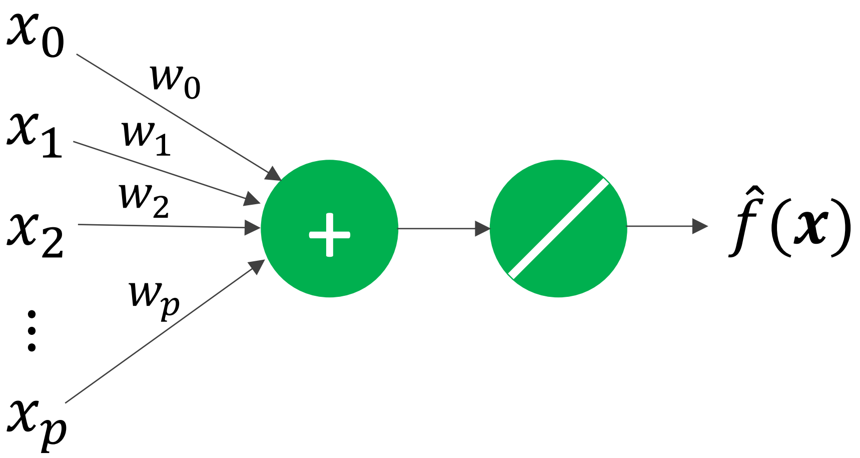
Machine Learning IV

Moving from regression to classification

Linear Regression

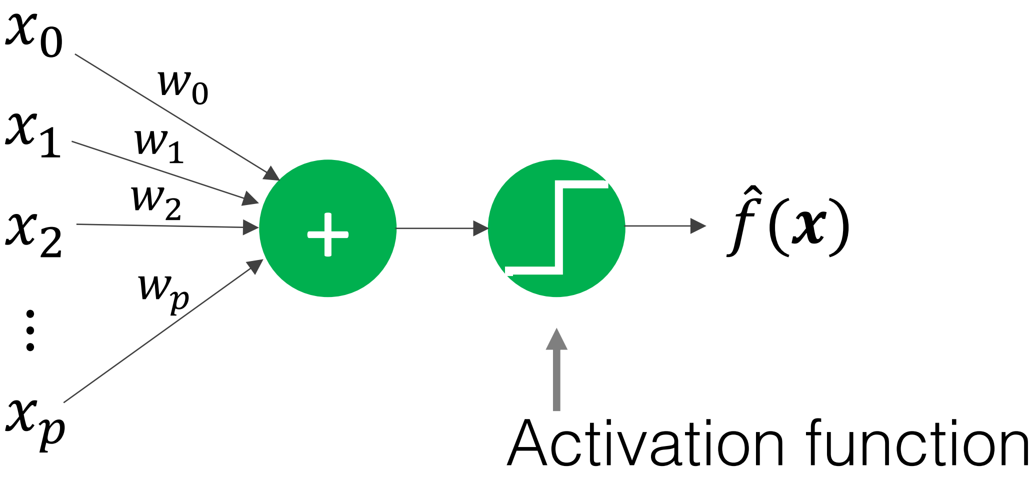
$$\hat{f}(\boldsymbol{x}) = \sum_{i=0}^{p} w_i x_i$$



Linear Classification

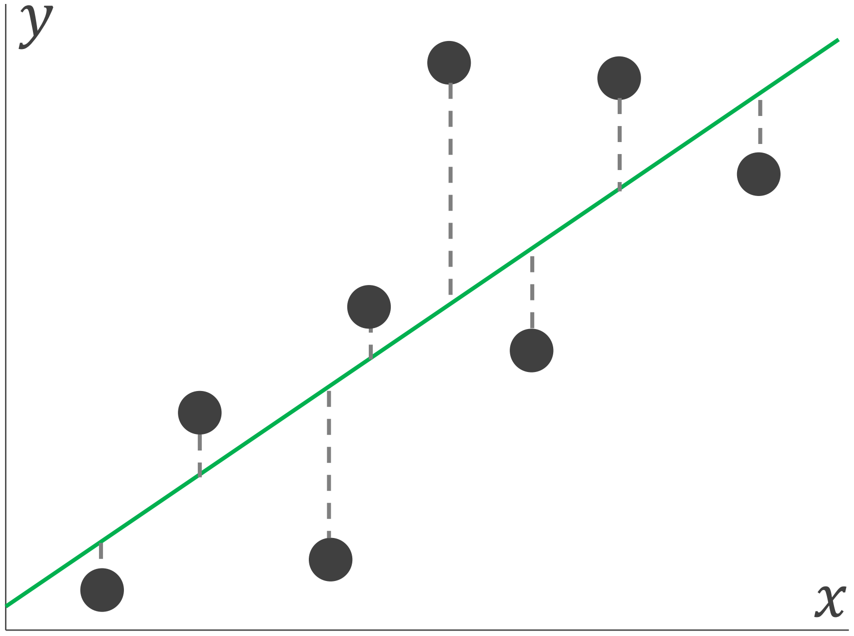
(perceptron)

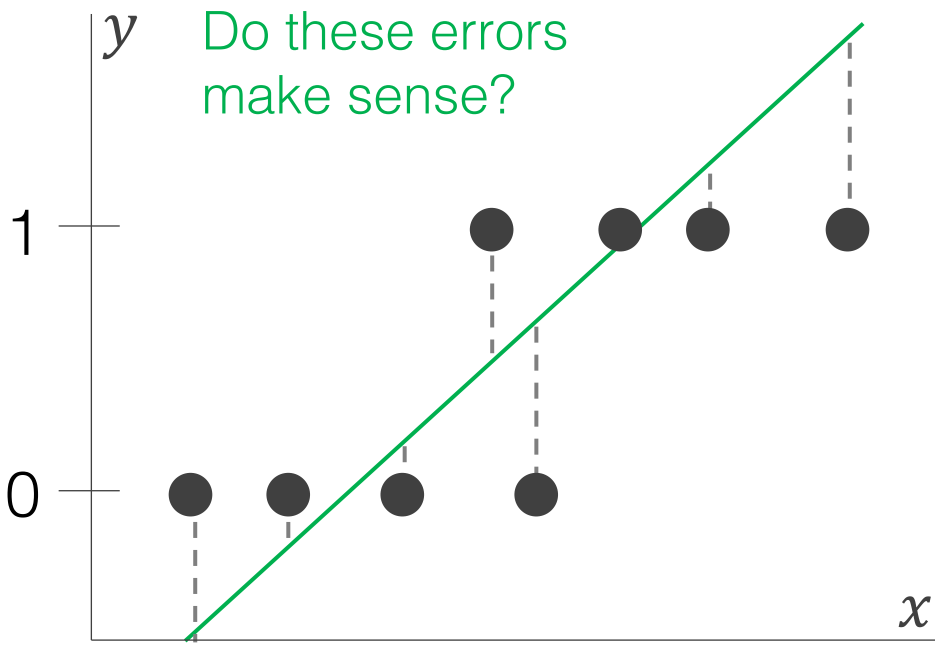
$$\hat{f}(\boldsymbol{x}) = sign\left(\sum_{i=0}^{p} w_i x_i\right)$$



