

Linear models II

Classification

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

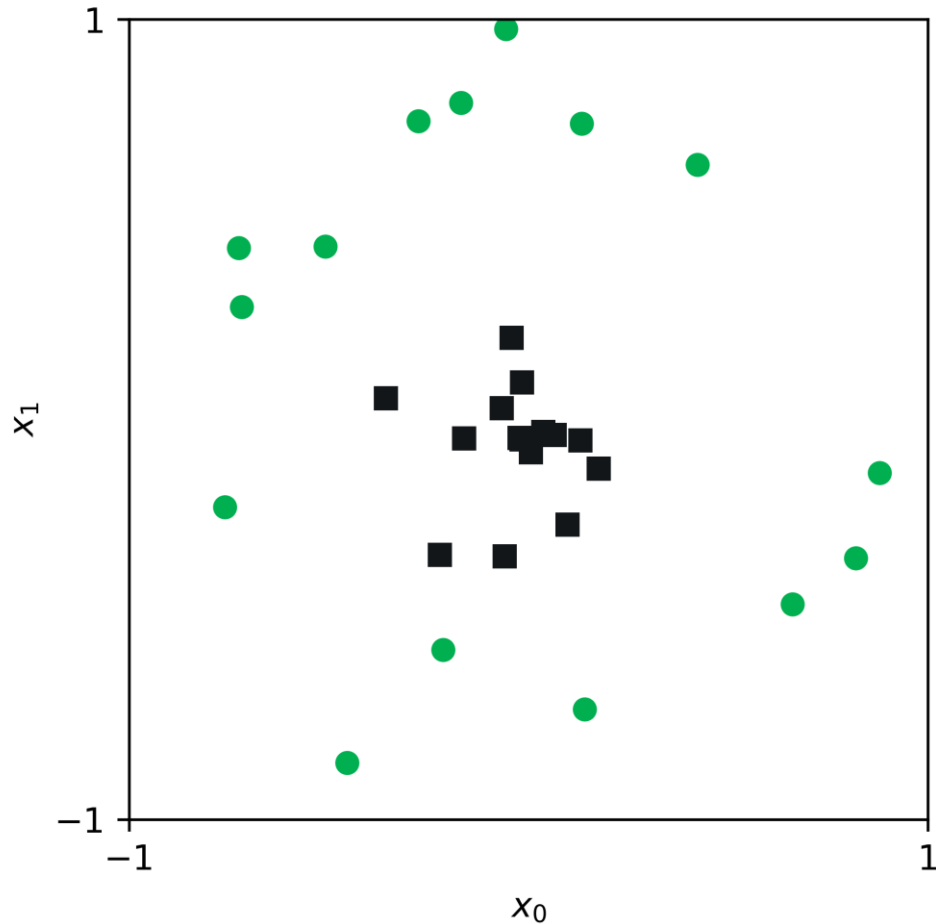
They can model **nonlinear relationships** between features and targets through **feature transformations**

There would need to be a linear relationship between the **transformed features** and the target variables for this to be effective

Limitations of linear decision boundaries

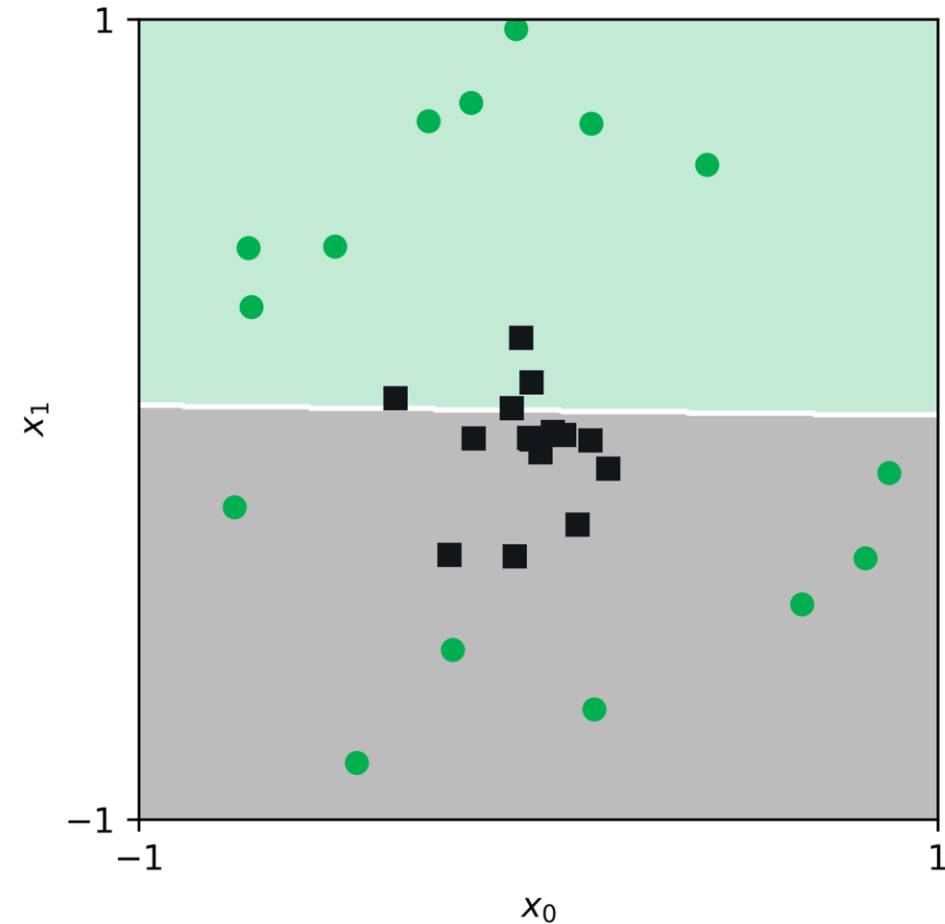
Original data

\mathbf{x}



Classify the features in this X -space

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



Transformations of features

Consider a digits example...

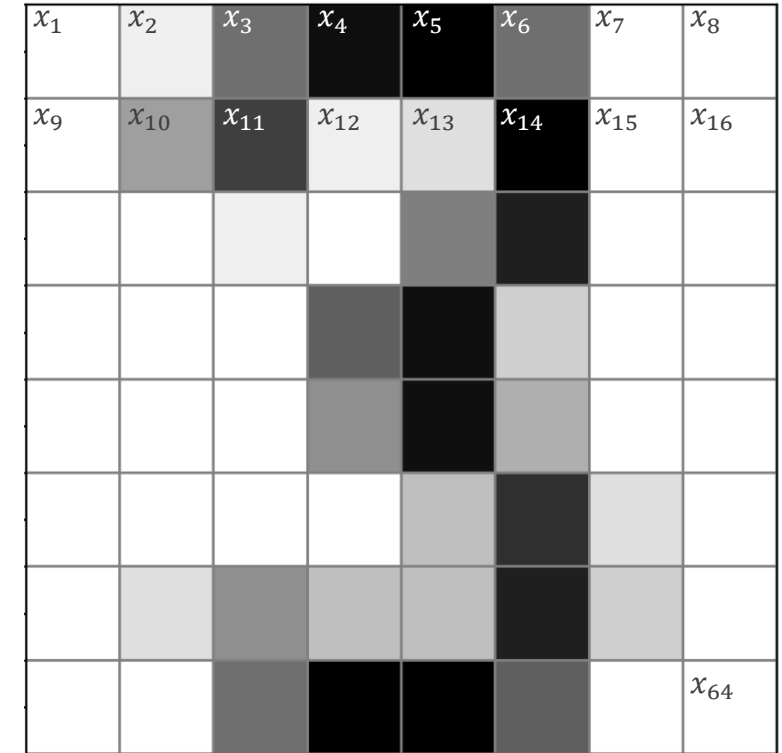
$$\mathbf{x} = [x_1, x_2, x_3, \dots, x_{64}]$$

