Carnegie Mellon University
Heinz College

95-865 Unstructured Data Analytics

Recitation: More on Hyperparameter Tuning and Model Evaluation

Slides by George H. Chen

(Flashback) Hyperparameter Tuning in General (Not Just for *k*-NN Classifier)

Suppose that we have a classifier with hyperparameter setting θ

could consist of multiple hyperparameters
(think of θ as a tuple)

For each hyperparameter setting θ (in a list of hyperparameter settings we are willing to try):

1. Train classifier on proper training data using hyperparameter setting θ
2. Use a score function to evaluate how well the trained model predicts on validation data

Use classifier corresponding to whichever value of θ achieves the best score

⚠ How we randomly split the training data into proper training/validation sets affects the scores we get

⚠ If the classifier's training procedure is random, then using different random seeds could also change the scores we get

(Flashback) Hyperparameter Tuning in General (Not Just for *k*-NN Classifier)

Suppose that we have a classifier with hyperparameter setting θ

could consist of multiple hyperparameters
(think of θ as a tuple)

For each hyperparameter setting θ (in a list of hyperparameter settings we are willing to try):

1. Train classifier on proper training data using hyperparameter setting θ
2. Use a score function to evaluate how well the trained model predicts on validation data

Use classifier corresponding to whichever value of θ achieves the best score

⚠ How we randomly split the training data into proper training/validation sets affects the scores we get

⚠ If the classifier's training procedure is random, then using different random seeds could also change the scores we get

Which score function is used for measuring accuracy matters!

Score Functions for Accuracy

What we already saw:

- **Raw accuracy:** fraction of predicted labels that are correct

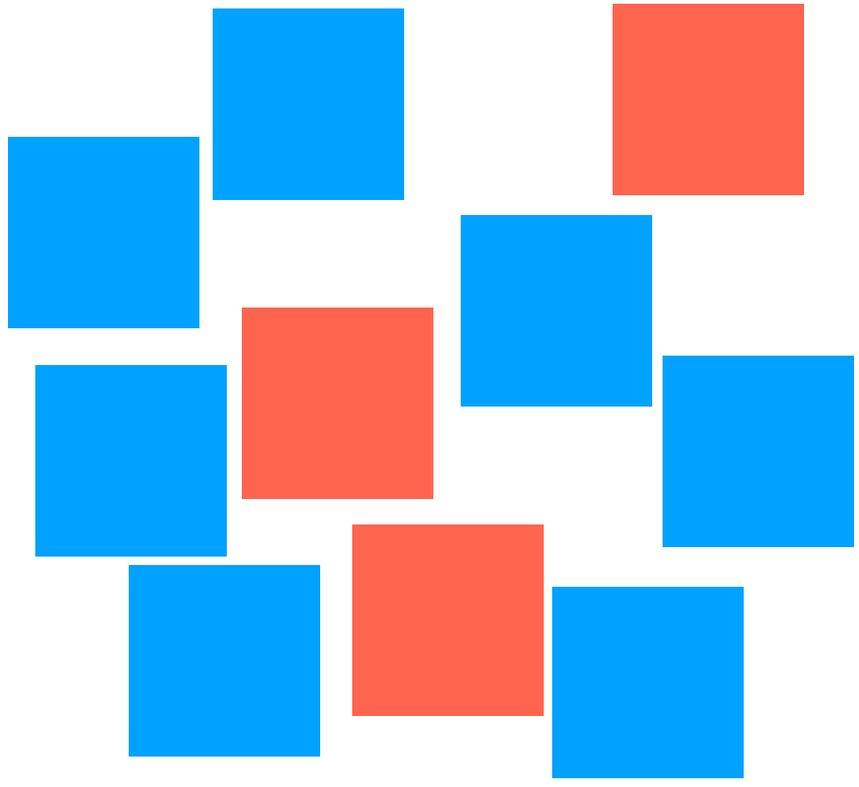
In “binary” classification (there are 2 classes such as spam/ham) when 1 class is considered “positive” and the other “negative”:

Score Functions for Accuracy

What we already saw:

- **Raw accuracy:** fraction of predicted labels that are correct

In “binary” classification (there are 2 classes such as spam/ham) when 1 class is considered “positive” and the other “negative”:

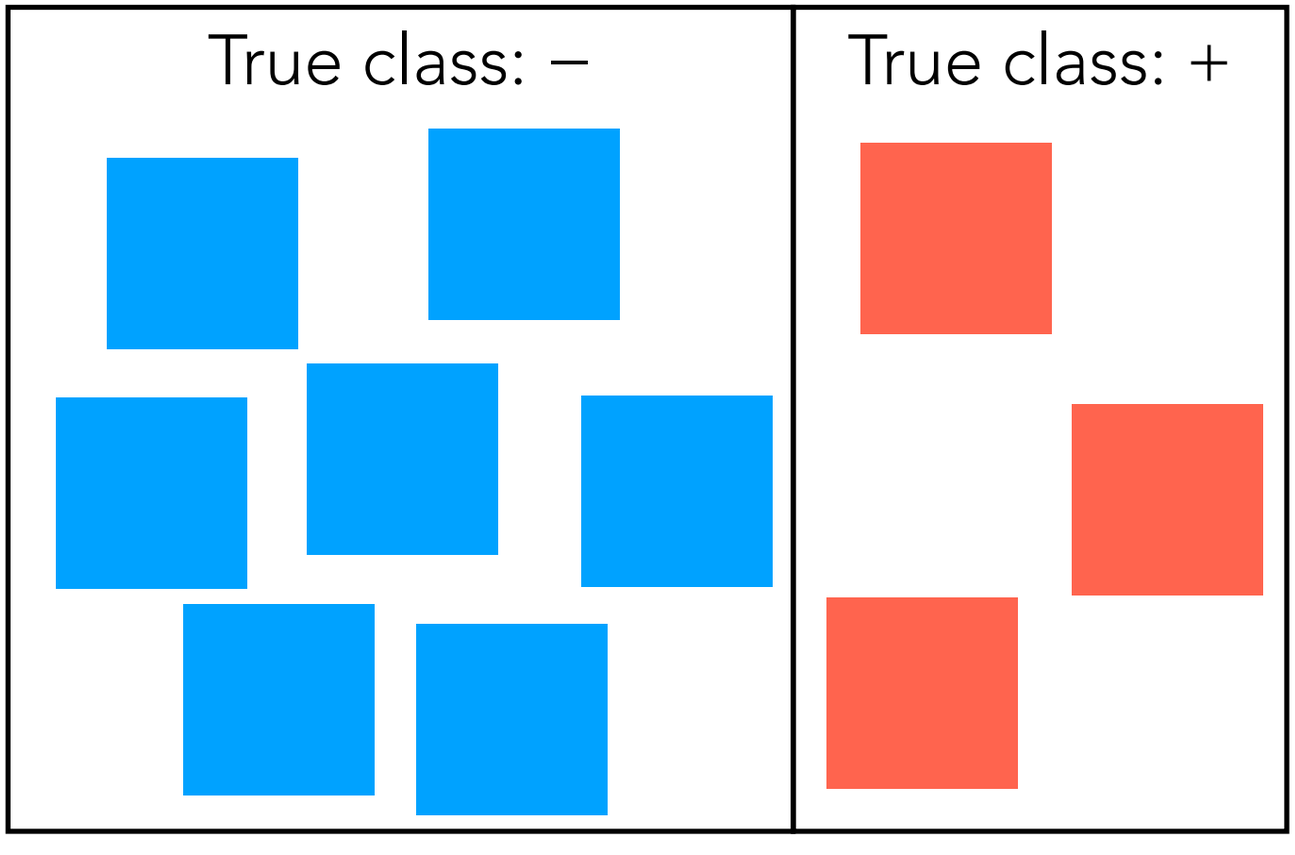


Score Functions for Accuracy

What we already saw:

- **Raw accuracy:** fraction of predicted labels that are correct

In "binary" classification (there are 2 classes such as spam/ham) when 1 class is considered "positive" and the other "negative":