Source: Abu-Mostafa, Learning from Data, Caltech

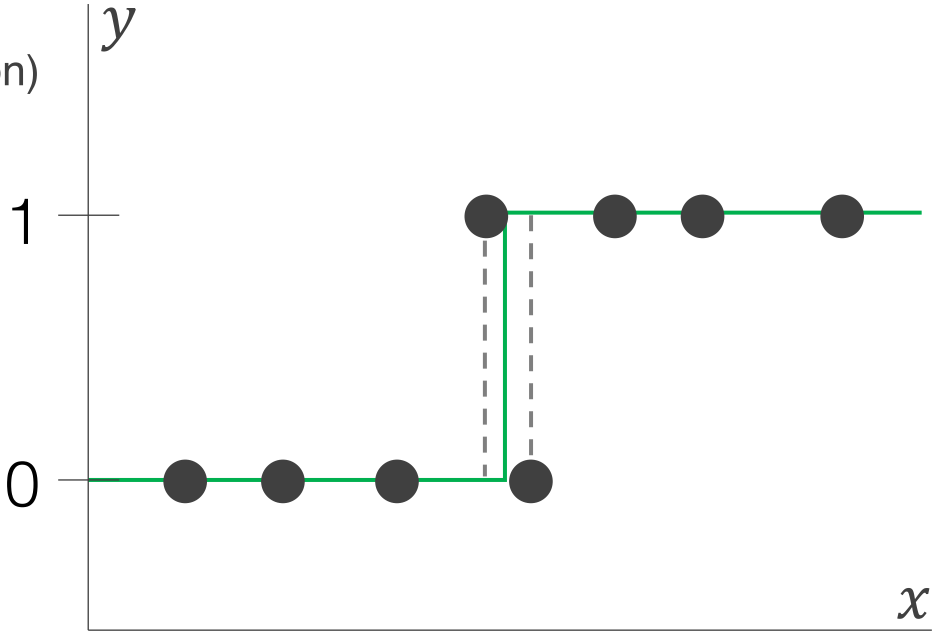
Linear regression
(linear activation)

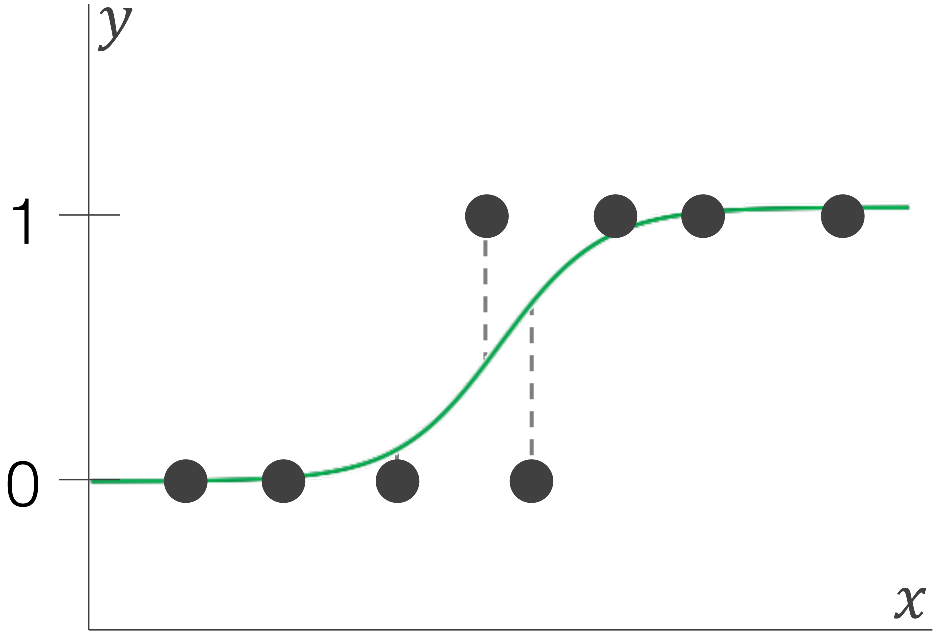




Linear regression
applied to a classification problem
(linear activation)

Perceptron
(sign activation)





Logistic regression
(sigmoid activation)