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

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

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

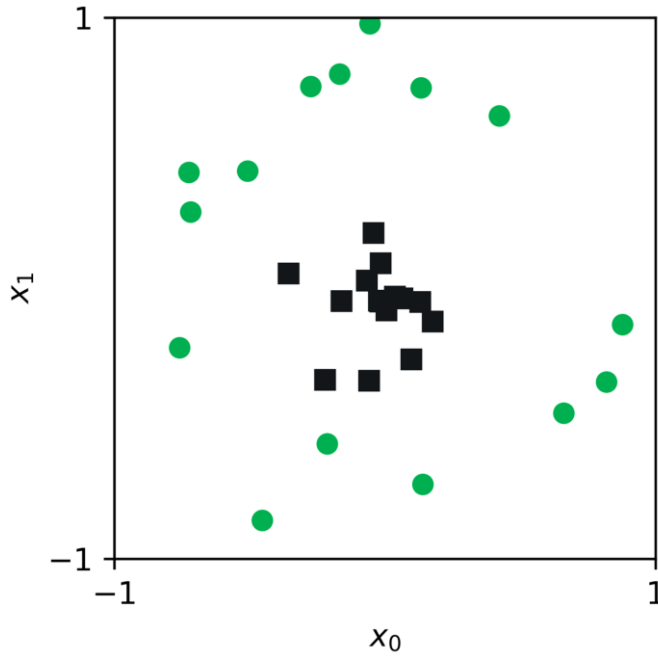
$$\mathbf{z} = [z_1, z_2, z_3]$$



Source: Abu-Mostafa, Learning from Data, Caltech

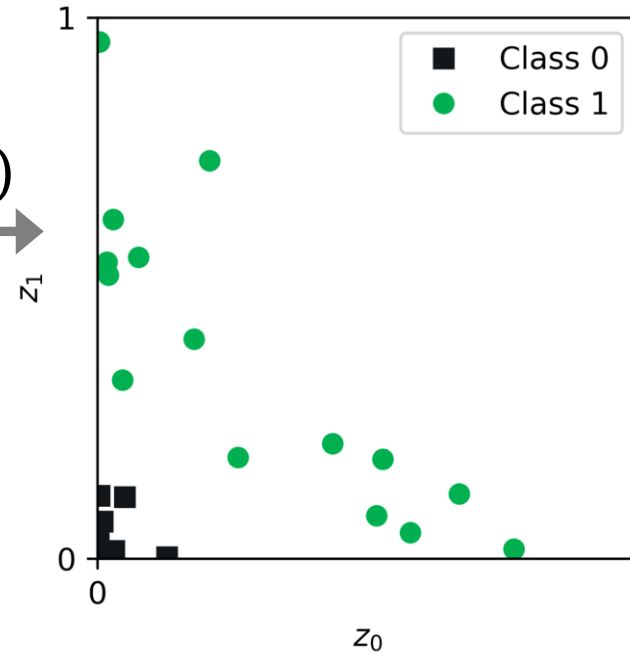
1

Original data
 \mathbf{x}



transform
the data

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



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

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

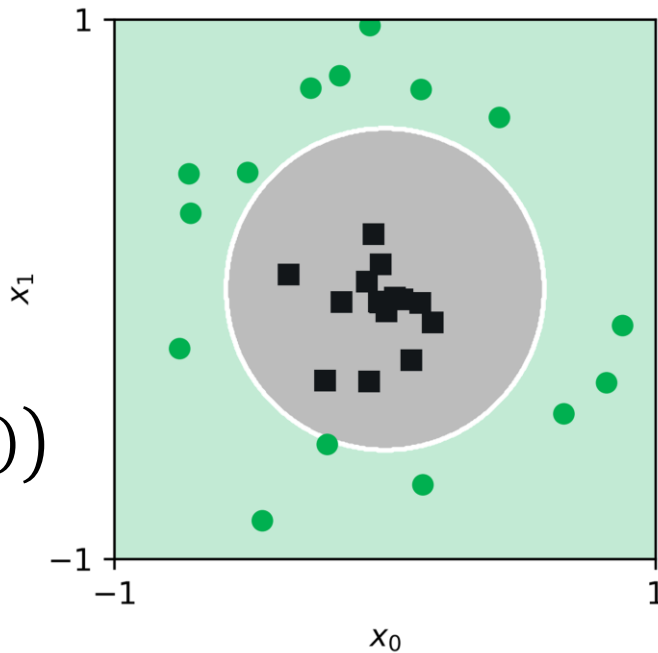
A new
representation
of our data

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

transform
the data back

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

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



Predictions in the
original X-space

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

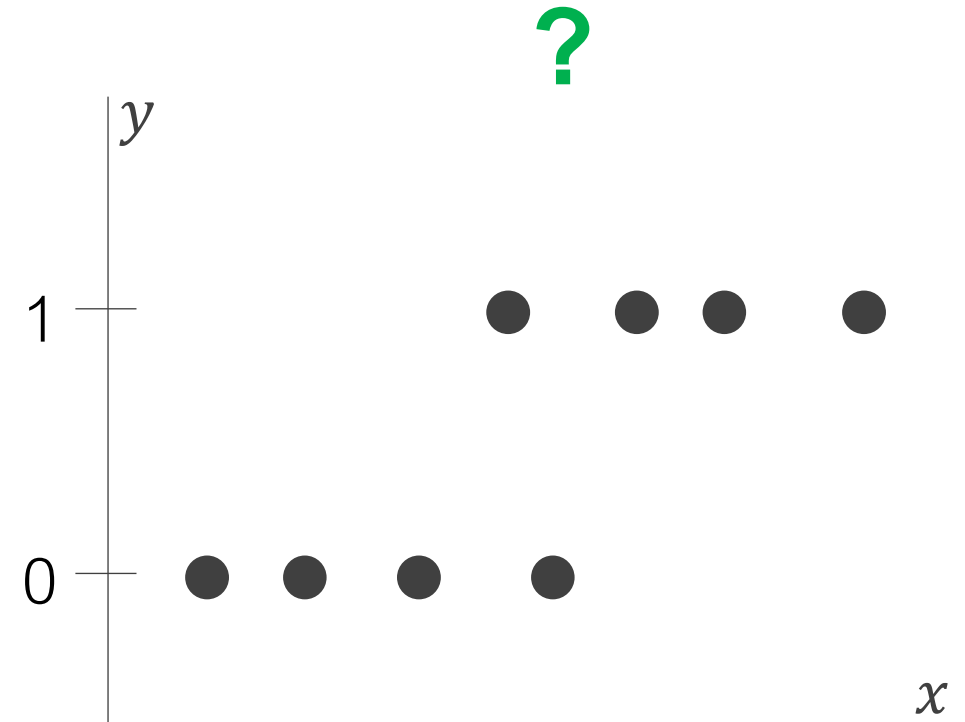
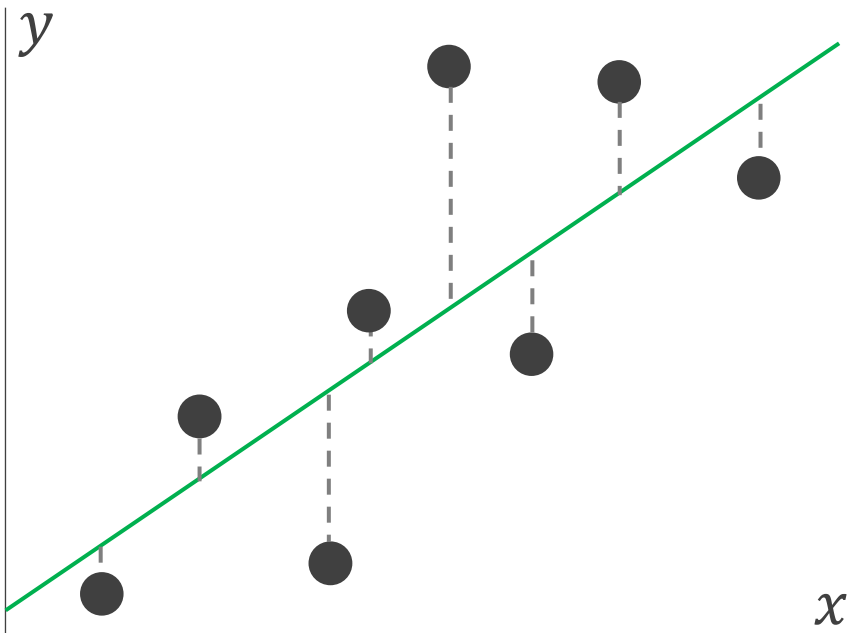
4

3

So how do we use linear models for classification?

How do we fit linear models for classification?

Linear regression



Moving from regression to classification

Regression

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

Classification
(perceptron)

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

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

where

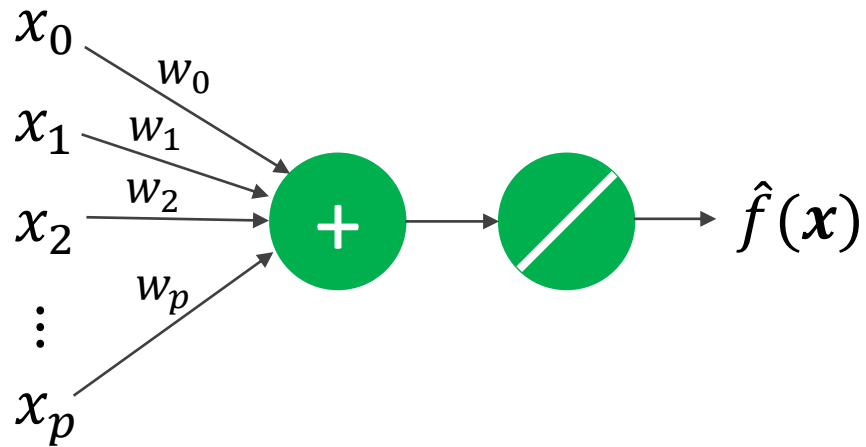
$$\text{sign}(x) = \begin{cases} 1 & x > 0 \\ -1 & \text{else} \end{cases}$$

