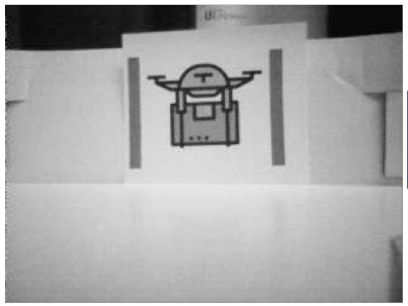
Lecture 9

Classification

CS 3630

Image Processing Pipeline



Original image
(76,800 pixels)

Preprocessing

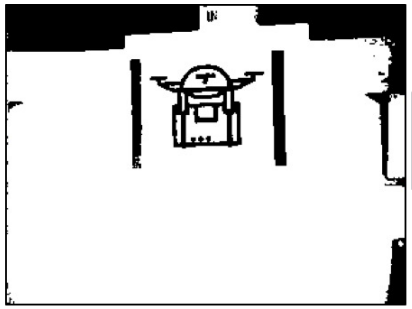


Image modified to make feature extraction easier or more consistent
(still 76,800 pixels)

Feature Extraction

Q	A	S	D	F	G	H
1	0	2	3	7	8	6
2	4	5	3	2	1	4
6	8	9	7	5	1	3
2	1	3	3	5	4	3
6	8	7	8	9	1	3
7	4	7	0	9	2	3

Table of image features
(somewhere in the range of 100-3000)

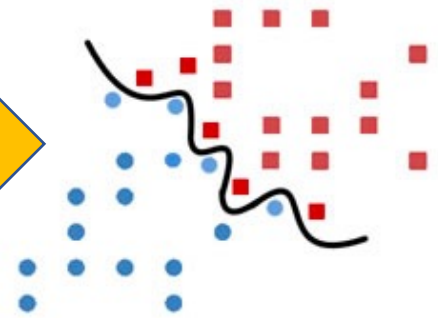
Image Classification

Q	A	S	D	F	G	H
1	0	2	3	7	8	6
2	4	5	3	2	1	4
6	8	9	7	5	1	3
2	1	3	3	5	4	3
6	8	7	8	9	1	3
7	4	7	0	9	2	3

+

label
truck
drone
drone
hands
truck
order

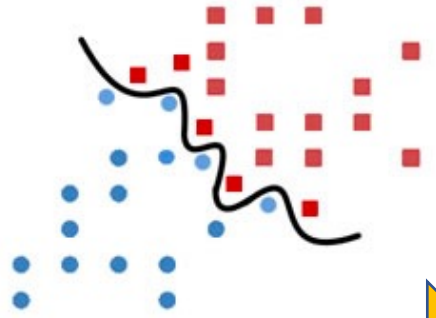
Train Model



new image

0	4	5	7	1	9

+



Classify new image

“drone”

Supervised Learning

- Given a set of data points with an outcome (a.k.a. *labels*), create a model to describe them

- Classification

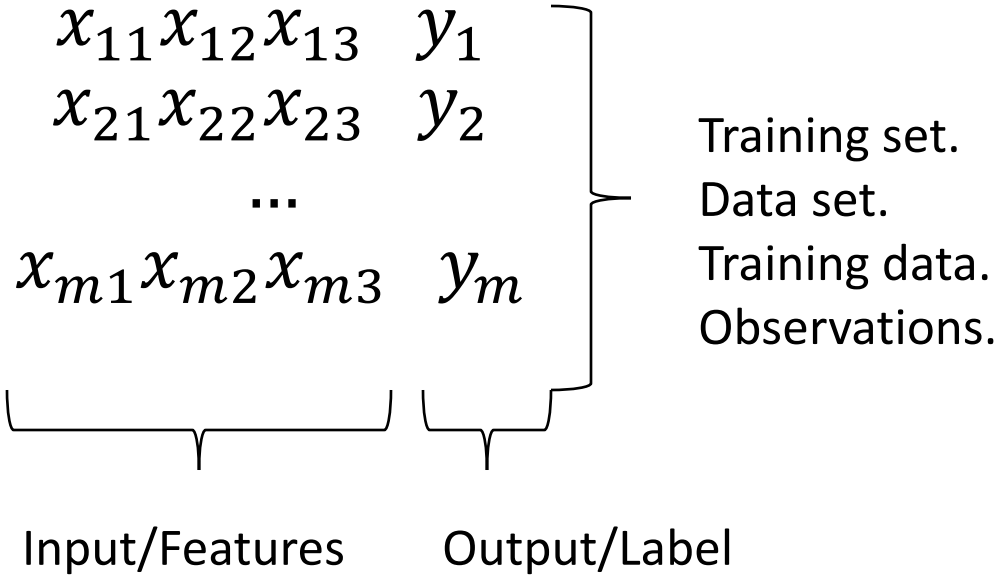
  - Outcome: a discrete variable (typically <10 outcomes)

- Regression
  - Outcome: continuous prediction

Other kinds of learning?

- Reinforcement
- Self-supervised
- Unsupervised
- Hybrid...

Supervised Learning: Training Data



- Each y_i was generated by equation $f(x_i) = y_i$, or more generally $f(x) = y$.
- Supervised learning aims to discover a function $h(x)$ that approximates $f(x)$. h is called a hypothesis. (sometimes it's also just called f even though we know it's just an estimate)

What else do we have?