Sigmoid function

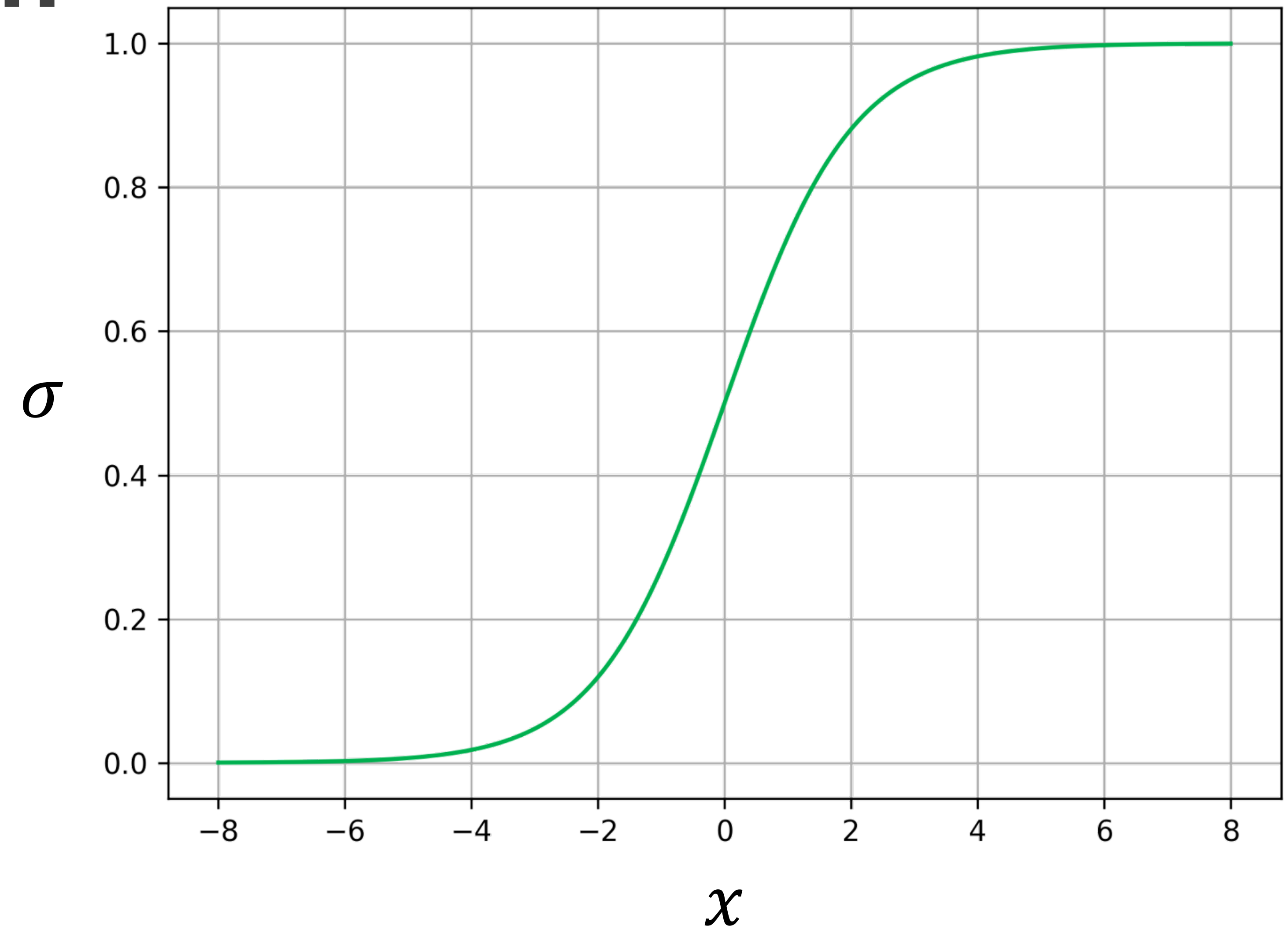
Definition

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

Useful properties

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

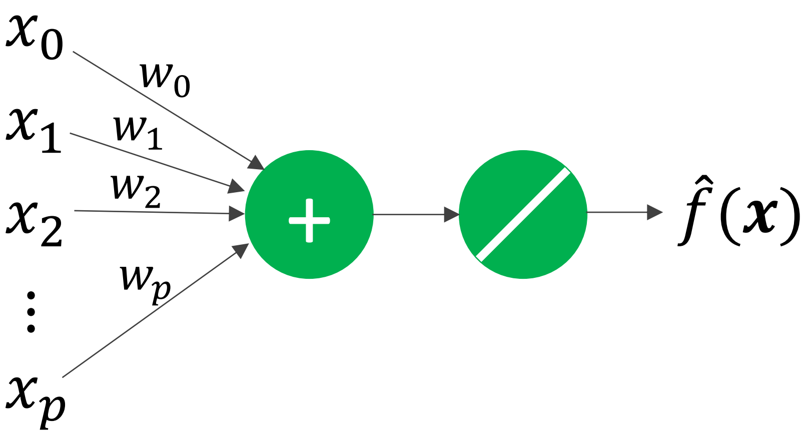
$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))$$



Moving from regression to classification

Linear Regression

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

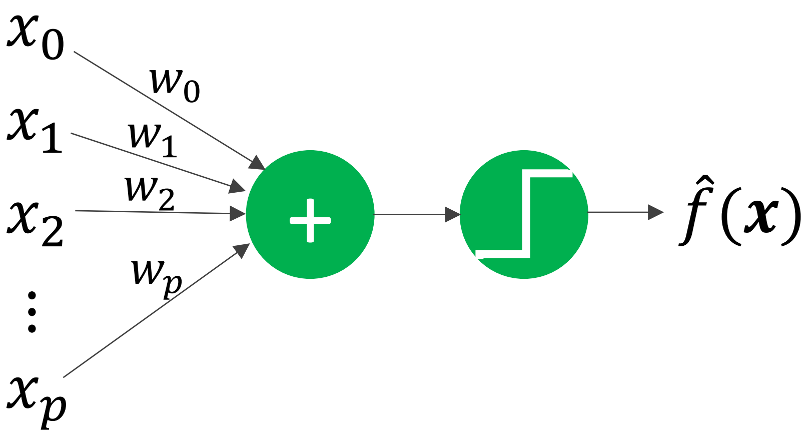


Linear Classification

Perceptron

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

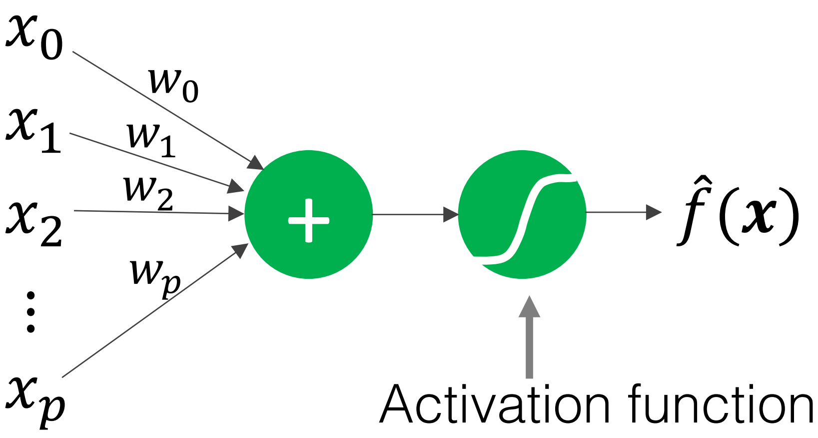
$$sign(x) = \begin{cases} 1 & x > 0 \\ -1 & \text{else} \end{cases}$$



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 doesn't work, we typically 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

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

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

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

But does MSE make sense in this situation?

But we don't for logistic regression...