Source: Abu-Mostafa, Learning from Data, Caltech

Moving from regression to classification

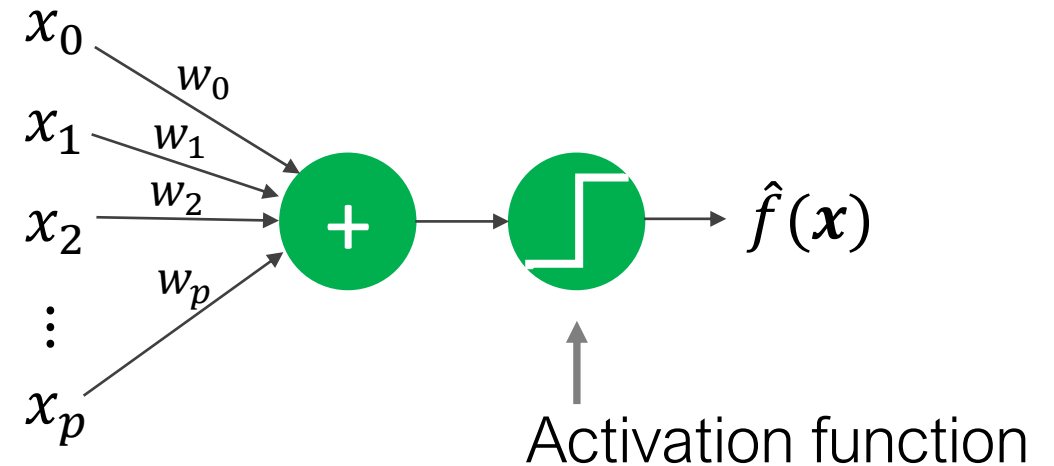
Linear Regression

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



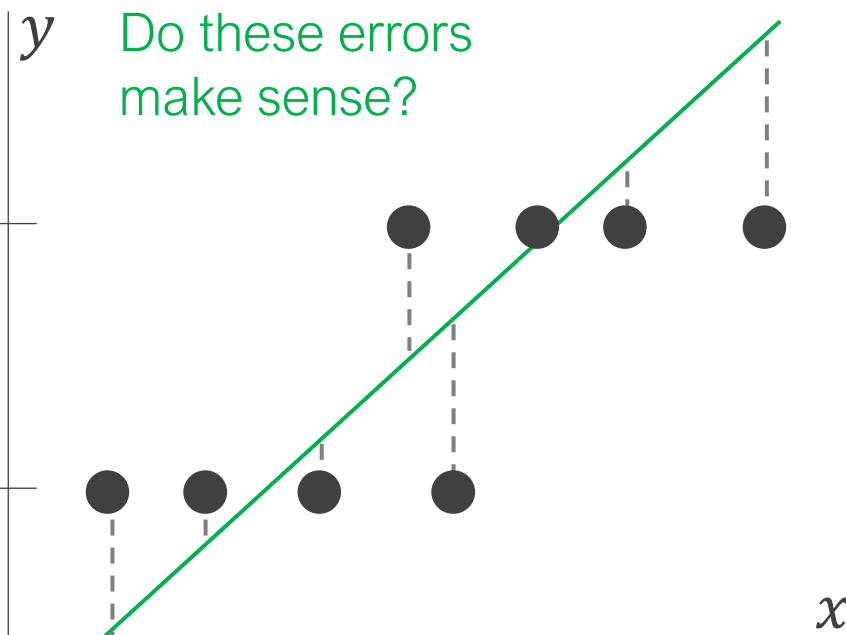
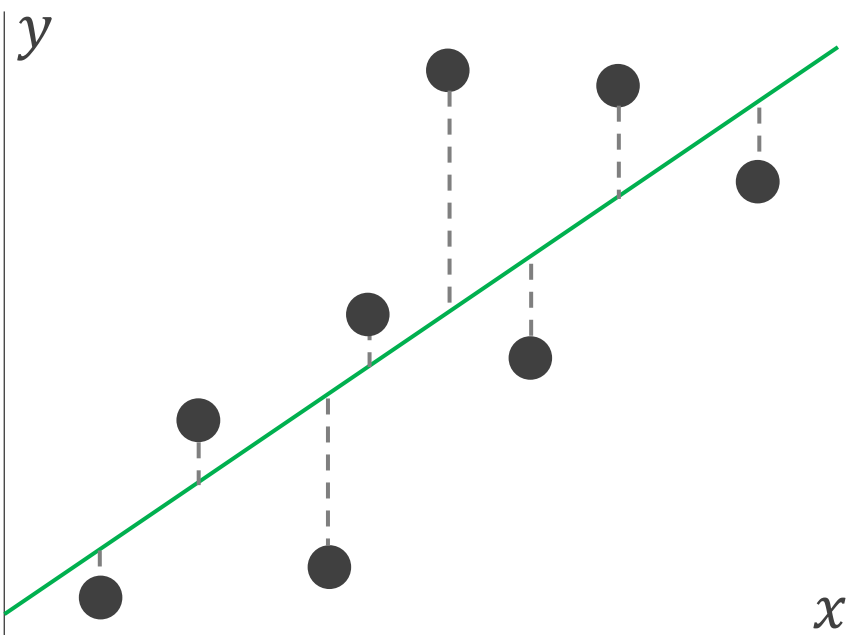
Linear Classification (perceptron)

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



Source: Abu-Mostafa, Learning from Data, Caltech

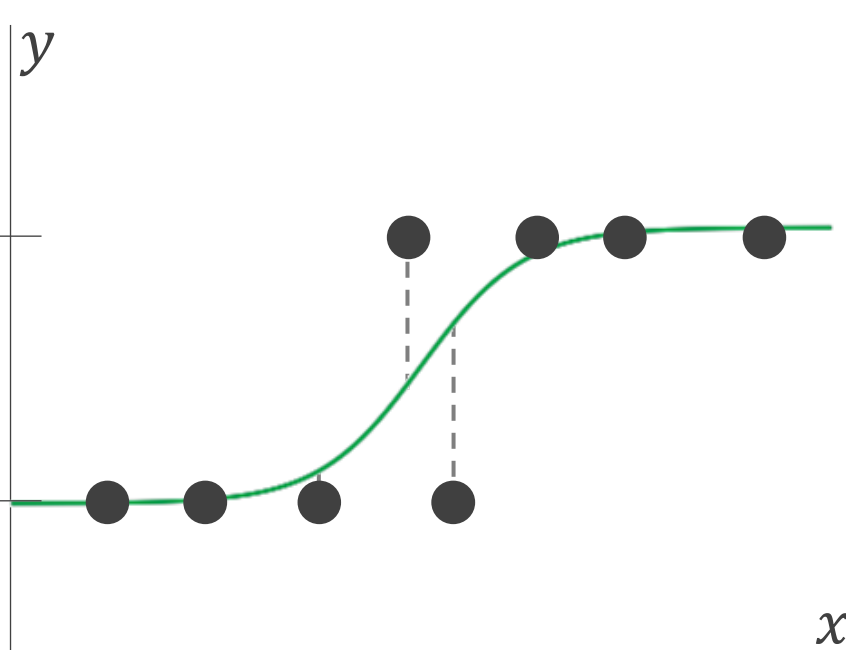
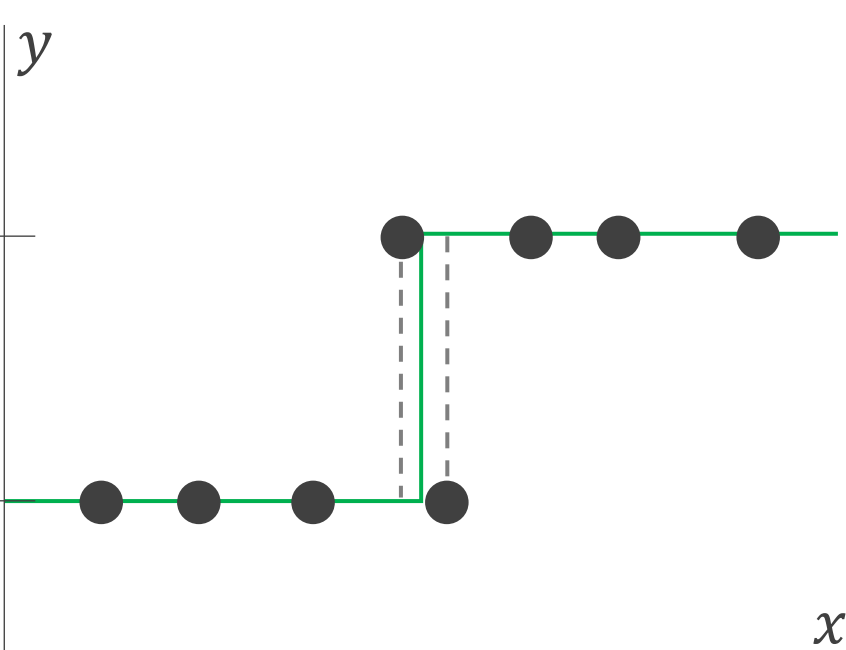
Linear regression
(linear activation)



Do these errors
make sense?

Linear regression
applied to a
classification
problem
(linear activation)

Perceptron
(sign activation)



Logistic regression
(sigmoid activation)

Sigmoid function

Definition

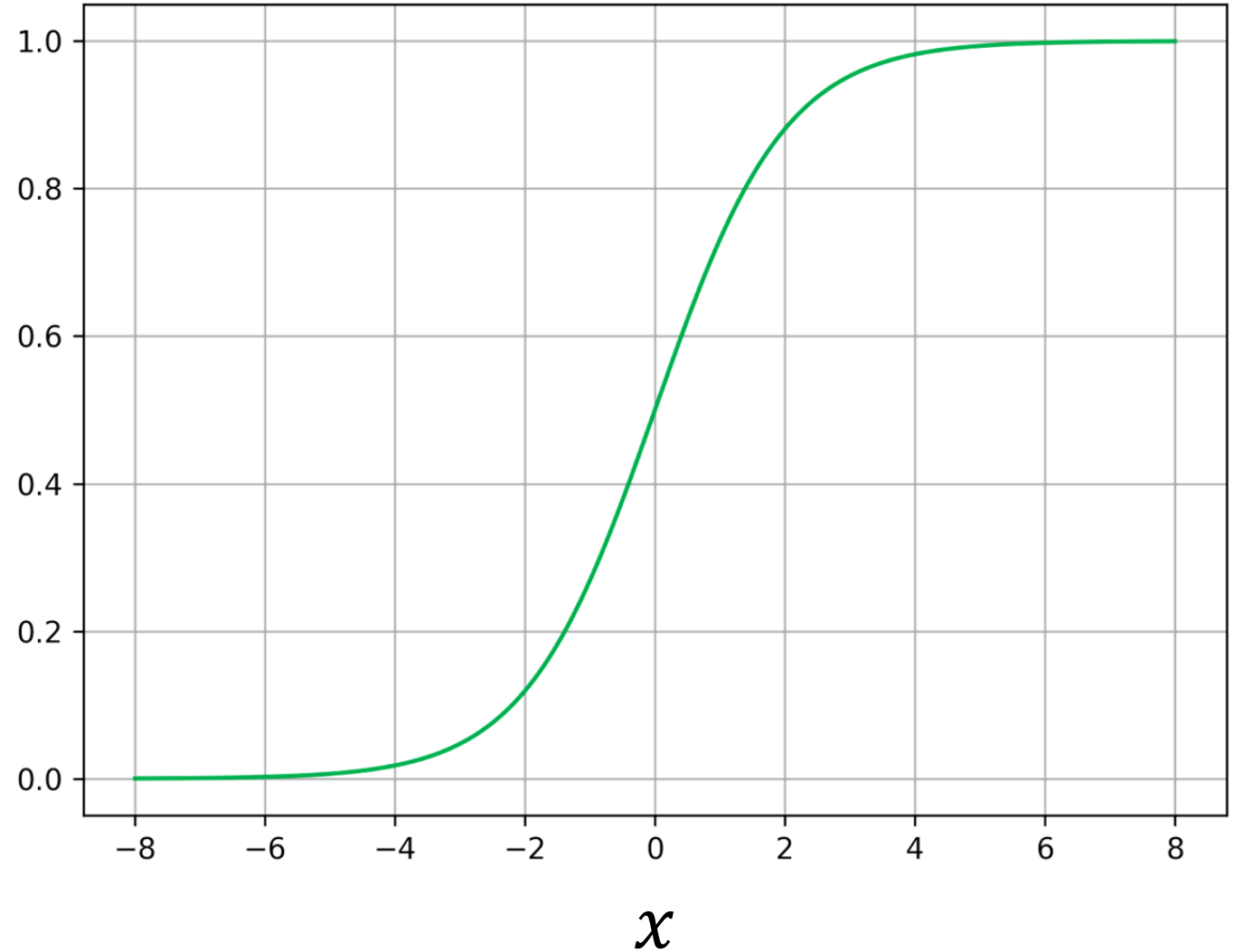
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