- In real life, we usually don’t have just a set of data
  - Also have background knowledge, theory about the underlying processes, etc.

- For now, we will assume **just the data** (this is called *inductive learning*)
  - Cleaner and a good base case.
  - More complex mechanisms needed to reason with prior knowledge (more on this later).

Inductive Learning

- Given a set of observations come up with a model, h, that describes them.

- What does "describes" mean?
  - h is the same as the function f that generated them

How can we pick the right function?

So many choices and challenges!

- There could be multiple models to generate the data ...
- Examples do not completely describe the function ...
- Search space is large ...
- Would we know if we got the right answer?

How can we pick the right function?

So many choices and challenges!

- There could be mu… …enerate the data …
- Examples do not com… …ribe the function …
- Search space is large
- Would we know if … …swer?

Stop! Not the right way of thinking about the problem!

Inductive Learning

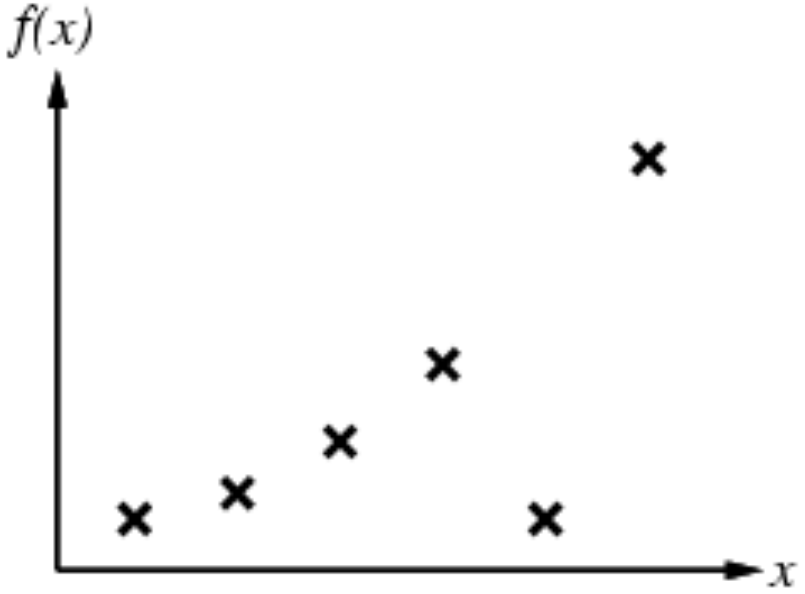
- Given a set of observations come up with a model, h, that describes them

- What does "describes" mean?
  - ~~h is the same as the function f that generated them~~
  - h models the observations well, and is *likely to predict future observations well.*

Inductive Learning

- Construct/adjust h to agree with f on training data

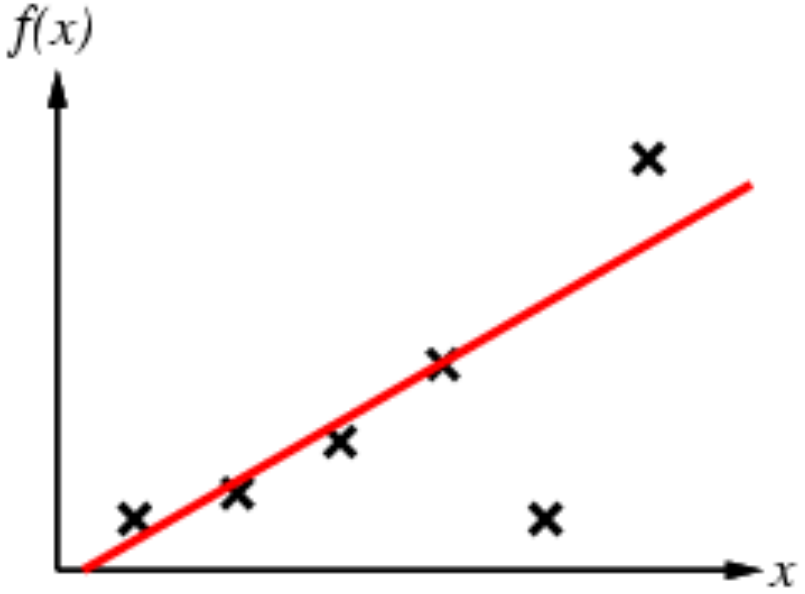
- E.g., curve fitting:



Inductive Learning

- Construct/adjust h to agree with f on training data

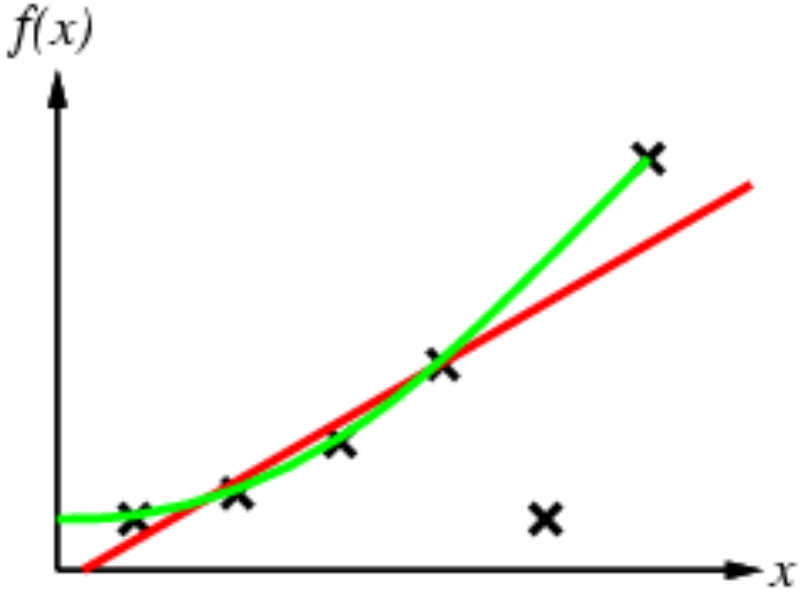
- E.g., curve fitting:



Inductive Learning

- Construct/adjust h to agree with f on training data

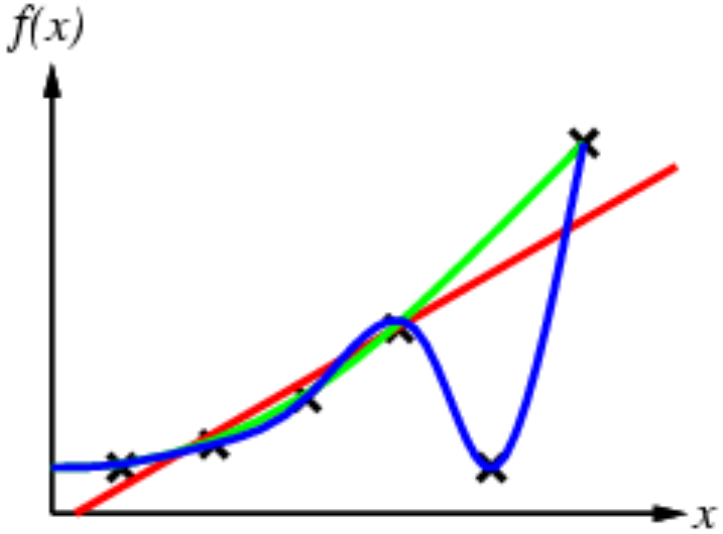
- E.g., curve fitting:



Inductive Learning

- Construct/adjust h to agree with f on training data

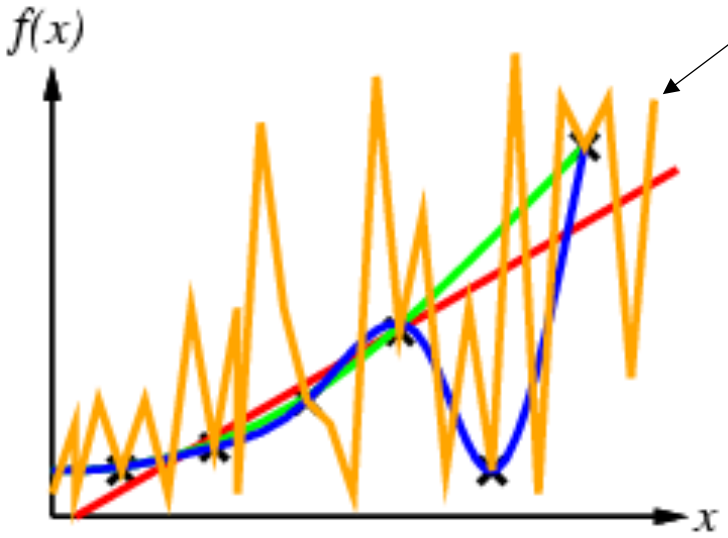
- E.g., curve fitting:



Inductive Learning

- Construct/adjust h to agree with f on training data

- E.g., curve fitting:

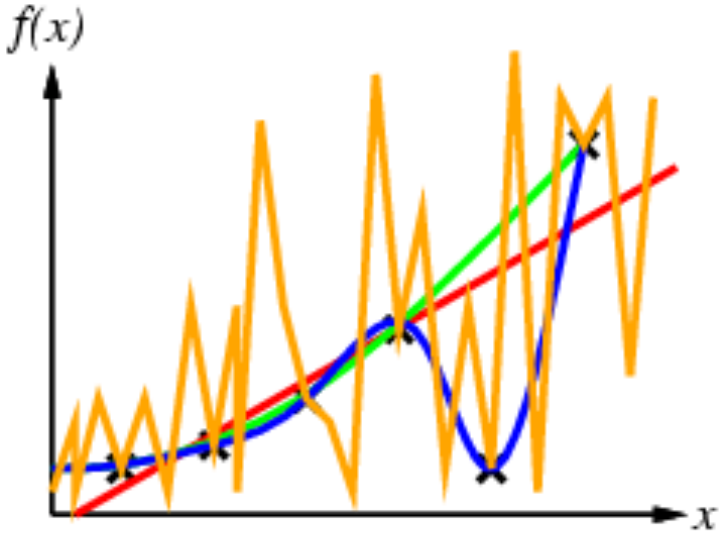


Overfitting
Overfitting occurs when the generated model is overly complex, and too closely tries to capture the idiosyncrasies in the data under study, instead of capturing the overall pattern. By doing so, the model will fail to accurately predict future (previously unseen) observations.

