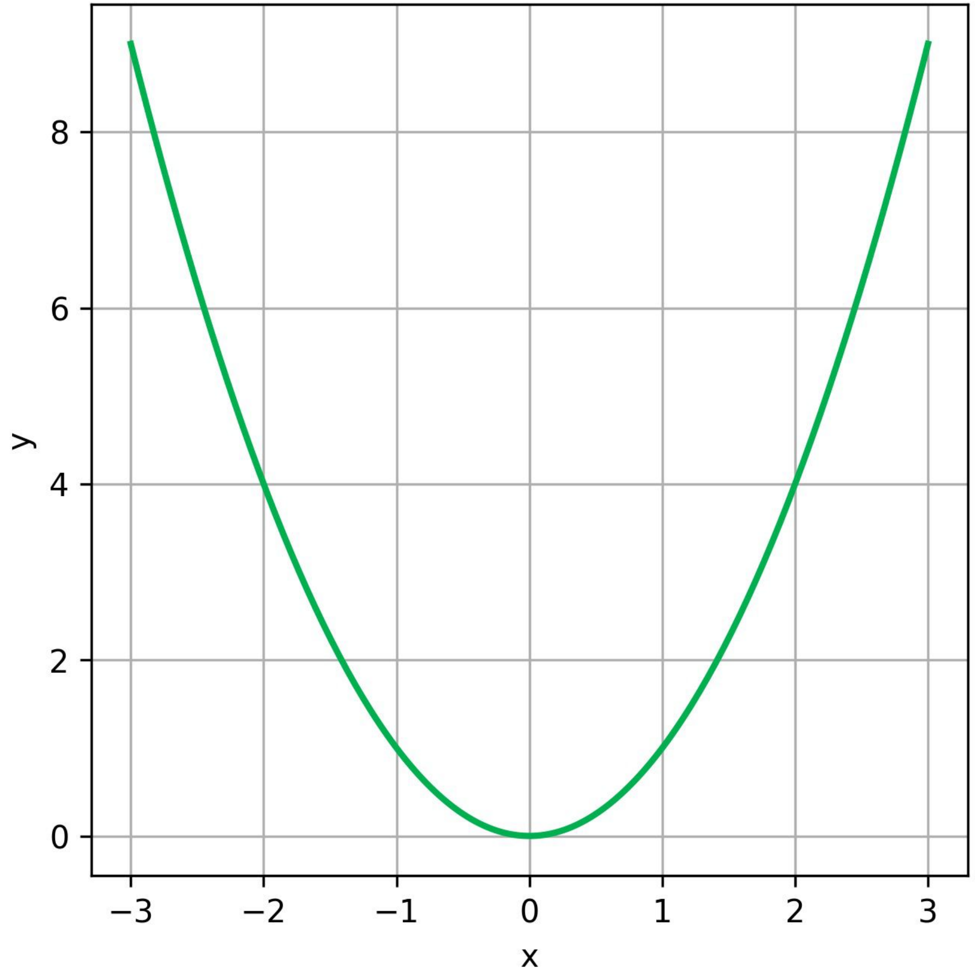
Minimize $f(x) = x^2$

The gradient points in the direction of steepest **positive** change

$$\frac{df(x)}{dx} = 2x$$

We want to move in the **opposite** direction of the gradient

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} - \eta \nabla f\left(\boldsymbol{x}^{(i)}\right)$$



Gradient descent

Derivative:

$$\frac{df(x)}{dx} = 2x$$

Gradient descent update equation:

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} - \eta \nabla f(\boldsymbol{x}^{(i)})$$

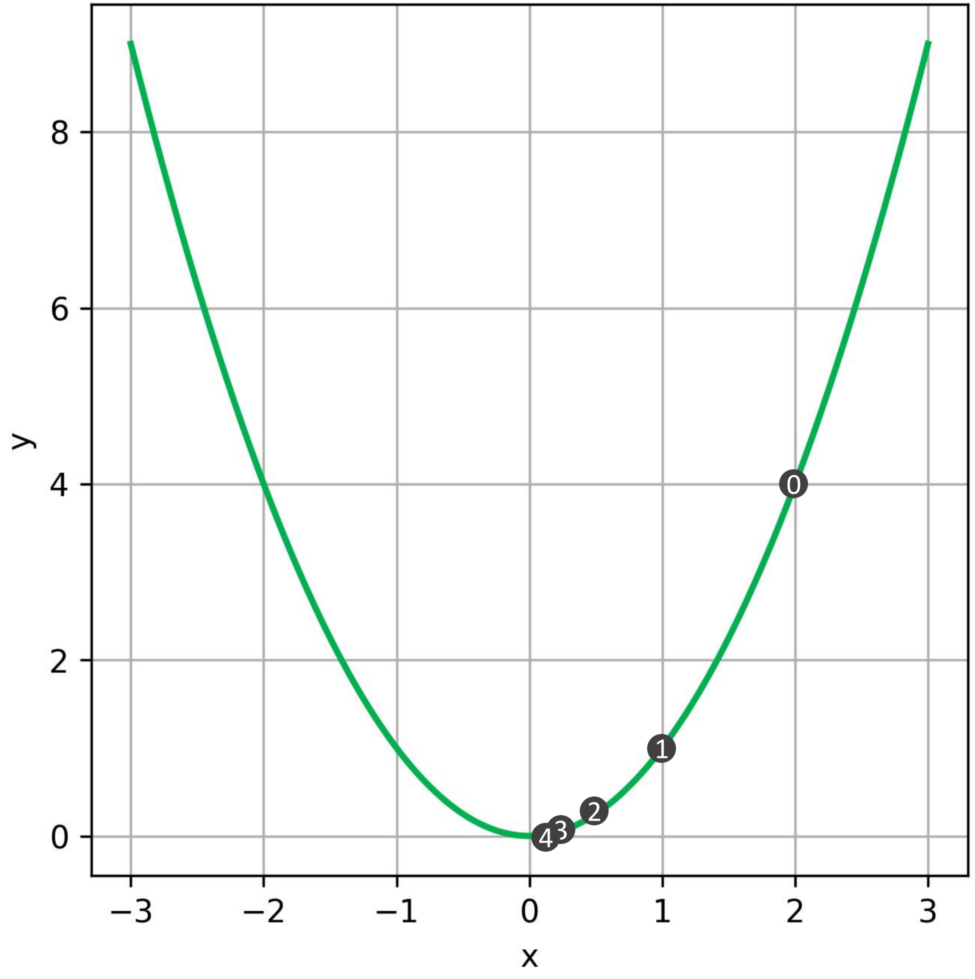
Minimize $f(x) = x^2$

Assume $x^{(0)} = 2$ and $\eta = 0.25$

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} - (0.25)(2\boldsymbol{x}^{(i)})$$

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} - (0.5)\boldsymbol{x}^{(i)}$$

i	$x^{(i)}$	$y^{(i)}$
0	2	4
1	1	1
2	0.5	0.25
3	0.25	0.0625
4	0.125	0.0156



Takeaways

Transformations of features (**feature extraction**) may help to overcome nonlinearities

Logistic regression is suited for classification

For classification problems, we typically apply **cross entropy loss** as the cost function

Logistic regression parameters required a different optimization strategy than OLS; one method for that optimization is **gradient descent**