σ

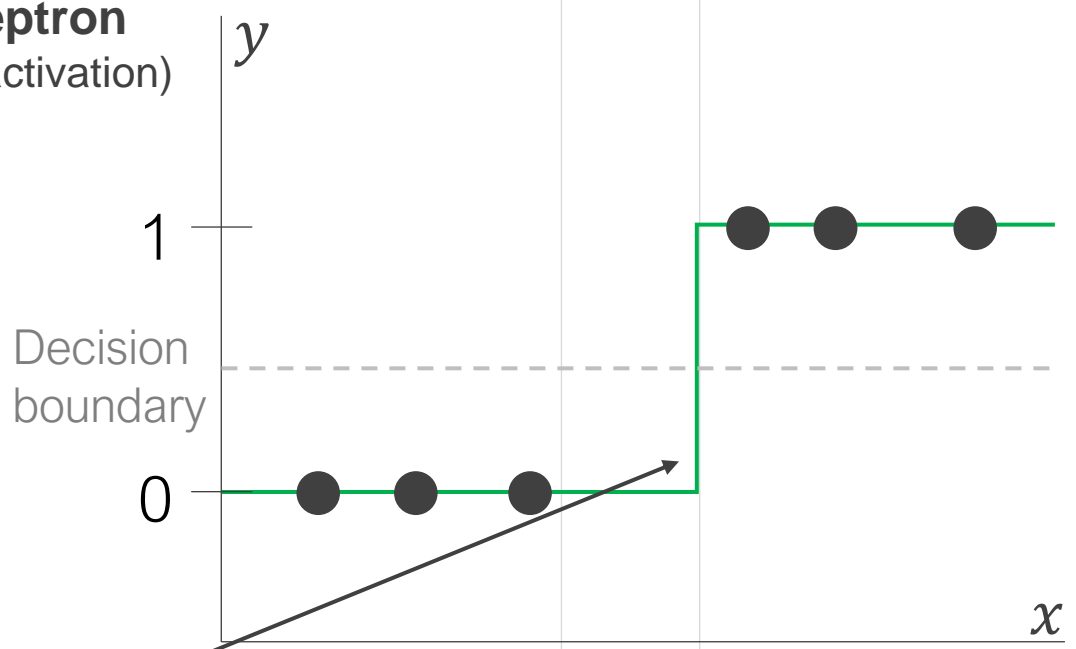
Useful properties

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

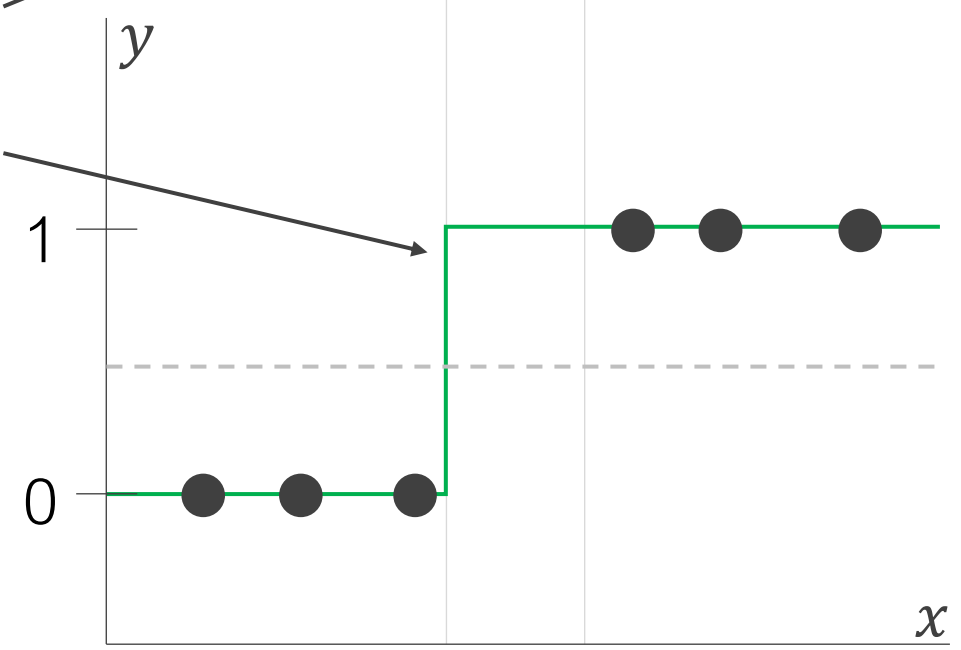
$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))$$



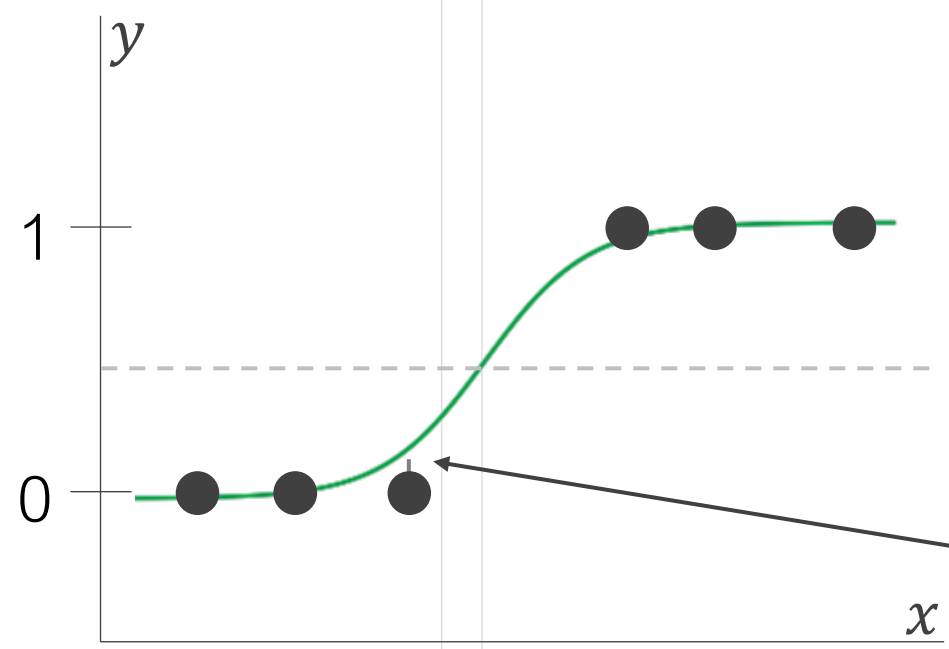
Perceptron (sign activation)



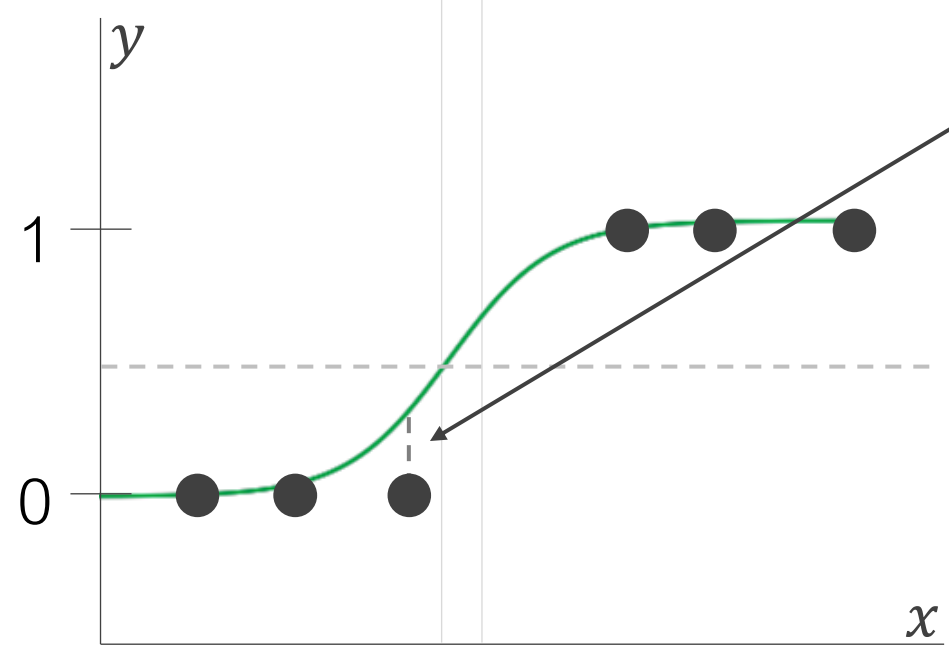
Both
decision
boundaries
incur the
same loss