Score Functions for Accuracy

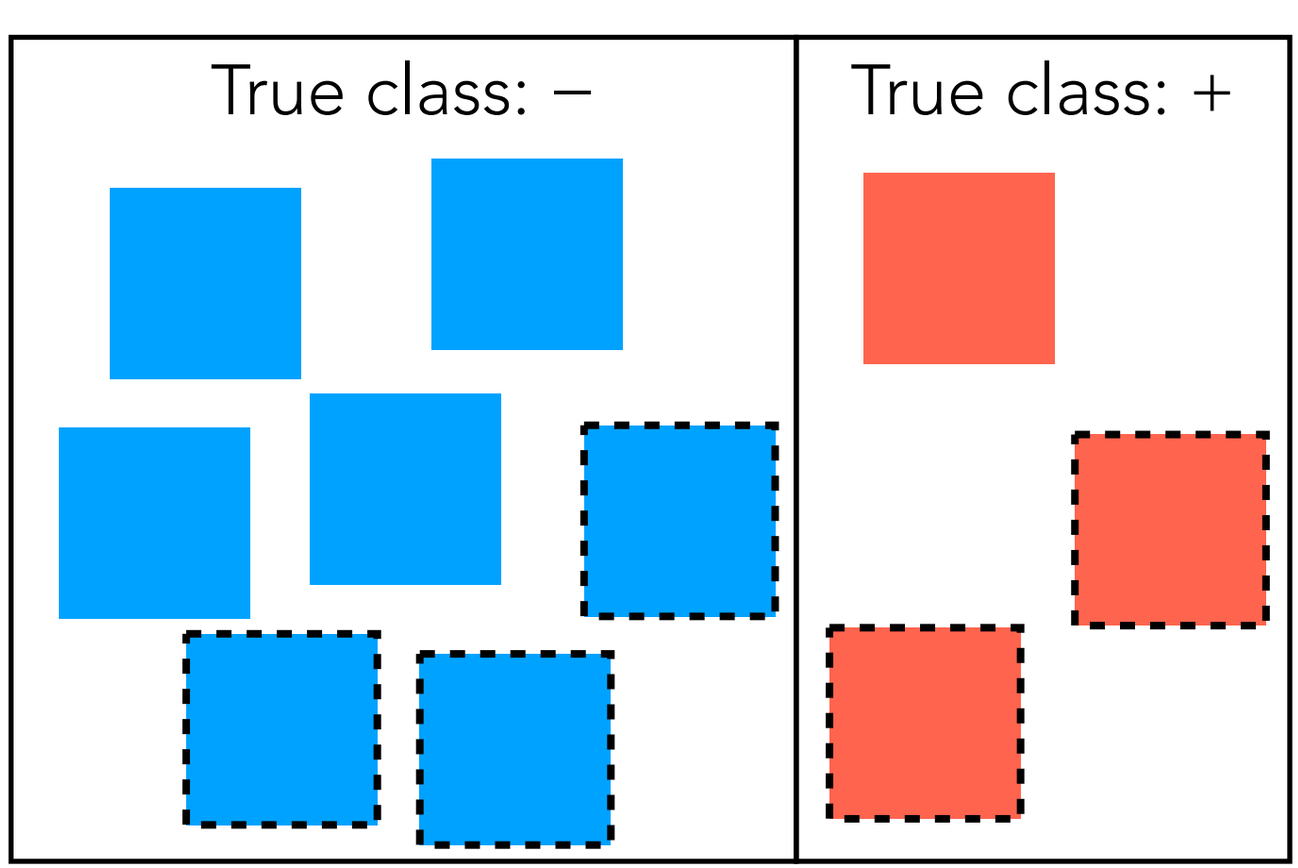
What we already saw:

- **Raw accuracy:** fraction of predicted labels that are correct

In "binary" classification (there are 2 classes such as spam/ham) when 1 class is considered "positive" and the other "negative":

Outlined in dotted black: predicted label +

(all other points predicted to be –)



Recall/True Positive Rate: fraction of red points correctly predicted = 2/3

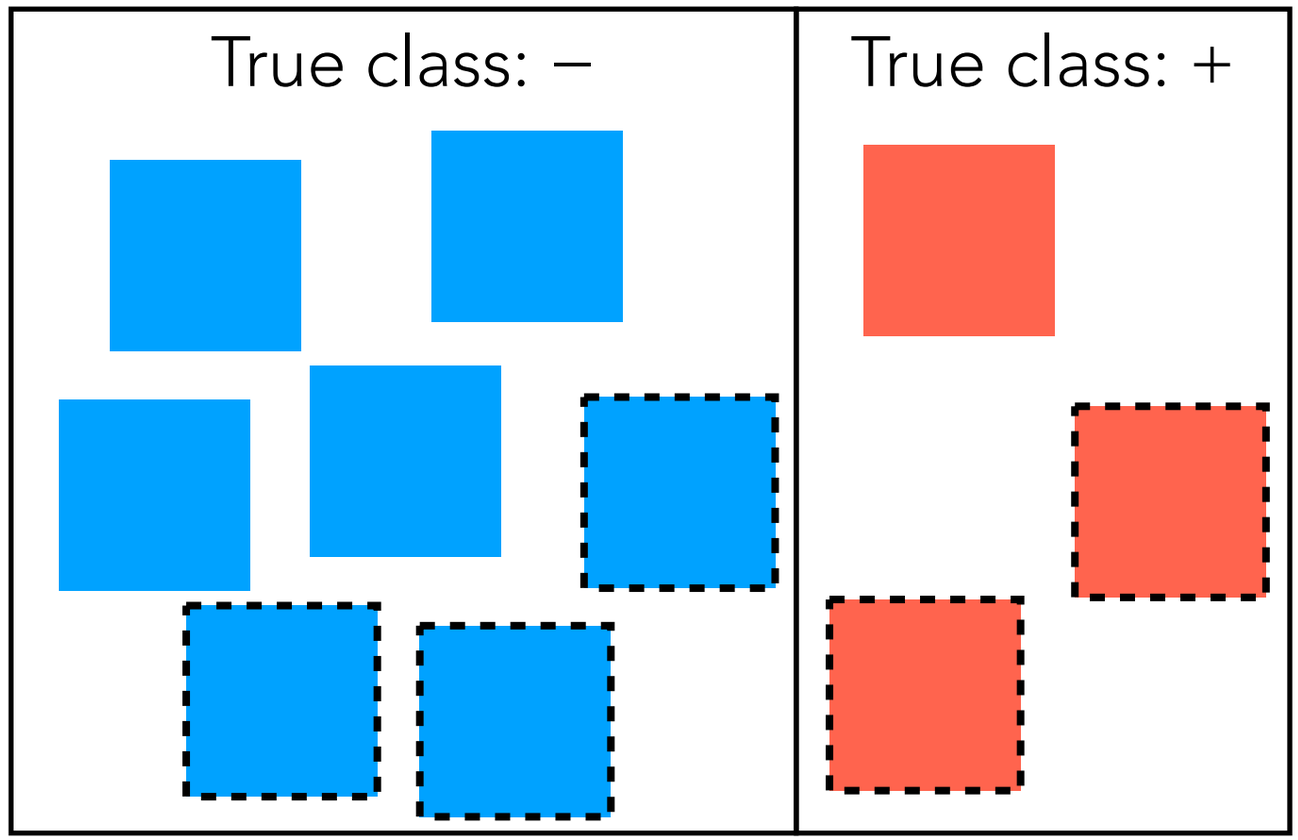
Precision: fraction of dotted points correctly predicted