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

Mean Square Error vs Cross Entropy

$$\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

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

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

We need a new approach…

Gradient descent

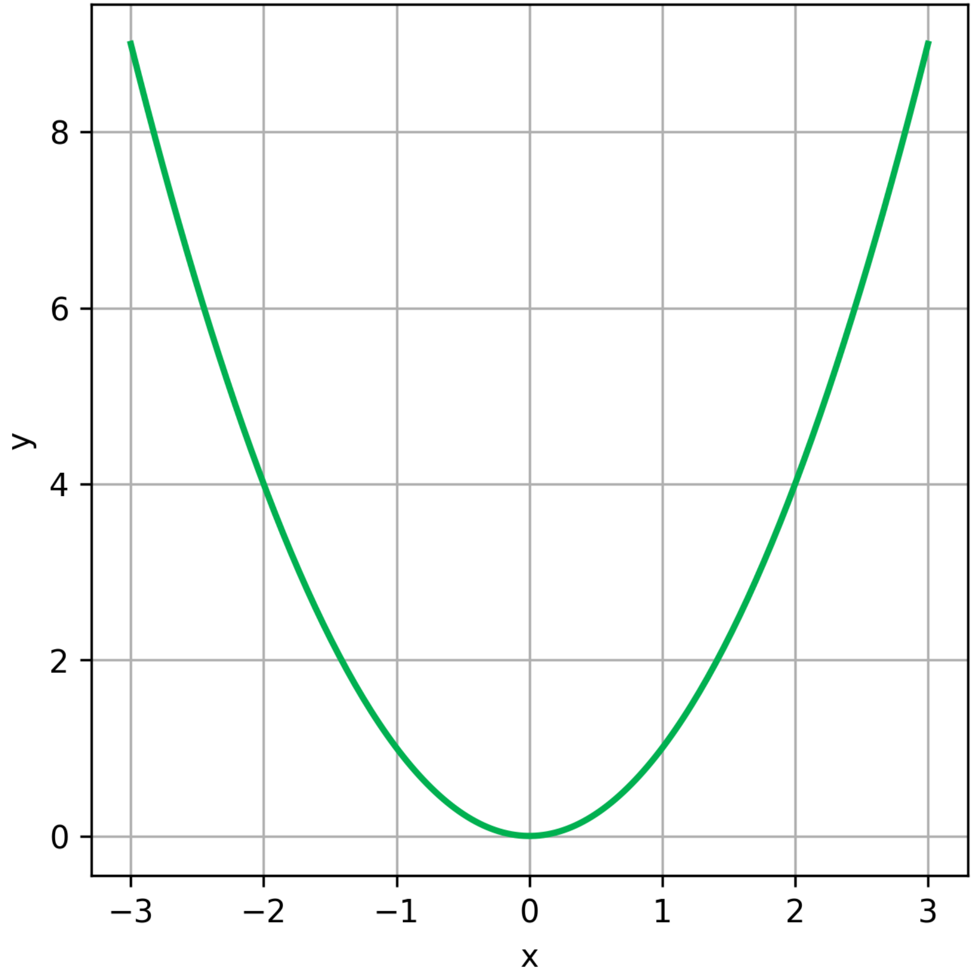
Minimize $y = x^2$

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

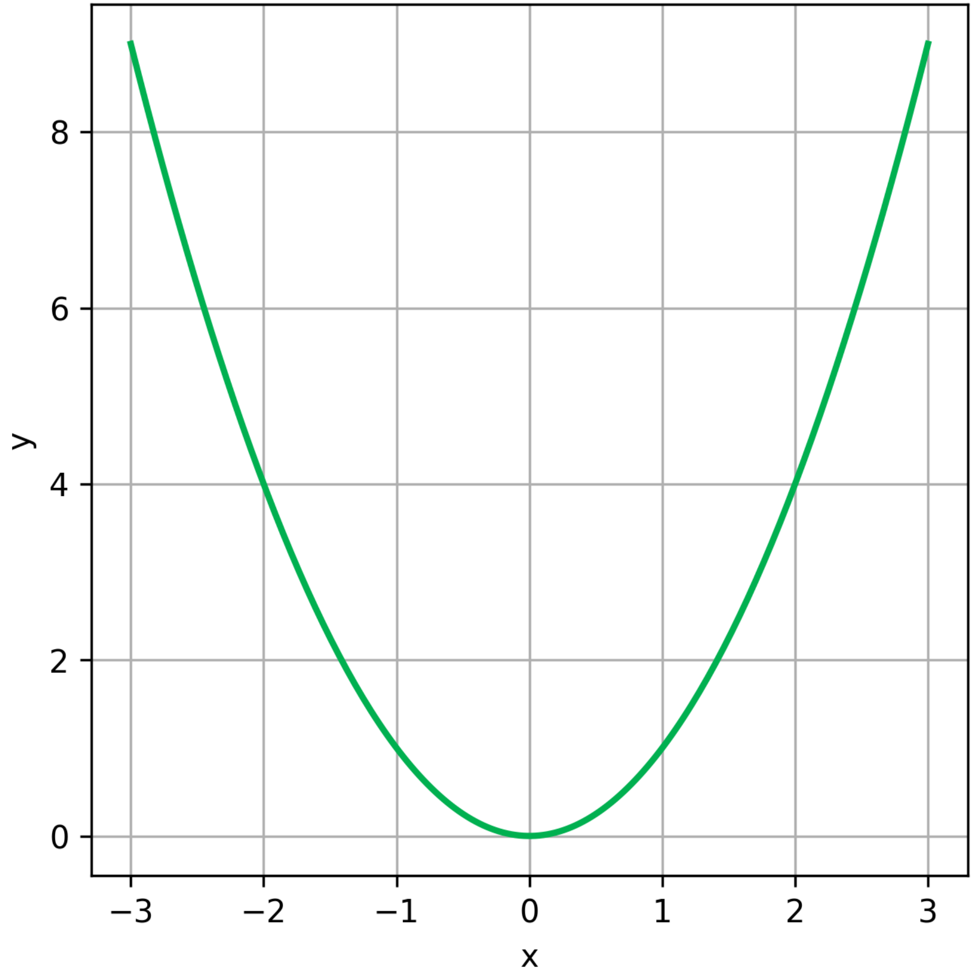
Minimize $f(x) = x^2$

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:

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

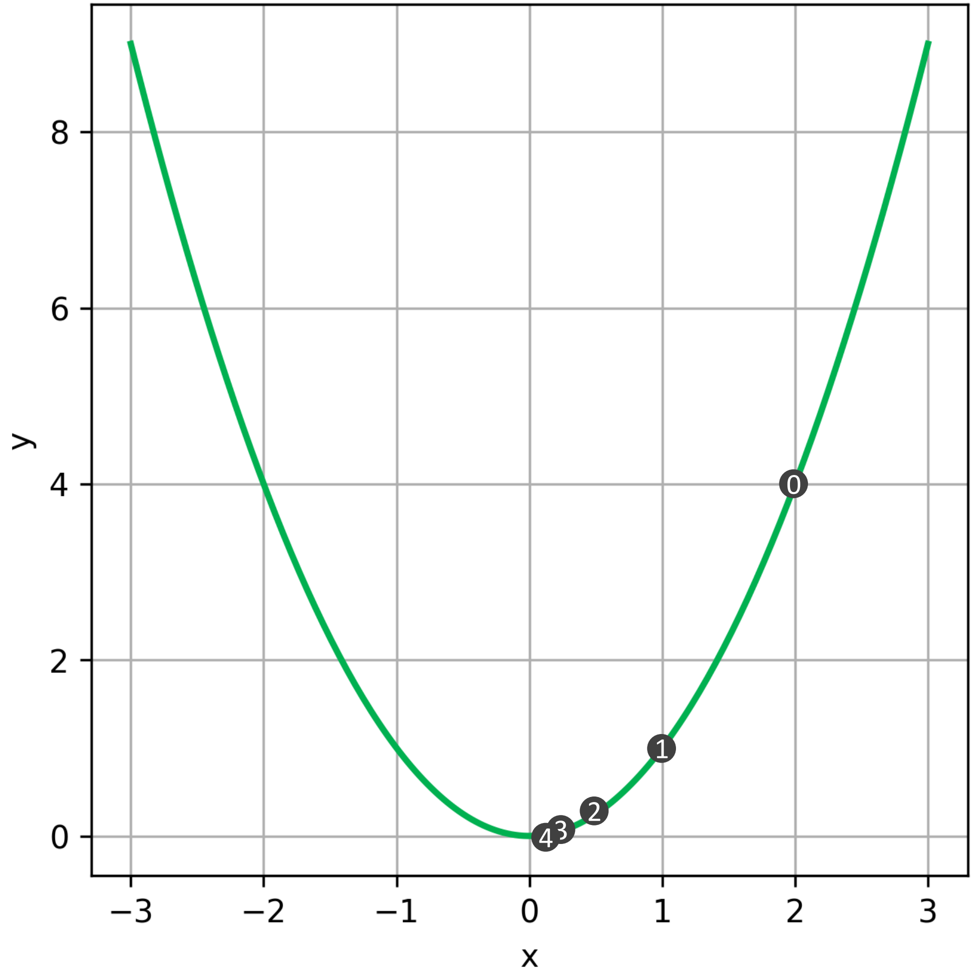
Minimize $f(x) = x^2$

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 much better suited for classification than linear regression

Logistic regression parameters must be estimated iteratively, and a method for that optimization is **gradient descent**

Gradient descent can be used for **cost function optimization** and there are a number of variants

Evaluating Model Performance

Supervised Learning Performance Evaluation

Regression

Classification

Binary

Receiver Operating Characteristic (ROC) curves

Multiclass

Confusion matrices

Common Metrics

Regression

- Mean squared error (MSE)
- Mean absolute error (MAE)
- R^2, coefficient of determination
- Adjusted R^2
- Explained variance