Logistic regression (sigmoid activation)



The sigmoid
assigns error
to samples
close to the
margin

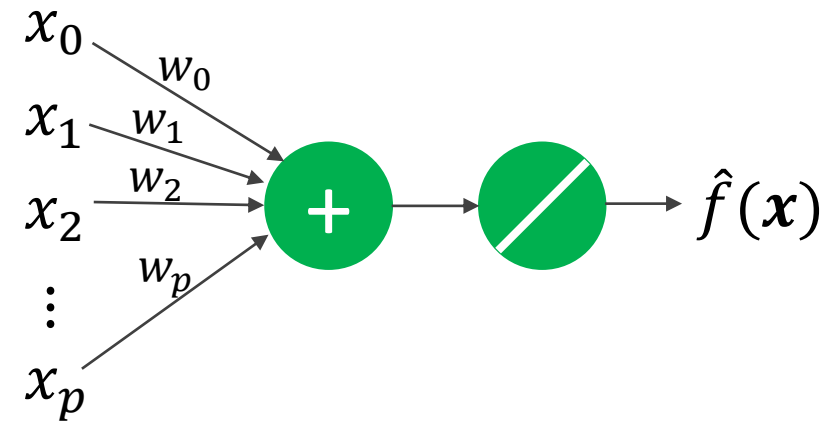


Favors a
larger margin

Moving from regression to classification

Linear Regression

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

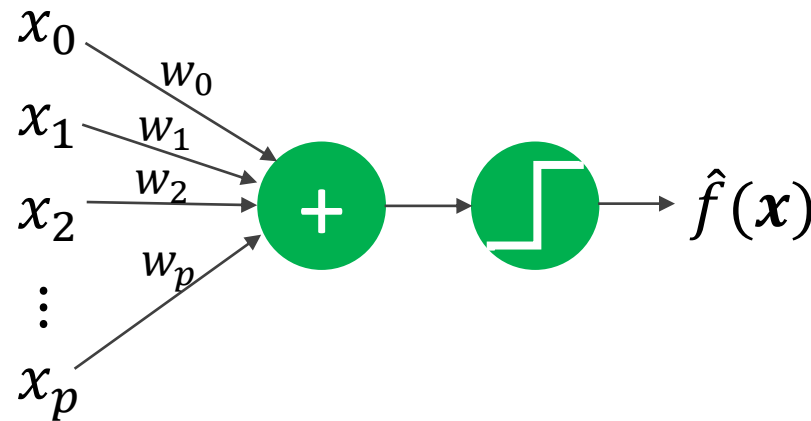


Linear Classification

Perceptron

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

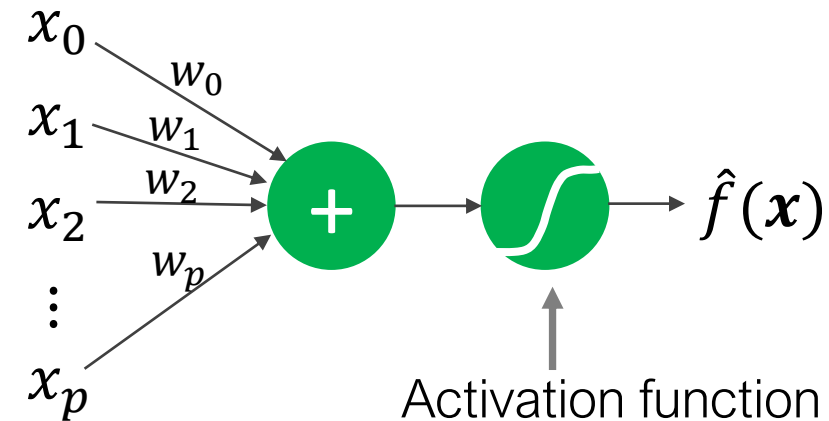
$$\text{sign}(x) = \begin{cases} 1 & x > 0 \\ -1 & \text{else} \end{cases}$$



Logistic Regression

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

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



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**

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

Plug in our model

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

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

Calculate the gradient

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

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

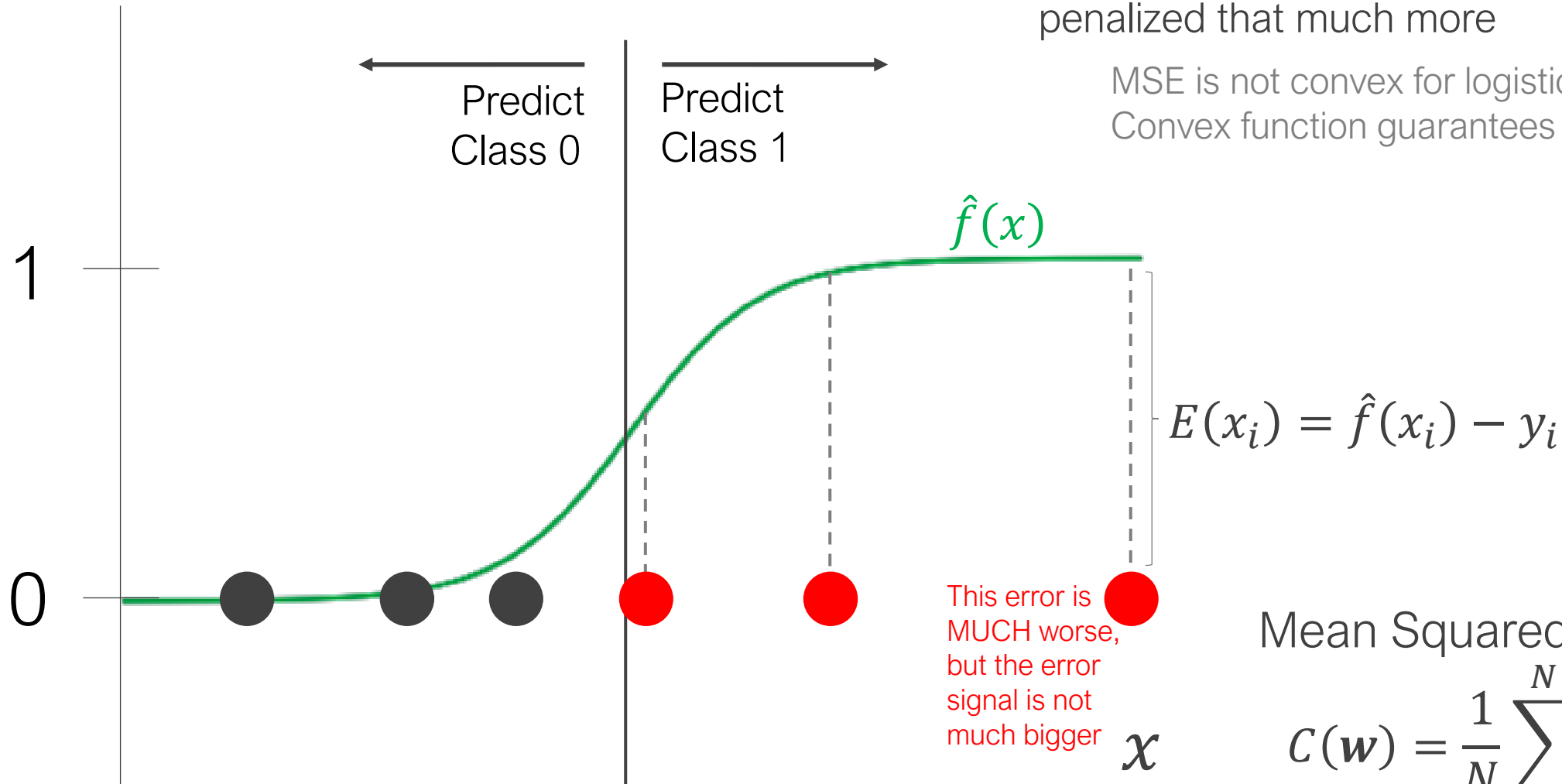
$$\nabla_{\mathbf{w}} C(\mathbf{w}) = \mathbf{0}$$

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:

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

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

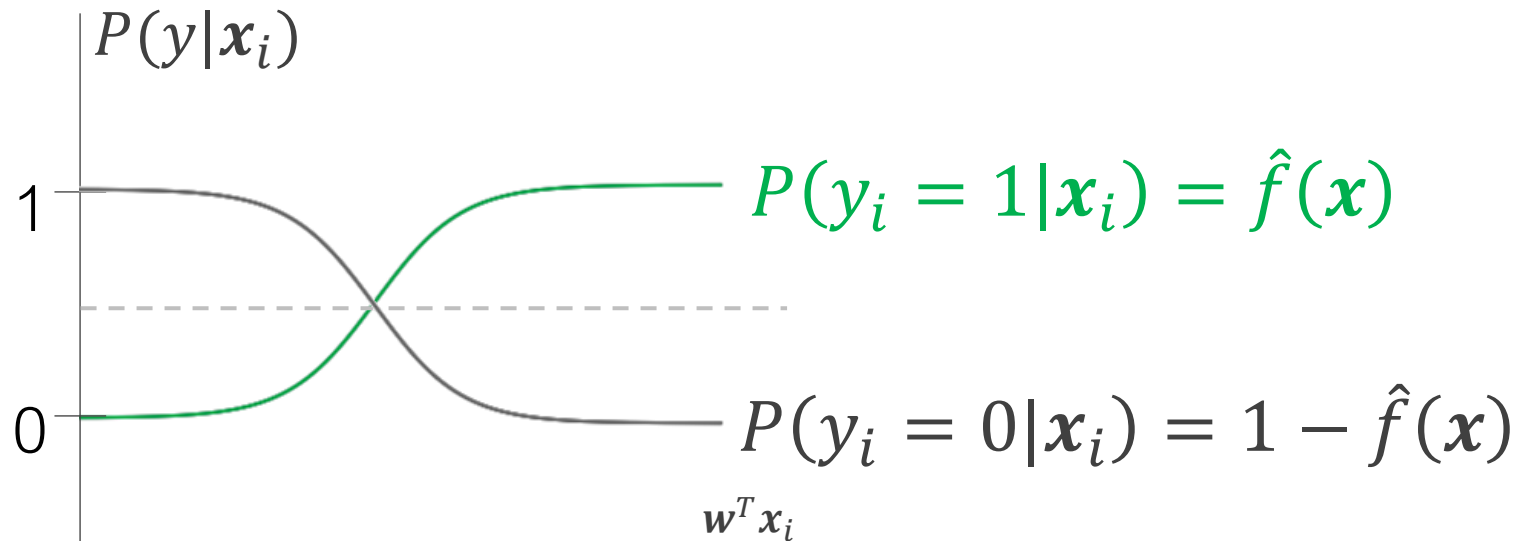
Is there a better cost function we could use for classification problems...?