Inductive Learning

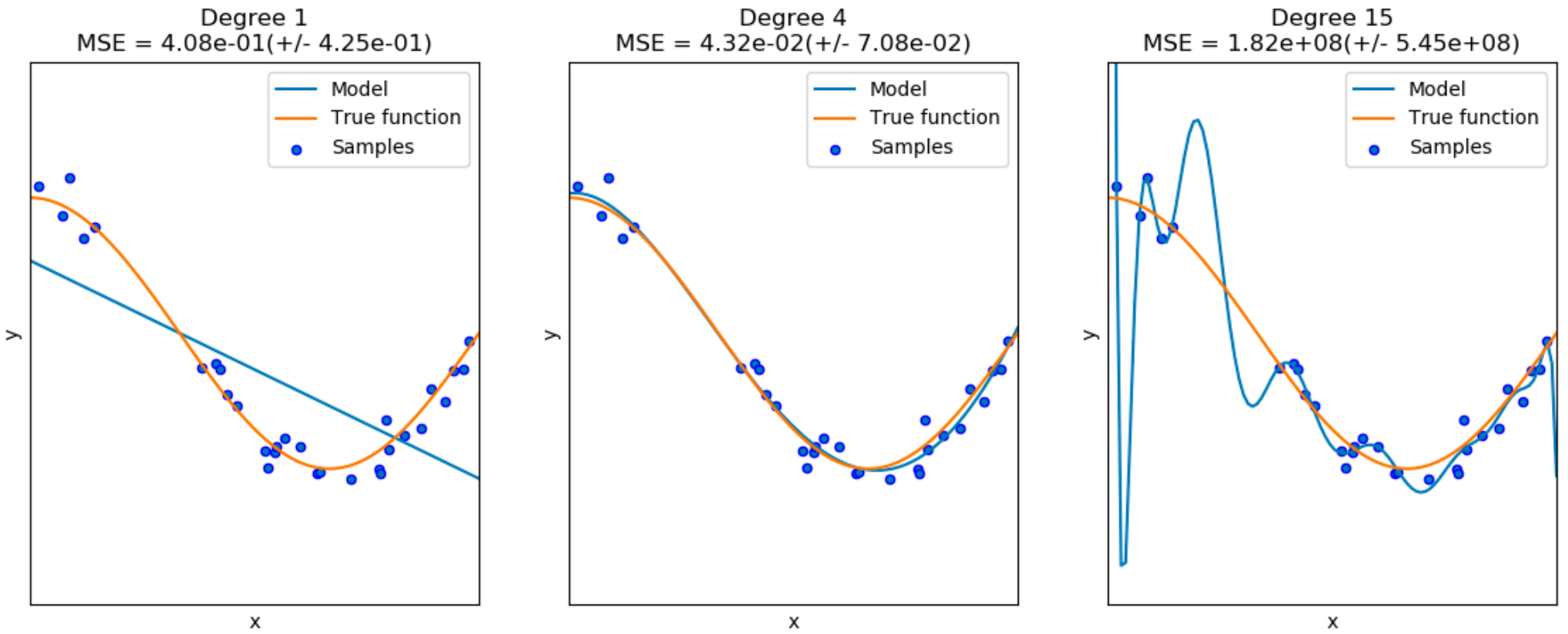
- Construct/adjust h to agree with f on training data

- E.g., curve fitting:



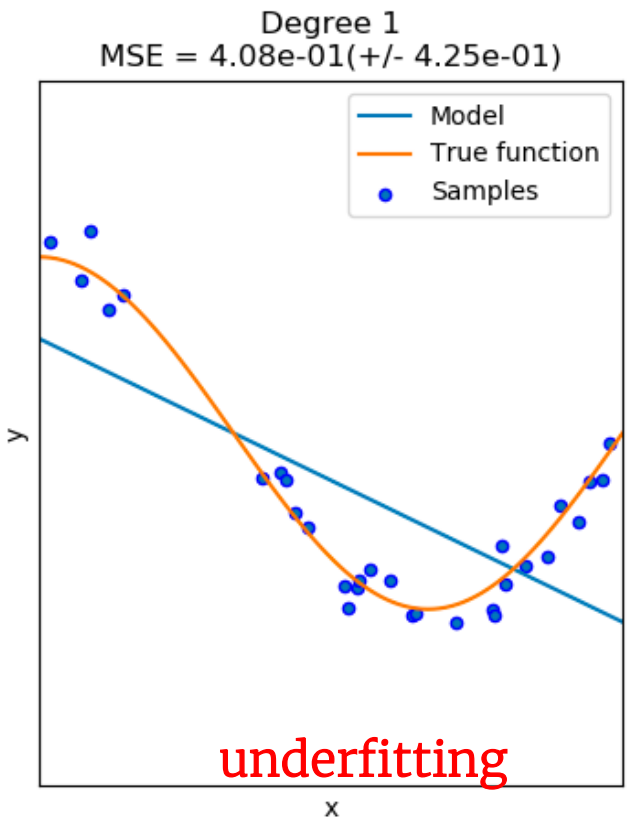
Ockham's razor: prefer the simplest hypothesis consistent with data (e.g., green)