Binary

- Classification accuracy
- True positive rate
- False positive rate
- Precision
- F_1 Score
- Area under the ROC curve (AUC)

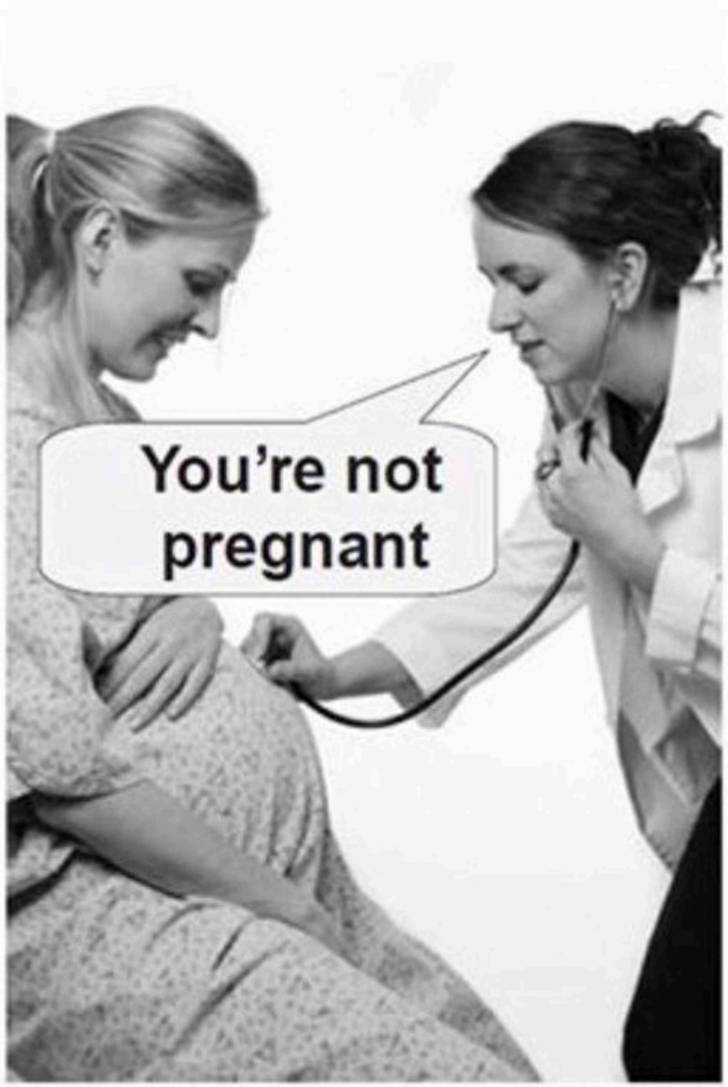
Multiclass

- Classification accuracy
- Micro-averaged F_1 Score
- Macro-averaged F_1 Score

Types of error

False Positive
(Type I error)

False Negative
(Type II error)



Binary Classification

True Class, y \ Predicted Class, $\hat{y}$	Class 1 (target)	Class 0 (non-target)
Class 1 (target)	true positive	false negative
Class 0 (non-target)	false positive	true negative

Type II Error
(missed target)

Type I Error

Binary Classification

Predicted Class, $\hat{y}$

		Class 1 (target)	Class 0 (non-target)
True Class, y	Class 1 (target)	true positive	false negative
	Class 0 (non-target)	false positive	true negative

True positive rate
Probability of detection, p_D
Sensitivity
Recall

$$\frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

How many targets (Class 1) were correctly classified as targets?

Binary Classification

		Predicted Class, $\hat{y}$: Class 1 (target)	Predicted Class, $\hat{y}$: Class 0 (non-target)
True Class, y	Class 1 (target)	true positive	false negative
True Class, y	Class 0 (non-target)	false positive	true negative

False positive rate
Probability of false alarm, p_{FA}

$$\frac{\text{false positive}}{\text{false positive} + \text{true negative}}$$

How many non-targets (Class 0) were incorrectly classified as targets?

Binary Classification

Predicted Class, $\hat{y}$

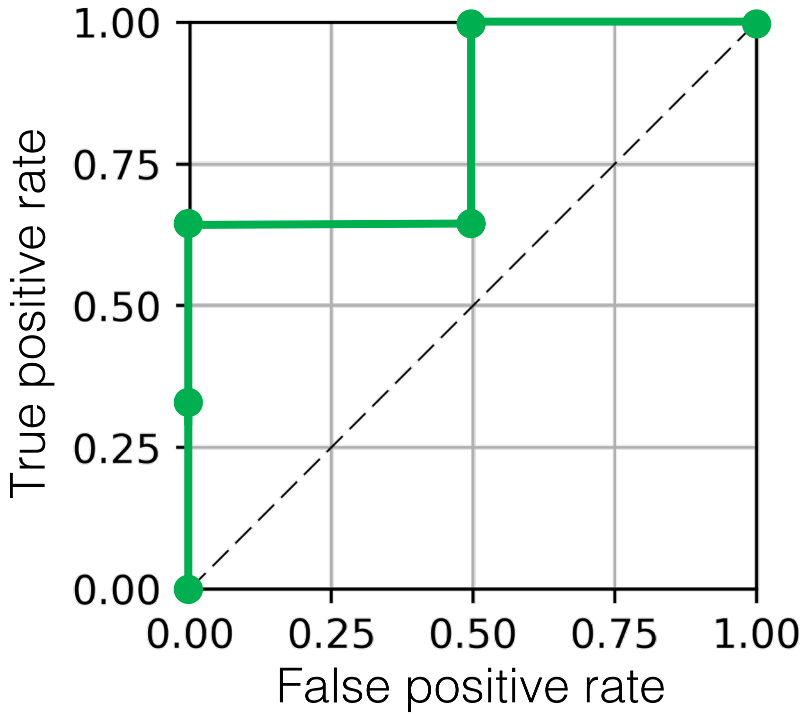
True Class, y		Class 1 (target)	Class 0 (non-target)
	Class 1 (target)	true positive	false negative
	Class 0 (non-target)	false positive	true negative

Precision

$$\frac{\text{true positive}}{\text{true positive} + \text{false positive}}$$

How many of the predicted targets are targets?

ROC Curves



Estimate ($\hat{y}$)	True Class Label (y)	Classifier Confidence
1	1	1.40
1	1	0.95
1	0	0.80
1	1	0.60
1	0	-0.10