Score Functions for Accuracy

In "binary" classification (there are 2 classes such as spam/ham) when 1 class is considered "positive" and the other "negative":

Outlined in dotted black: predicted label +

(all other points predicted to be –)



Recall/True Positive Rate: fraction of red points correctly predicted = 2/3

Precision: fraction of dotted points correctly predicted

False Positive Rate: fraction of blue points incorrectly predicted = 3/7

F1 score: $\dfrac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = 1/2$

Generalizing F1 Score to More Than 2 Classes

For each class $c \in \mathcal{C}$: ($\mathcal{C}$: set of possible classes)

- Treat class c as the positive class and compute the F1 score

  Denote the resulting F1 score as: $F_1^{(c)}$

How do we aggregate across the different classes' F1 scores to produce a single number as an overall score?

Option #1: report an equally weighted average across classes

$$F_1^{\text{equally weighted}} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} F_1^{(c)}$$

Option #2: weight each class by how often it appears in the data that we're evaluating the F1 score for

$$F_1^{\text{weighted}} = \sum_{c \in \mathcal{C}} [\text{fraction of points in class } c] \times F_1^{(c)}$$

“Receiver Operating Characteristic” (ROC) Curves

Probability Thresholding

Recall that logistic regression predicts the probability of each class for any test feature vector ***x***

(MNIST: for any test image, we predict probabilities for all 10 digits)

To get final predicted class of test feature vector ***x***:
pick whichever class has the highest probability