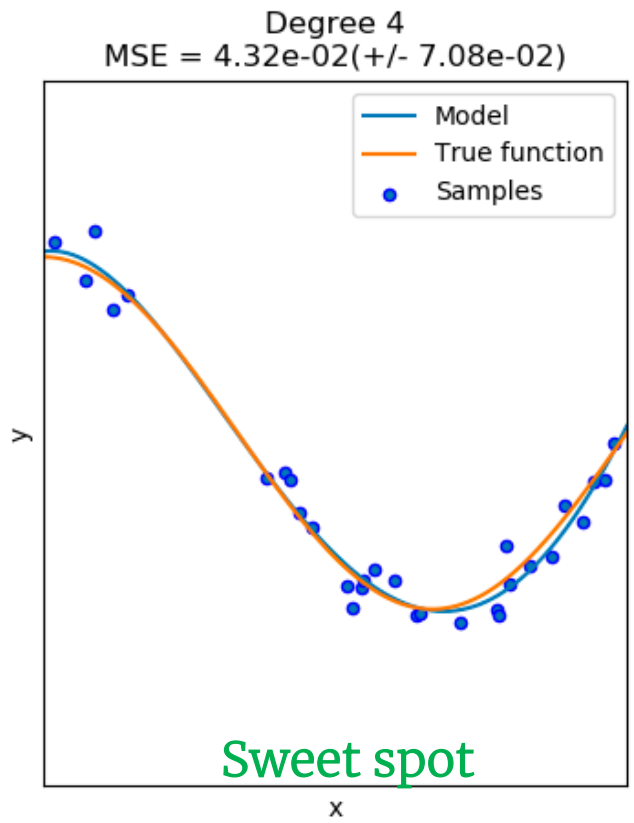
Example from sklearn

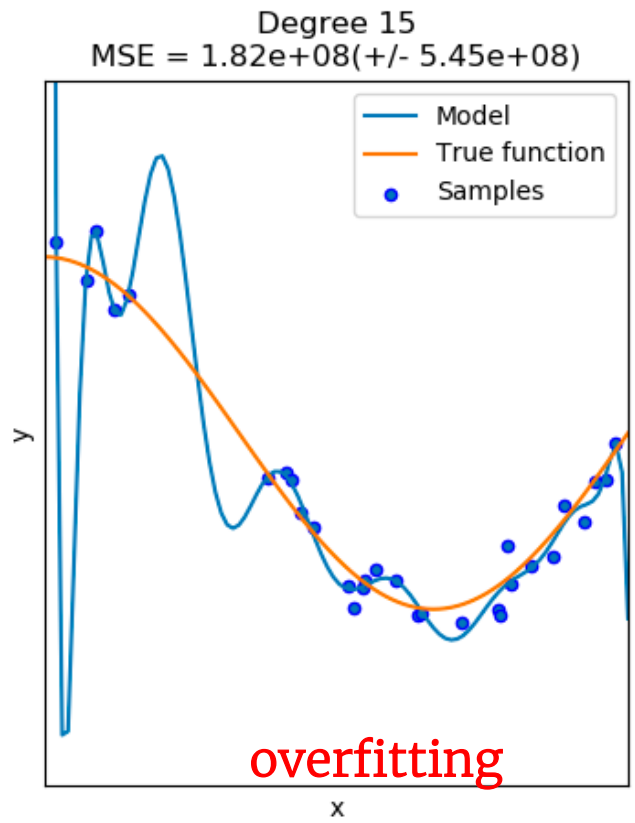


http://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

Example from sklearn





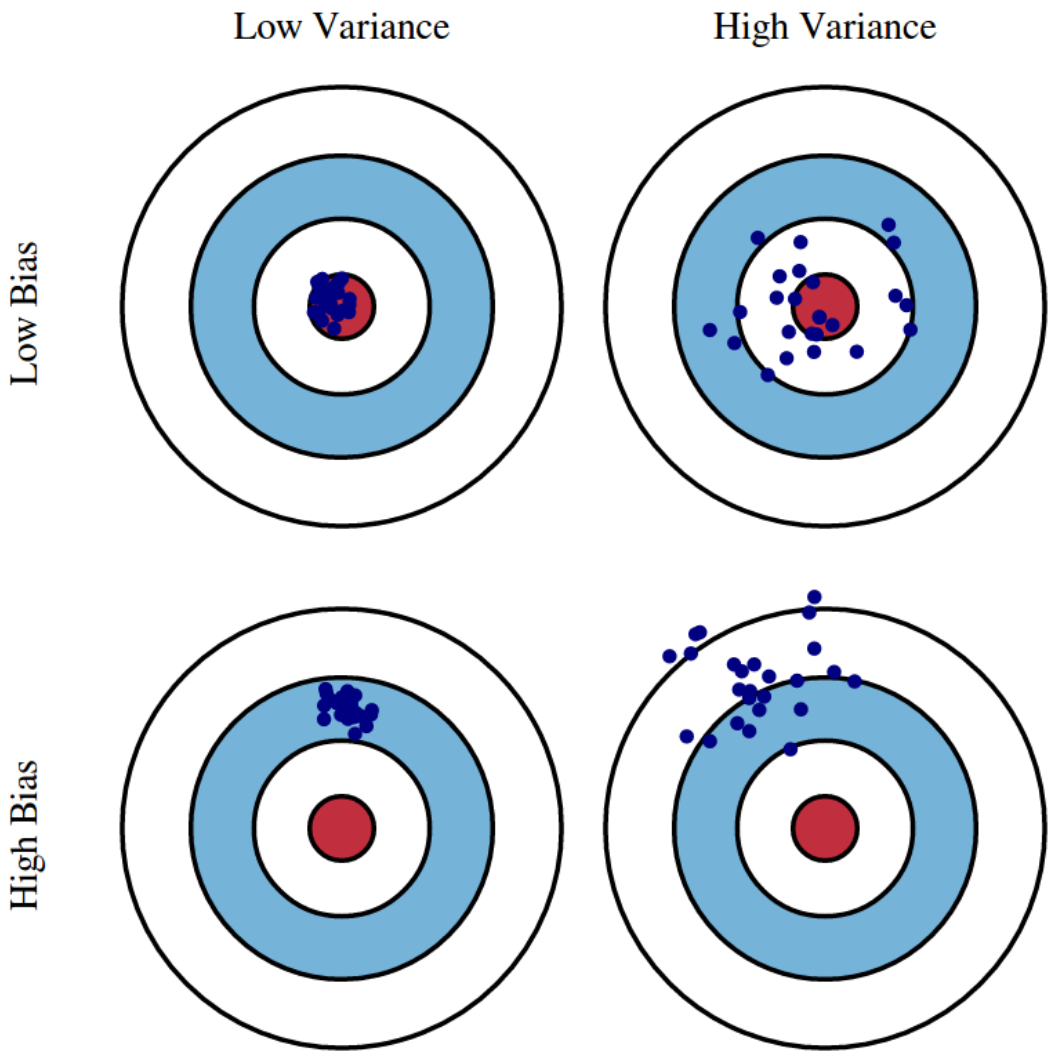


http://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

Bias-Variance Trade-off