Classifier decision rule:

$$\hat{y} = \begin{cases} 1, & \text{confidence score} > \text{thresh} \\ 0, & \text{confidence score} \leq \text{thresh} \end{cases}$$

$$AUC = \left(\frac{2}{3}\right)\left(\frac{1}{2}\right) + (1)\left(\frac{1}{3}\right) = \frac{5}{6} \cong 0.833$$

$$\frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

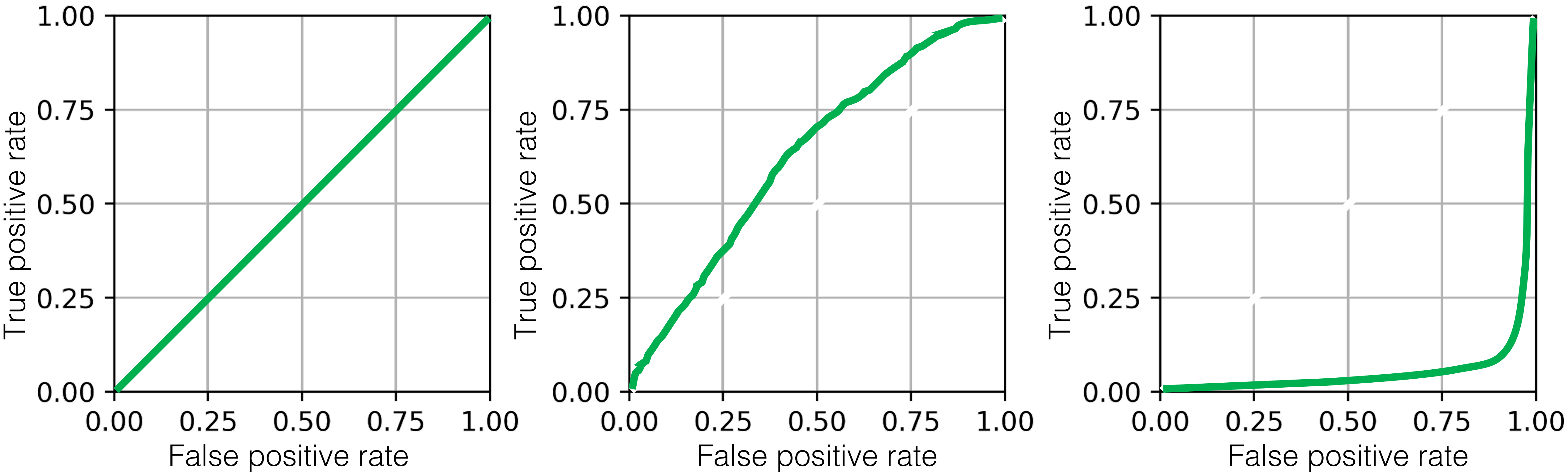
$$\frac{\text{false positive}}{\text{false positive} + \text{true negative}}$$

Total Positives = 3

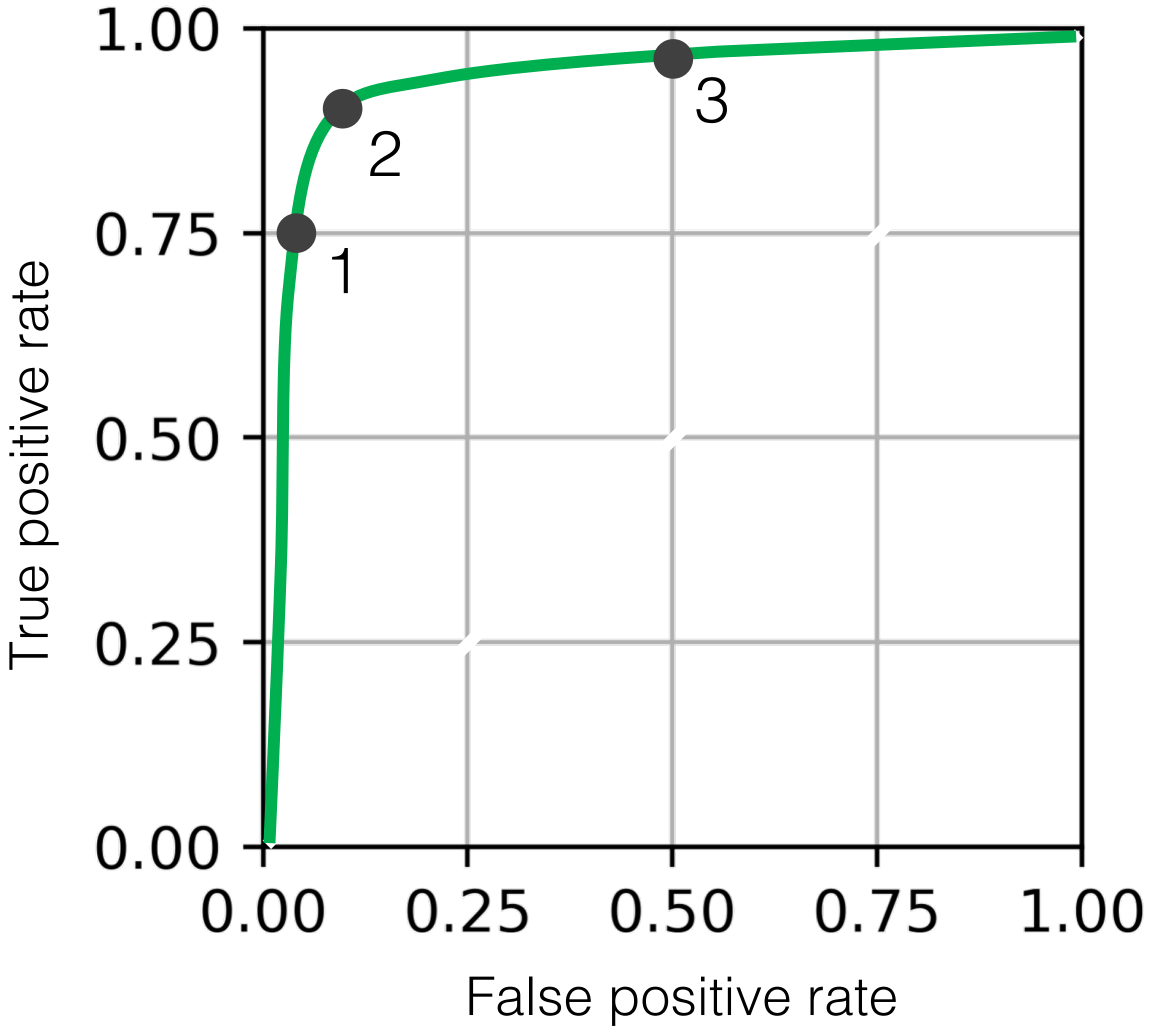
Total Negatives = 2

Threshold	# True Positives	True Positive Rate	# False Positives	False Positive Rate

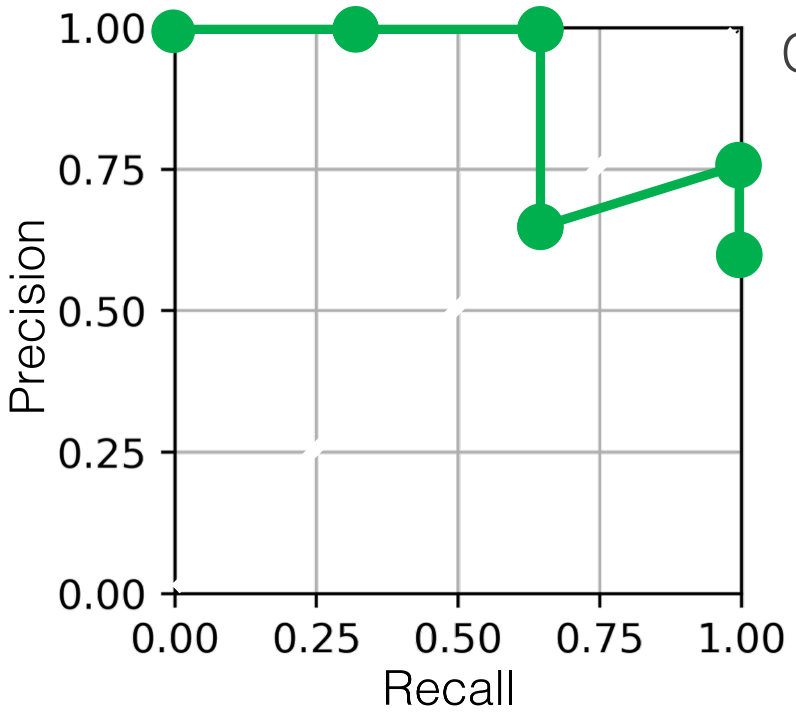
ROC Curves: how do they compare?