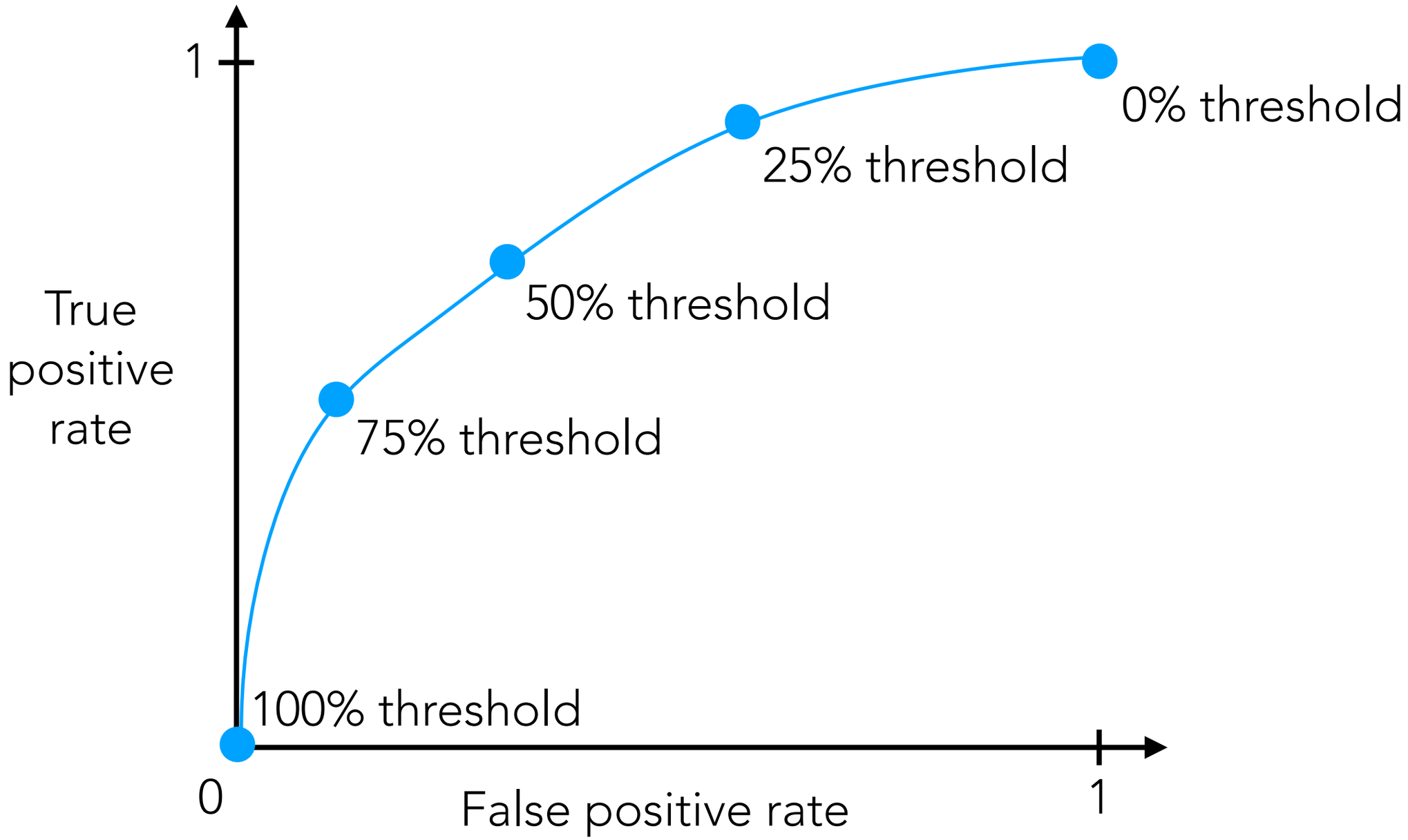
When there are 2 classes positive and negative

Predict positive if P(positive | test feature vector ***x***) ≥ 0.5

Predict negative otherwise

We can vary this 50% threshold!