Another interpretation of logistic regression

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

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

Logistic regression models the
conditional probability that a sample belongs to a class



What's linear about logistic regression?

$$\hat{f}(\mathbf{x}) = \hat{y} = \sigma(\mathbf{w}^T \mathbf{x})$$

$$\hat{y} = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

$$\frac{1}{\hat{y}} = 1 + e^{-\mathbf{w}^T \mathbf{x}}$$

$$\frac{1}{\hat{y}} - 1 = e^{-\mathbf{w}^T \mathbf{x}}$$

$$\frac{1}{\hat{y}} - \frac{\hat{y}}{\hat{y}} = e^{-\mathbf{w}^T \mathbf{x}}$$

$$\frac{1 - \hat{y}}{\hat{y}} = e^{-\mathbf{w}^T \mathbf{x}}$$

$$\frac{1 - \hat{y}}{\hat{y}} = e^{-\mathbf{w}^T \mathbf{x}}$$

$$\log\left(\frac{1 - \hat{y}}{\hat{y}}\right) = \log(e^{-\mathbf{w}^T \mathbf{x}})$$

$$\log\left(\frac{1 - \hat{y}}{\hat{y}}\right) = -\mathbf{w}^T \mathbf{x}$$

$$-\log\left(\frac{1 - \hat{y}}{\hat{y}}\right) = \mathbf{w}^T \mathbf{x}$$

$$\log\left(\frac{\hat{y}}{1 - \hat{y}}\right) = \mathbf{w}^T \mathbf{x}$$

If we interpret our target variable, \hat{y} , as the probability of class 1, then $\mathbf{w}^T \mathbf{x}$ models the log odds ratio

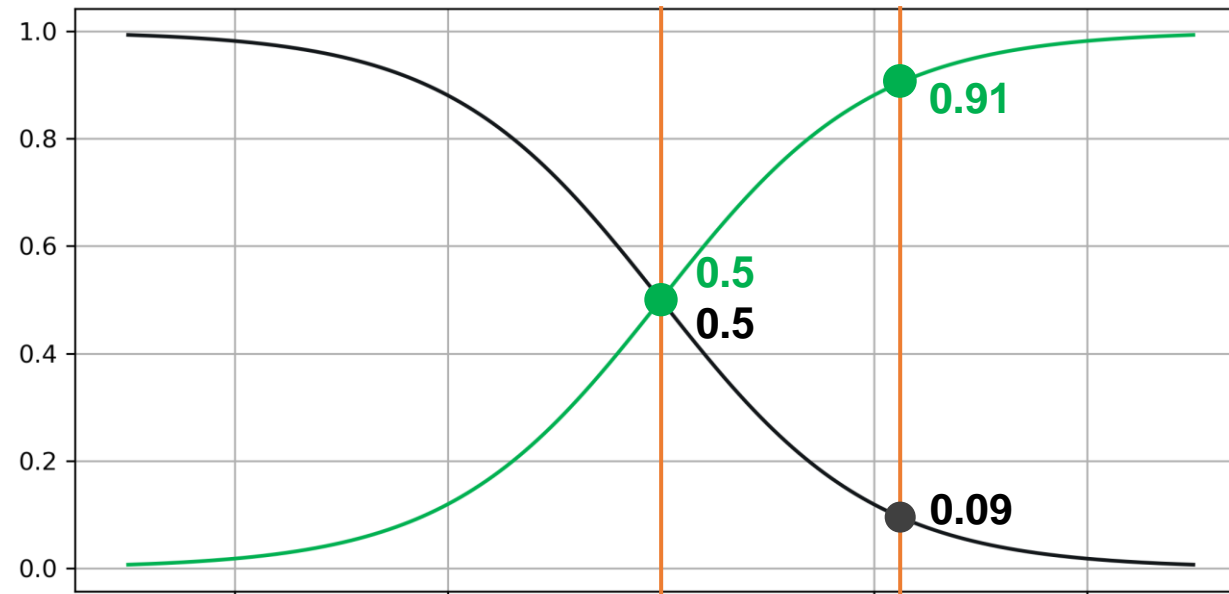
$$\hat{y} = P(Y = 1 | \mathbf{x})$$

$$\log\left[\frac{P(Y = 1 | \mathbf{x})}{1 - P(Y = 1 | \mathbf{x})}\right] = \mathbf{w}^T \mathbf{x}$$

$$\log\left[\frac{P(Y = 1 | \mathbf{x})}{P(Y = 0 | \mathbf{x})}\right] = \mathbf{w}^T \mathbf{x}$$

Interpretation of the odds ratio

$$P(y_i|x_i)$$

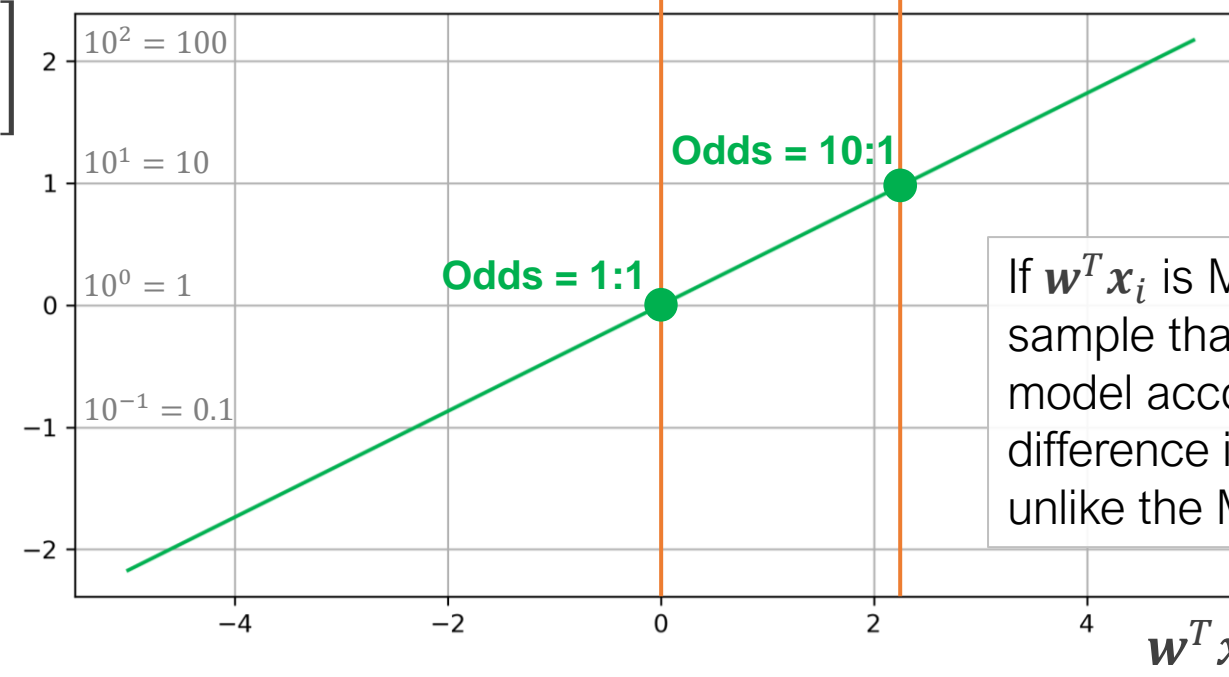


$$P(y_i = 1|x_i) = \sigma(w^T x_i)$$

$$P(y_i = 0|x_i) = 1 - \sigma(w^T x_i)$$

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

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



If $w^T x_i$ is MUCH larger for one sample than another, this model accounts for this difference in the odds ratio unlike the MSE approach