ROC Curves: where do we operate?



PR Curves



Classifier decision rule:

$$\hat{y} = \begin{cases} 1, \text{confidence score} > \text{thresh} \\ 0, \text{confidence score} \leq \text{thresh} \end{cases}$$

Estimate ($\hat{y}$)	True Class Label (y)	Classifier Confidence
1	1	1.40
1	1	0.95
1	0	0.80
1	1	0.60
1	0	-0.10

$$\frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

$$\frac{\text{true positive}}{\text{true positive} + \text{false positive}}$$

Total Positives = 3

Total Negatives = 2

Threshold	# True Positives	Recall	# Predicted Positive	Precision

Case study 1

i	y_i	$\hat{y}_i$
1	1	1
2	1	1
3	1	1
4	1	1
5	1	1
6	1	1
7	1	0
8	0	1
9	0	0
10	0	0
11	0	0
12	0	0
13	0	0
14	0	0
15	0	0

Overall classification accuracy = 13/15 = **0.87**

ROC Curves measure the tradeoff between…

A False positive rate = 1/8 = **0.13**

B True positive rate (Recall) = 6/7 = **0.86**

PR Curves measure the tradeoff between…

B True positive rate (Recall) = 6/7 = **0.86**

C Precision= 6/7 = **0.86**

A
$$\frac{\text{false positive}}{\text{false positive} + \text{true negative}}$$

B
$$\frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

C
$$\frac{\text{true positive}}{\text{true positive} + \text{false positive}}$$

Case study II

i	y_i	$\hat{y}_i$
1	**1**	**1**
2	**1**	**1**
3	**1**	**0**
4	**1**	**0**
5	**0**	**0**
6	**0**	**0**
7	**0**	**0**
8	**0**	**0**
9	**0**	**0**
10	**0**	**0**
11	**0**	**0**
12	**0**	**0**
13	**0**	**0**
14	**0**	**0**
15	**0**	**0**

Overall classification accuracy = 13/15 = **0.87**

ROC Curves measure the tradeoff between…

A False positive rate = 0/11 = **0**

$$\text{A} = \frac{\text{false positive}}{\text{false positive} + \text{true negative}}$$

B True positive rate (Recall) = 2/4 = **0.5**

$$\text{B} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

PR Curves measure the tradeoff between…

B True positive rate (Recall) = 2/4 = **0.5**

C Precision= 2/2 = **1**

$$\text{C} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}$$

Case study III

i	y_i	$\hat{y}_i$
1	1	1
2	1	1
3	1	1
4	1	1
5	1	1
6	1	1
7	1	1
8	1	1
9	1	1
10	1	1
11	1	1
12	1	1
13	1	1
14	0	1
15	0	1

Overall classification accuracy = 13/15 = **0.87**

ROC Curves measure the tradeoff between…

A False positive rate = **2/2 = 1**

B True positive rate (Recall) = **13/13 = 1**

PR Curves measure the tradeoff between…

B True positive rate (Recall) = **13/13 = 1**

C Precision= **13/15 = 0.87**

A $\dfrac{\text{false positive}}{\text{false positive} + \text{true negative}}$

B $\dfrac{\text{true positive}}{\text{true positive} + \text{false negative}}$

C $\dfrac{\text{true positive}}{\text{true positive} + \text{false positive}}$

Multiclass Classification: Confusion Matrix

Predicted Class, $\hat{y}$

True Class, y

	Class 1	Class 2	Class 3	No. samples from class
Class 1	190	8	2	[200]
Class 2	1	5	4	[10]
Class 3	24	24	25	[73]

confusion matrix with number of samples

F_1-score

$$F_1 = 2\frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}}$$

Harmonic mean of precision and recall

$$= 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Generally:

$$F_\beta = (1 + \beta^2)\frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$$

β controls the relative weight of precision/recall

Training, Test Split

Learning model parameters

Training	Test
Learn model parameters	Evaluate generalization performance

For small datasets, this reduction in dataset size may be detrimental

Training, Validation, Test Split

Learning parameters AND hyperparameters

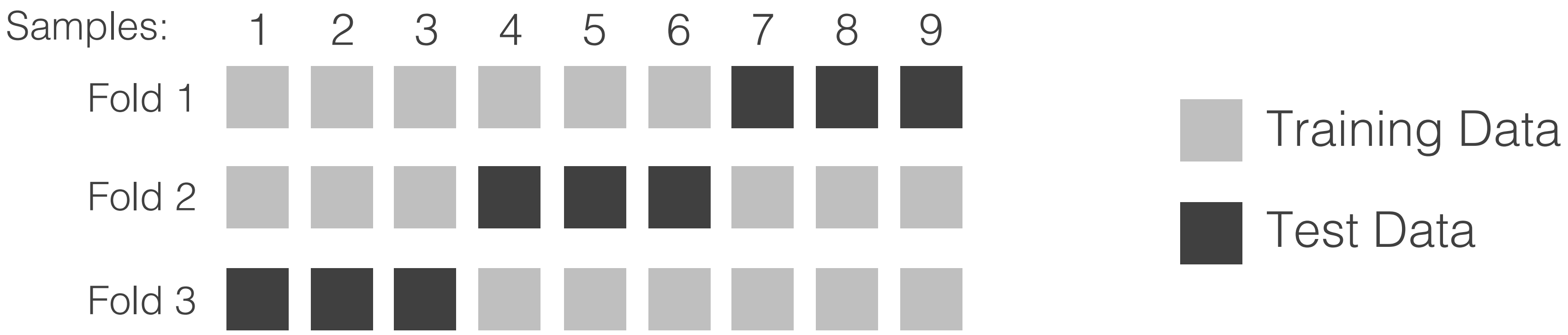
Training	Validation	Test
Learn model parameters	Learn hyperparameters	Evaluate generalization performance

Hyperparameters: parameters of your learning algorithm or parameters of you model that are set before training begins