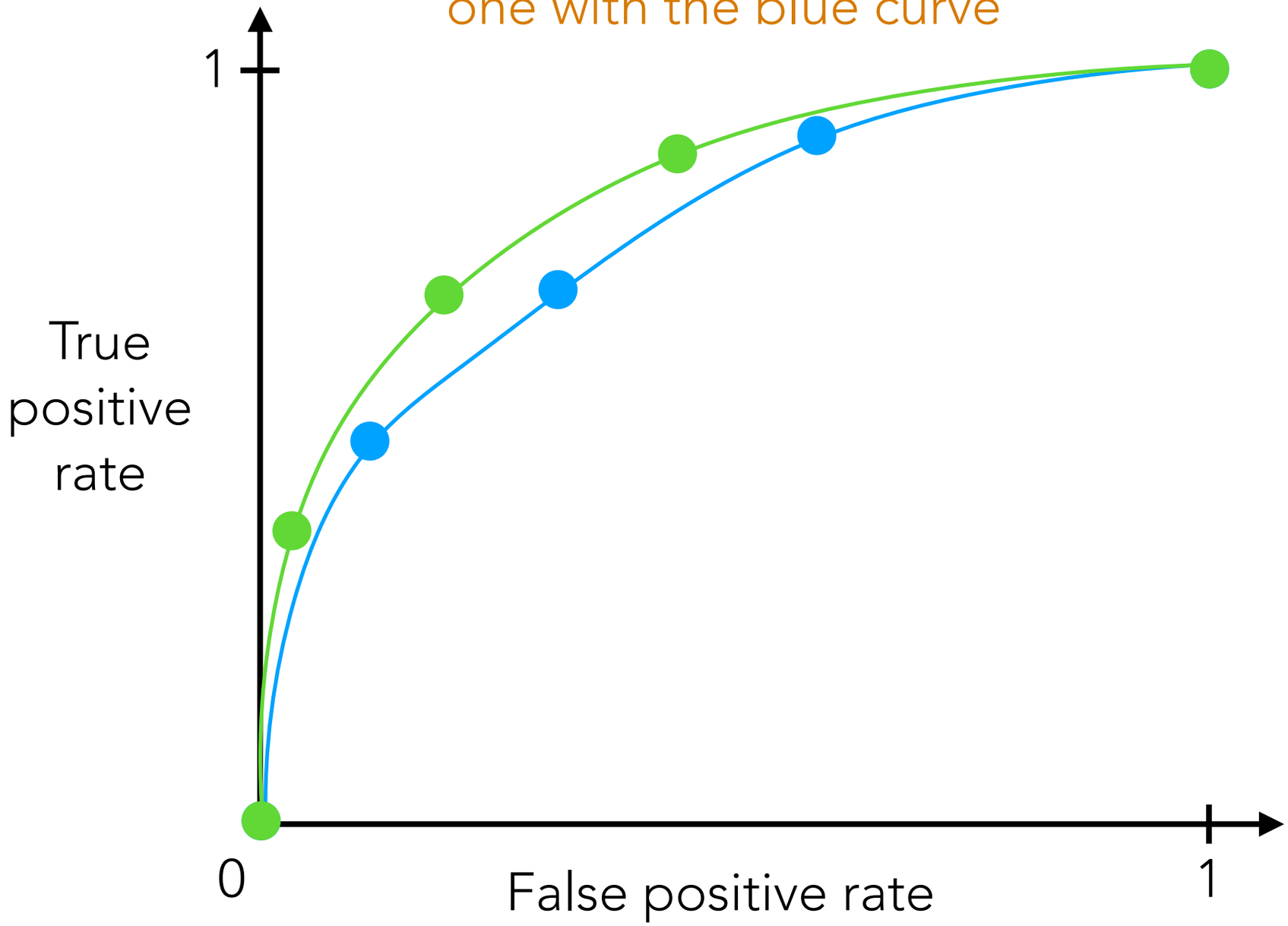
Binary Classification: ROC Curves



TPR and FPR are computed using test data

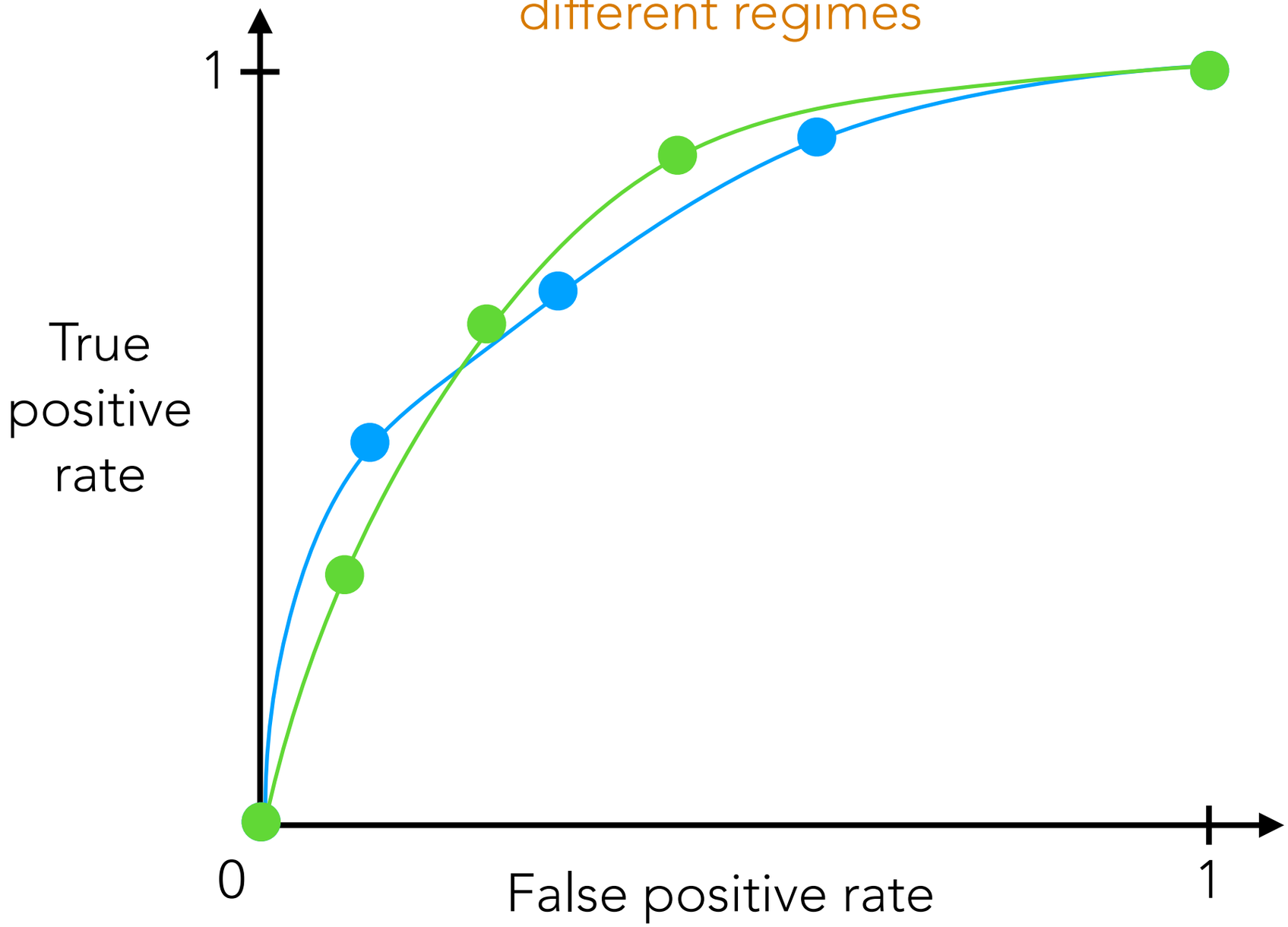
Binary Classification: ROC Curves

A classifier with the green curve is better than the one with the blue curve