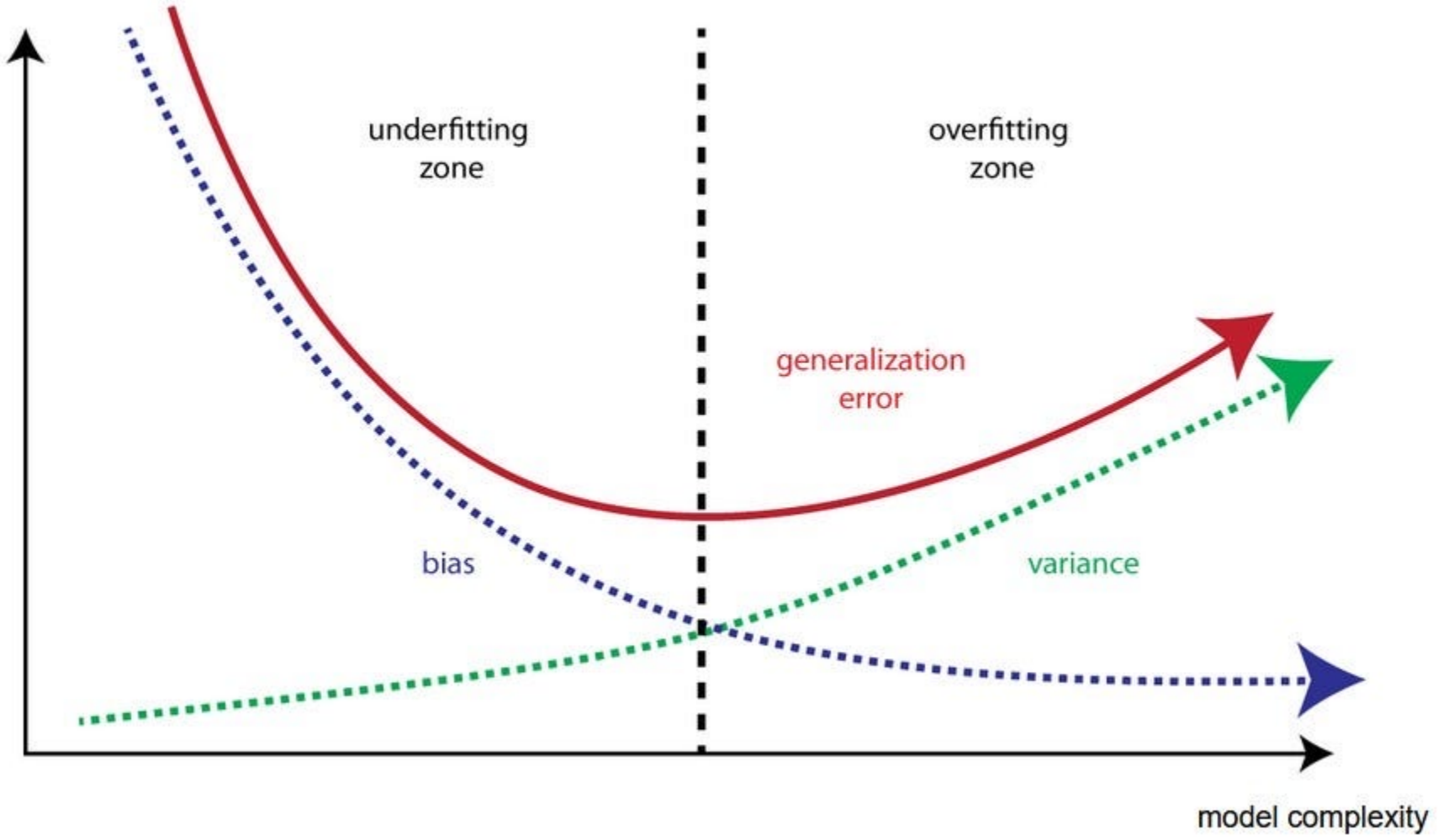
Variance: Captures how much your classifier changes if you train on a different training set. How "over-specialized" is your classifier to a particular training set (overfitting)? If we have the best possible model for our training data, how far off are we from the average classifier?