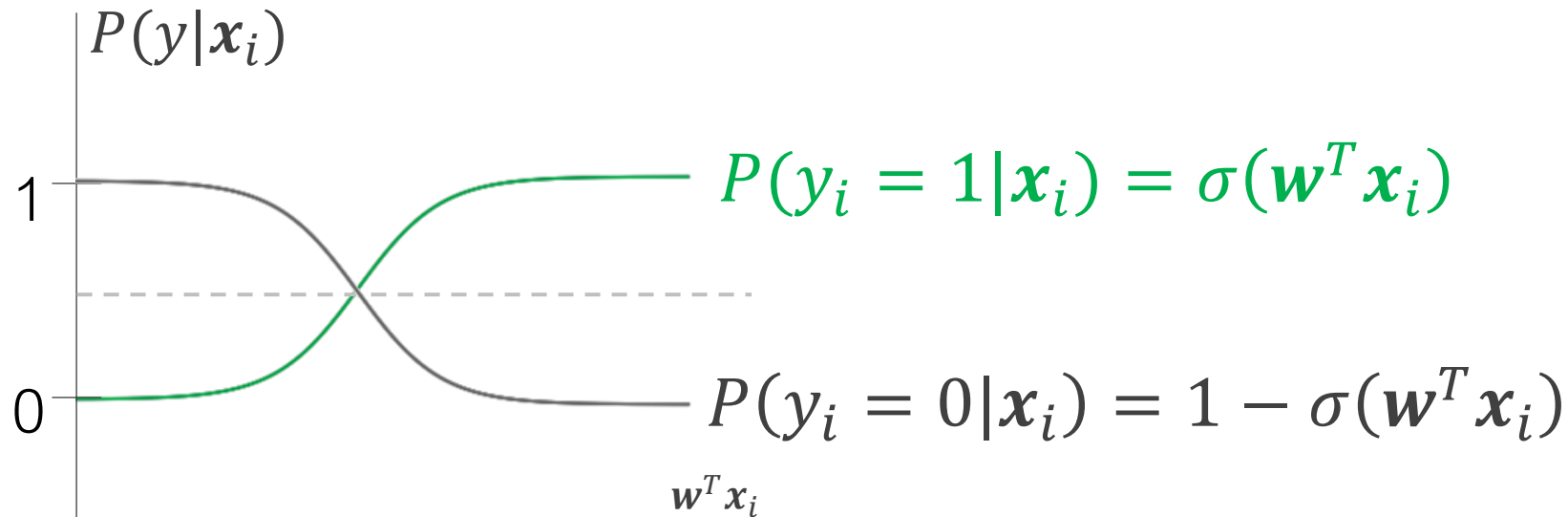
The log-odds ratio is a linear function of the features

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

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

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

Logistic regression models the
conditional probability that a sample belongs to a class



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

For independent random events, the probability of both events is the product of their individual probabilities:

$$P(A \text{ and } B) = P(A)P(B)$$

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

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

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 $L(p|\mathbf{x})$

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:

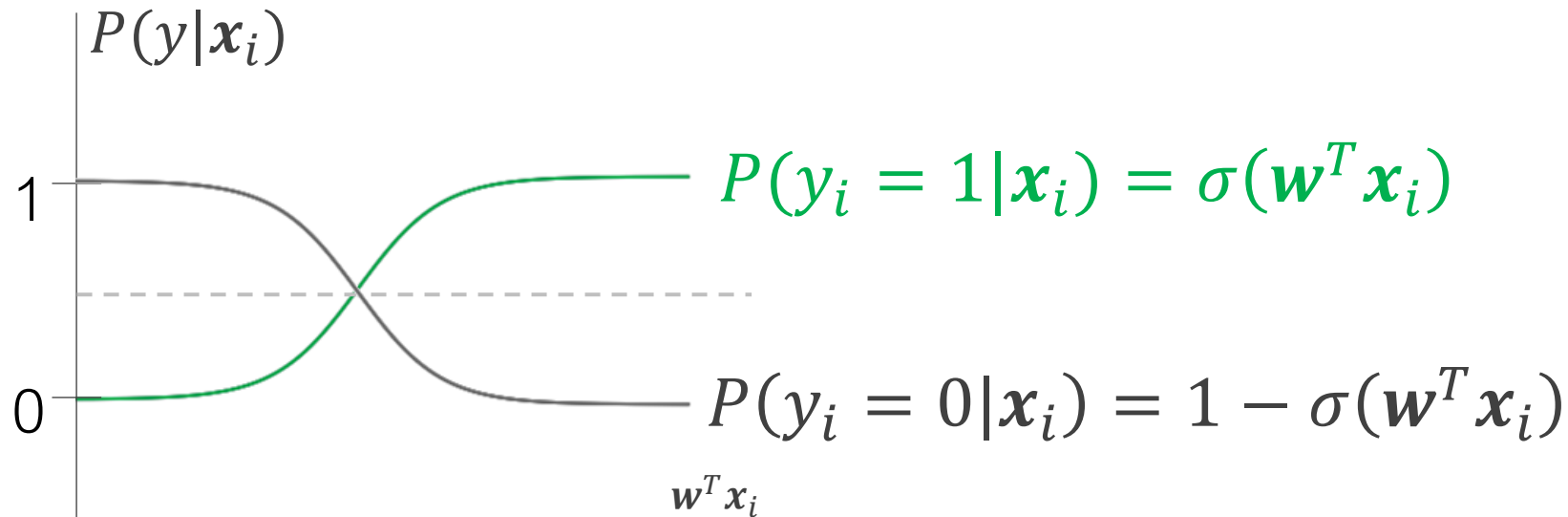
$$\hat{p} = \frac{1}{N} \sum_{i=1}^N x_i$$

Applying this to logistic regression...

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

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

Logistic regression models the
conditional probability that a sample belongs to a class



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

The interpretation of the **Likelihood**

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

The likelihood for **one observation**:

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

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

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

The likelihood for **all observations**:

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

Source: Malik Magdon-Ismail, Learning from Data

The likelihood for all observations:

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

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

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

We want to **MAXIMIZE the likelihood (minimize its negative)**

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

A little algebra

$$\begin{aligned} L(\mathbf{w}|\mathbf{y}, \mathbf{X}) &= \prod_{i=1}^N \sigma(\mathbf{w}^T \mathbf{x}_i)^{y_i} [1 - \sigma(\mathbf{w}^T \mathbf{x}_i)]^{1-y_i} \\ &= \prod_{i=1}^N \hat{y}_i^{y_i} [1 - \hat{y}_i]^{1-y_i} \quad \text{assuming} \quad \hat{y}_i \triangleq \sigma(\mathbf{w}^T \mathbf{x}_i) \end{aligned}$$

If we take the log of both sides:

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

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

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

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

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

This quantity is often normalized by
dividing by N for interpreting the
results as **mean cost per sample**

For logistic regression,
 $\hat{y}_i \triangleq \sigma(\mathbf{w}^T \mathbf{x}_i)$