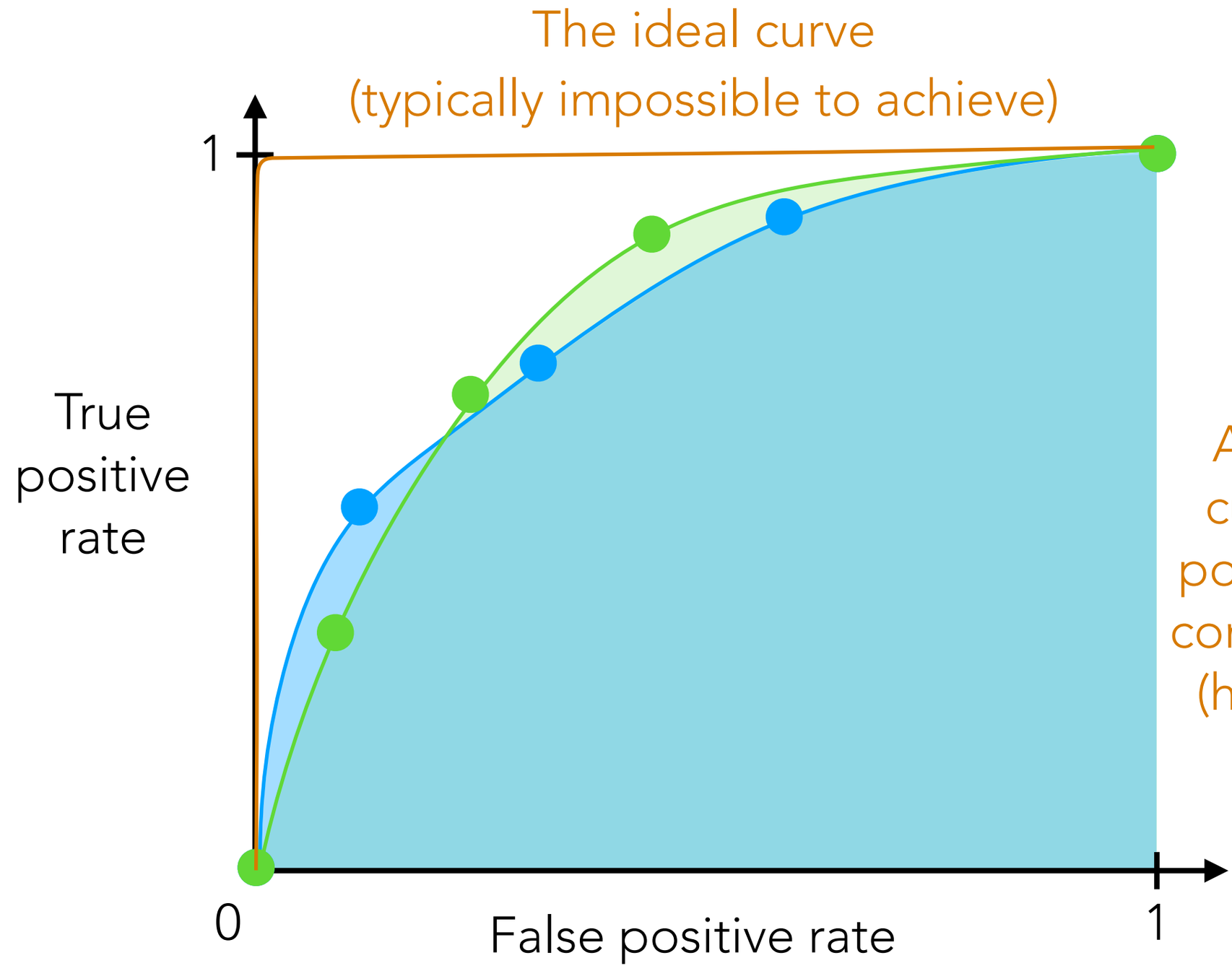


Binary Classification: ROC Curves

It's possible that different models are better in different regimes



Binary Classification: ROC Curves



Area under the curve (AUC) is a popular metric for comparing models (higher is better)