Bias: What is the inherent error that you obtain from your classifier even with infinite training data? This is due to your classifier being "biased" to a particular kind of solution (e.g. linear classifier). In other words, bias is inherent to your model.



http://scott.fortmann-roe.com/docs/BiasVariance.html

Bias-Variance Trade-off

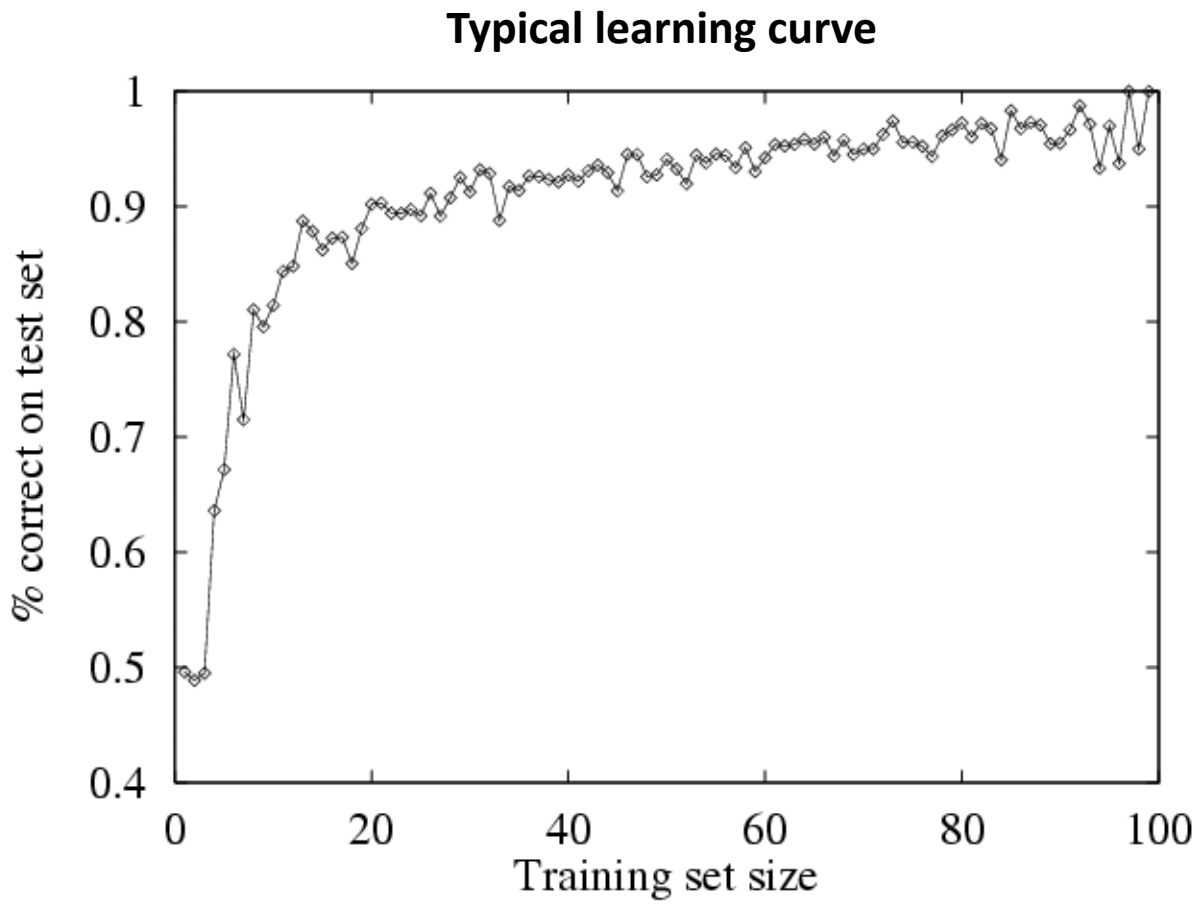


Avoiding **Overfitting**

1. Divide the data that you have into a distinct training set and test set.
2. Use only the training set to train your model.
3. Verify performance using the test set.
   - Measure error rate, accuracy, F1 score (more on this later)

- Drawback: the data withheld for the test set is not used for training
  - 50-50 split of data means we didn't train on half the data
  - 90-10 split means we might not get a good idea of the accuracy

More Data Is Typically Better



Binary vs Multiclass (Multinomial) Classification

- **Binary classification**: labels belong to only two classes (e.g., 0 and 1)

- **Multiclass classification**: labels belong to three or more classes (e.g., “sparrows”, “pigeons”, “crows” if classifying birds)

Some classification algorithms are naturally designed for more than two classes, while others are inherently binary algorithms. Binary classifiers can be turned into multinomial classifiers by a variety of strategies.

Classification Algorithms

- K-nearest neighbors
- Support vector machines
- Decision trees
- Neural networks
- Logistic regression
- ...