Simple cross-validation

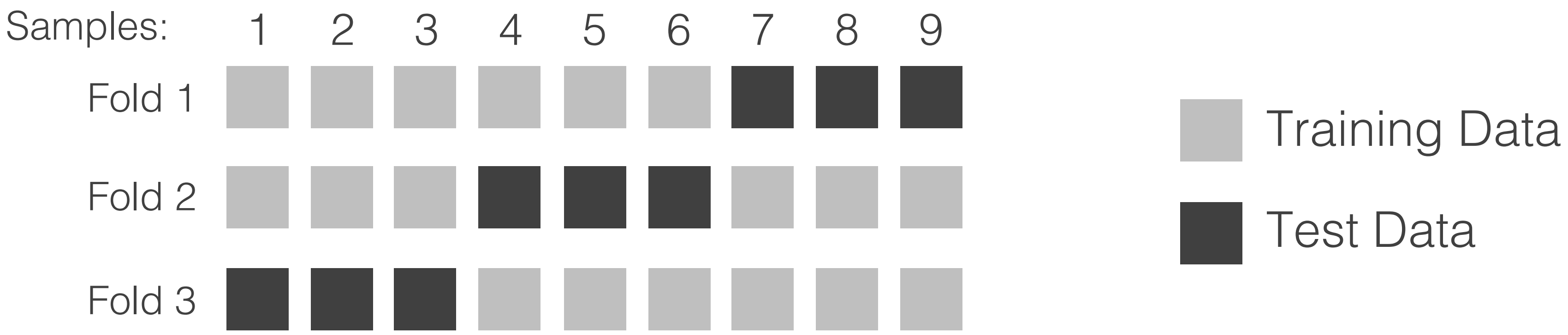
K-fold cross validation **K = 3**

❶ **Performance evaluation**: Train your model K times, once for each fold

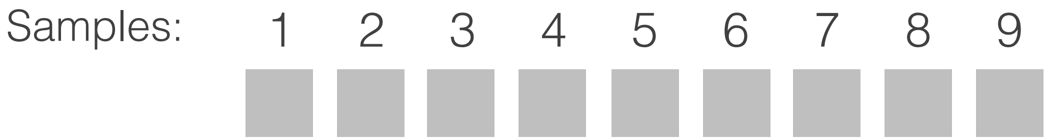


After performance has been validated, train on all the data you have before you apply the model in practice

1 **Performance evaluation**: Train your model K times, once for each fold



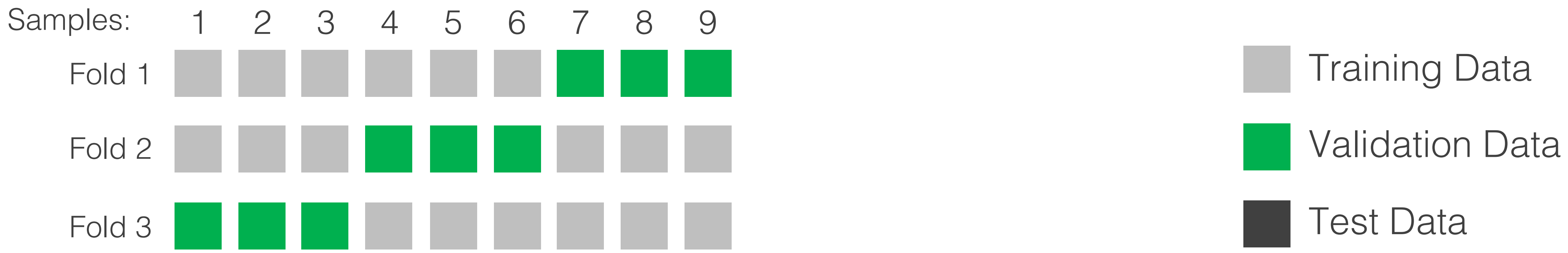
2 **Model application**: Once you've evaluated model performance and are ready apply the model then retrain the model on ALL of your data to prepare it for unseen data



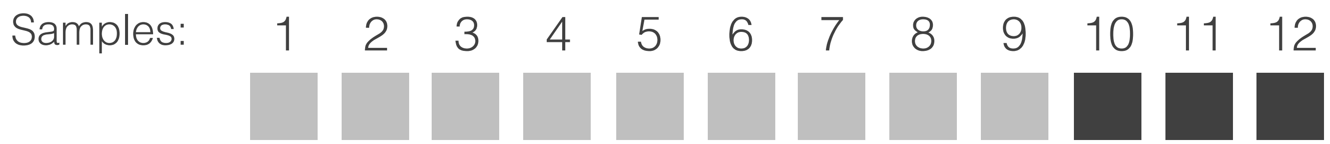
(this is not a model evaluation step, but only when you're ready to apply in practice)

Cross-validation with hyperparameters

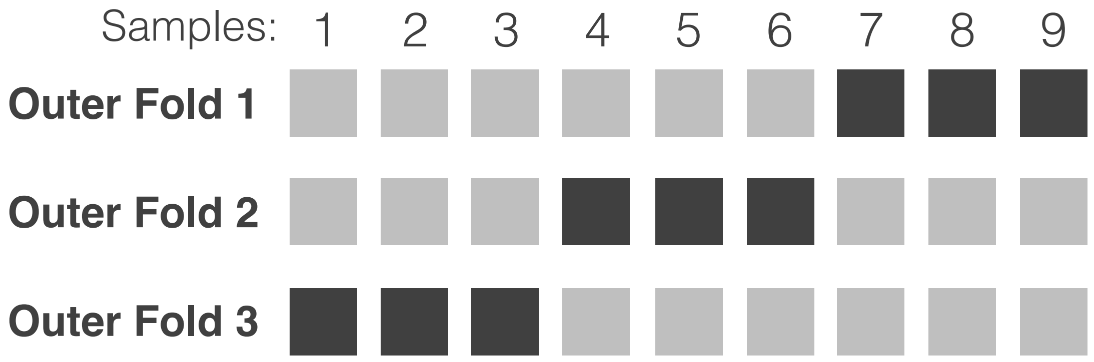
1. Repeatedly fit your model to your K folds. Each iteration try different hyperparameters

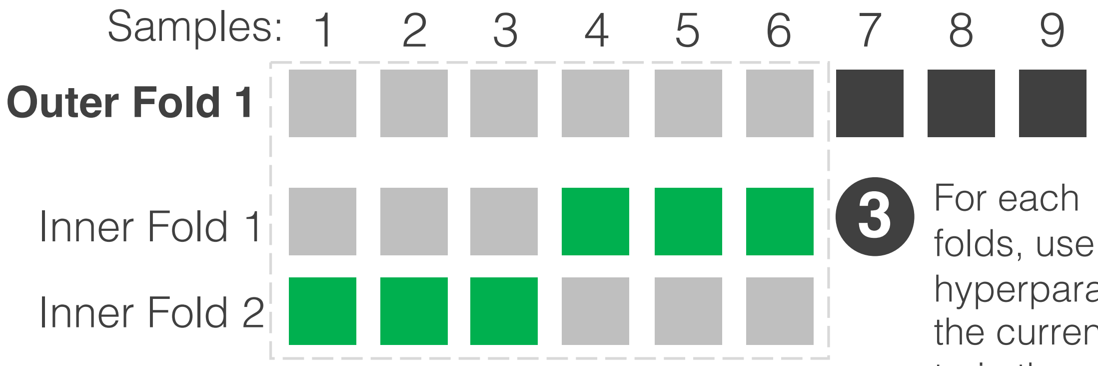


2. Using the best-performing hyperparameters from (a), train on all training data and evaluate performance on the test data



Nested cross-validation with hyperparameters





1 For each outer fold, train your model with the best-performing hyperparameters from the inner folds

2 For the inner folds, repeatedly run K-folds, each time with a different set of hyperparameters

3 For each of the K inner folds, use the hyperparameters from the current iteration to train the model, then evaluate performance on validation data

4 Repeat steps (2) and (3) for the remaining outer folds

Training Data

Validation Data

Test Data