Binary Classification: ROC Curves

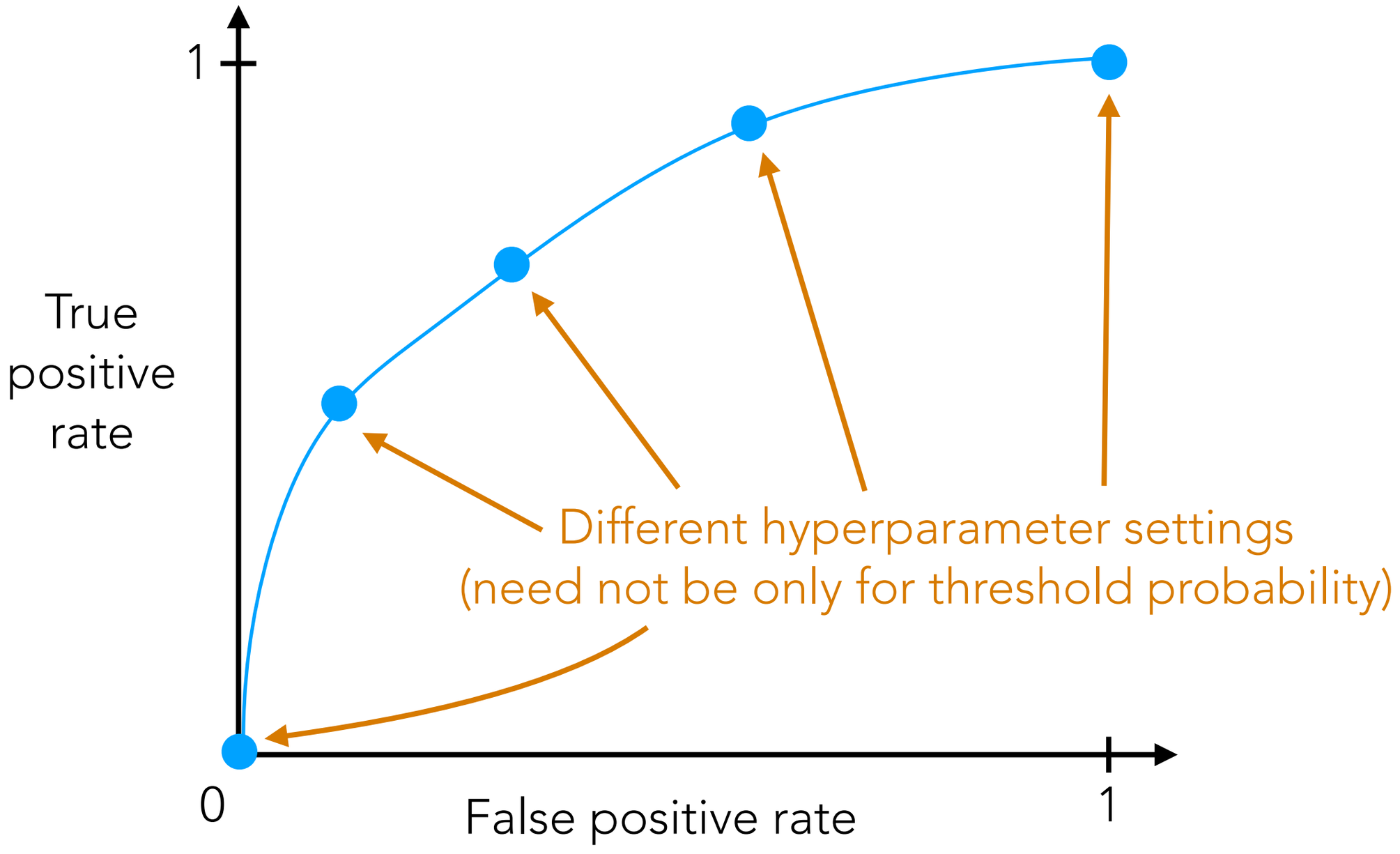
What we just saw:

- For a classifier that we can set the threshold probability to different values, we can plot an ROC curve
- True positive rate (TPR) and false positive rate (FPR) are evaluated on test data

Other variants are possible:

- Plot precision vs recall instead of TPR vs FPR
- Can actually plot ROC/precision-recall curves sweeping over hyperparameters *aside from threshold probability*!
- For ROC/precision-recall, rather than evaluating on test data, can evaluate on validation data during training *to help choose hyperparameters*

Binary Classification: ROC Curves



Can also be computed on validation data instead of test data!