This is the **cross entropy** cost function

Cross Entropy

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

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

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

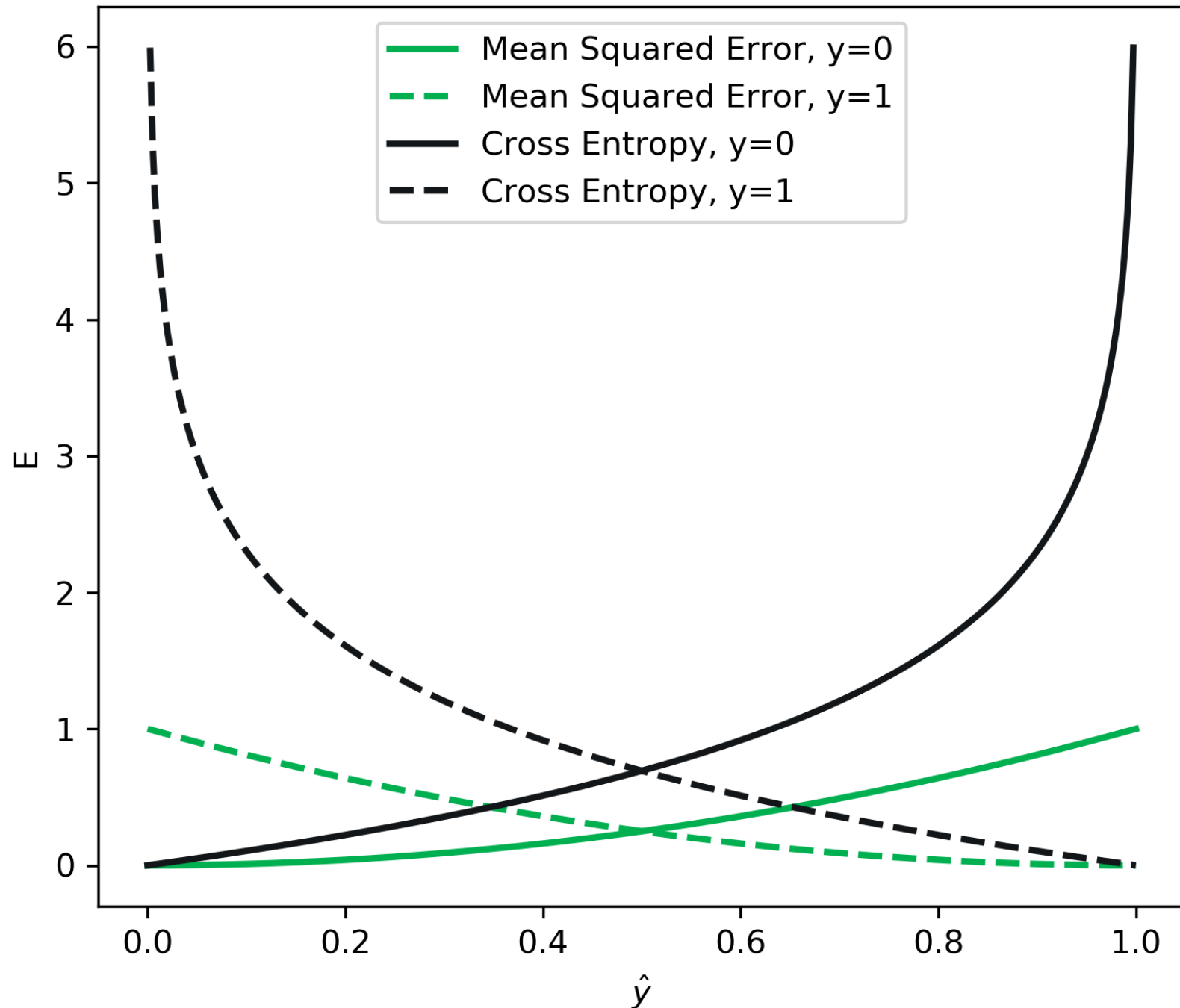
Single sample cost

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

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

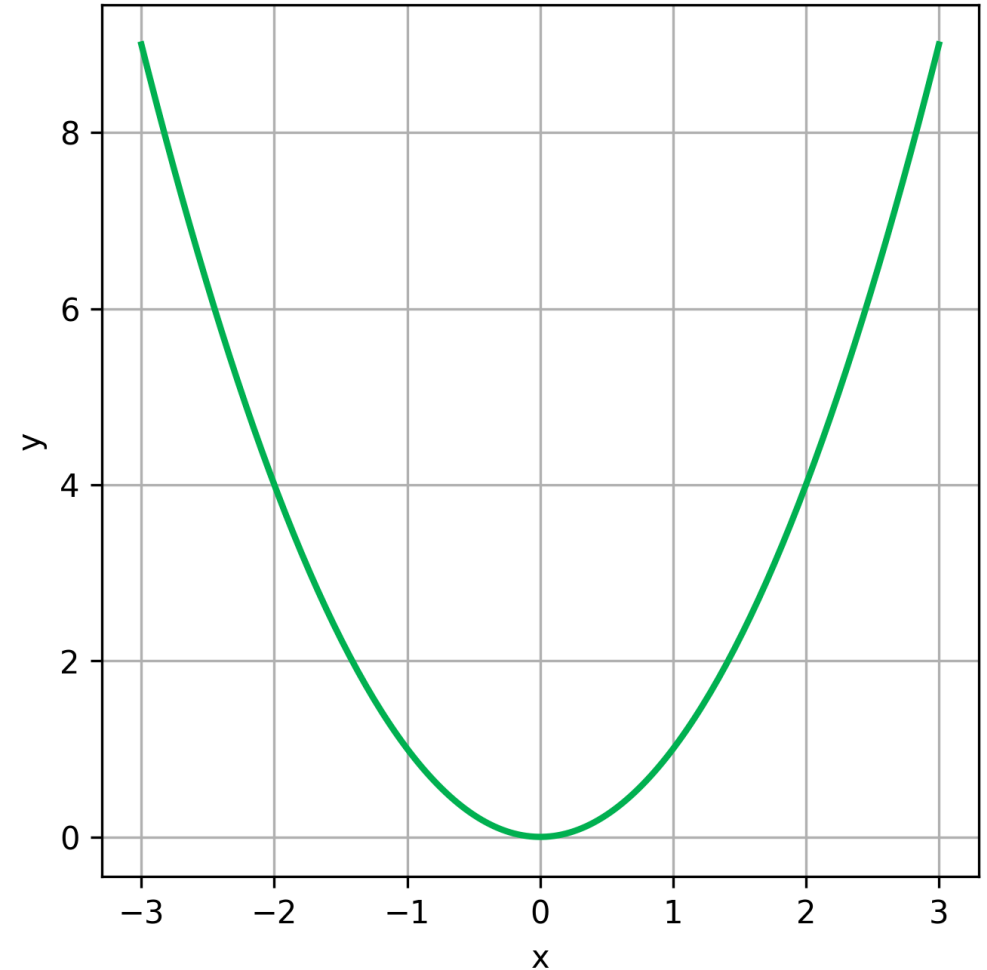
Minimize $y = x^2$

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

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

Learning
rate

Direction to
move in



Gradient descent

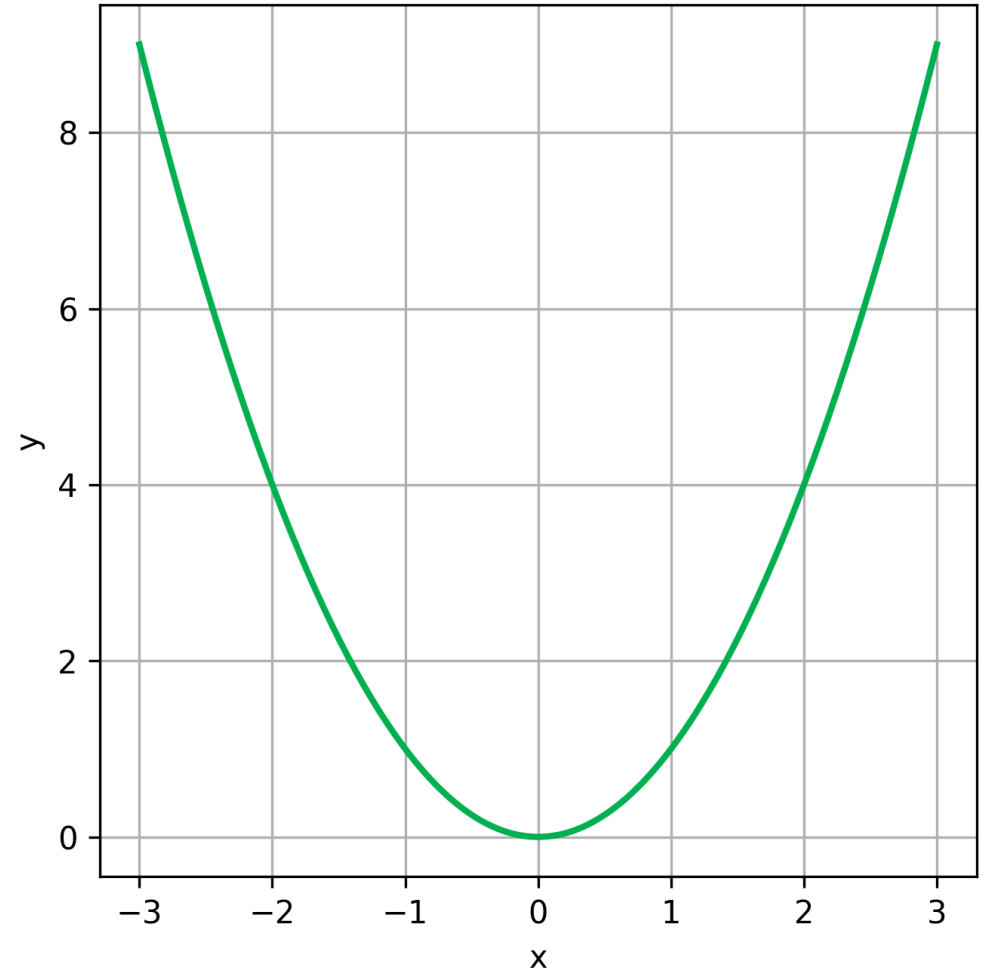
Minimize $f(x) = x^2$

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

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



Gradient descent

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

Gradient descent update equation:
$$x^{(i+1)} = x^{(i)} - \eta \nabla f(x^{(i)})$$

Minimize $f(x) = x^2$

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

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

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

i	$x^{(i)}$	$y^{(i)}$
-----	-----------	-----------

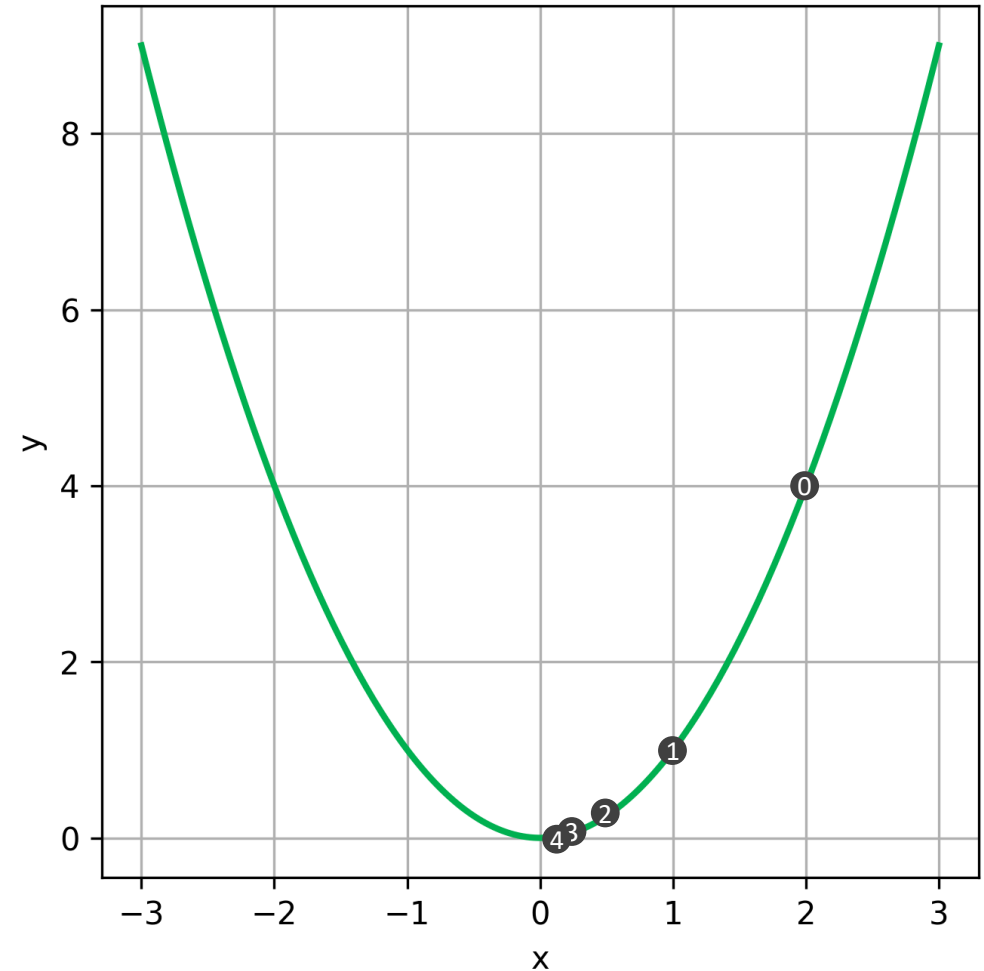
0	2	4
---	---	---

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 change nonlinear relationships into linear relationships

Logistic regression is suited for classification

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

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