Classification Algorithms

- K-nearest neighbors
- Support vector machines
- Decision trees
- Neural networks
- Logistic regression
- …

We'll talk about these

K-Nearest Neighbors (KNN)

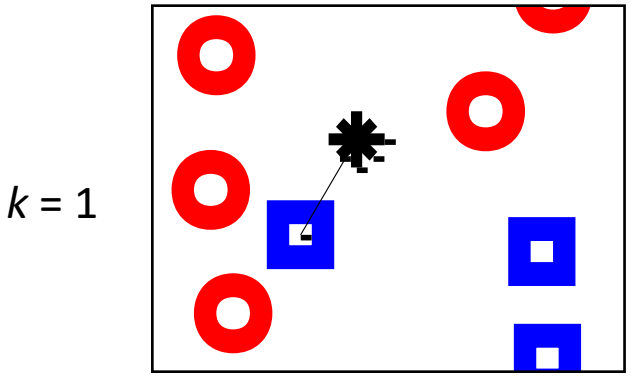
K-Nearest Neighbor algorithm

- Learning Algorithm:
  - Store all training examples
- Prediction Algorithm:
  - Classify the new example x_{new} by finding the training example (x_i, y_i) that is most similar
  - Return the corresponding label: $y_{new} = y_i$

Note: there are efficient data structures that can make the retrieval step efficient, though it remains computationally expensive for large datasets

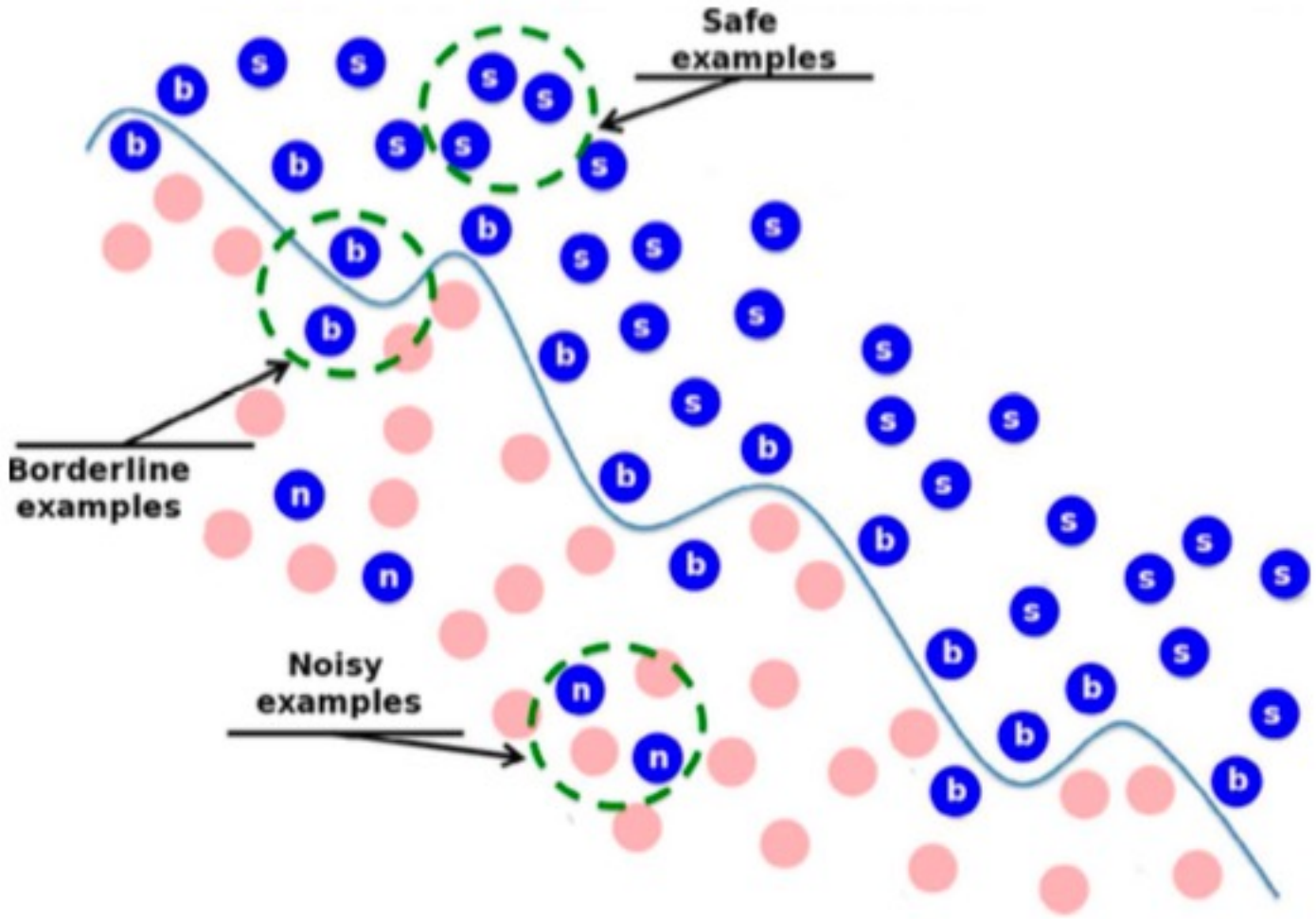
“Most similar”?

- 1–NN:
  - For a given query point q, assign the class of the nearest neighbour.



- This might not always work!
  - Unfortunately, most data contain some amount of random, meaningless information called *noise*.
  - Might be problematic near boundaries

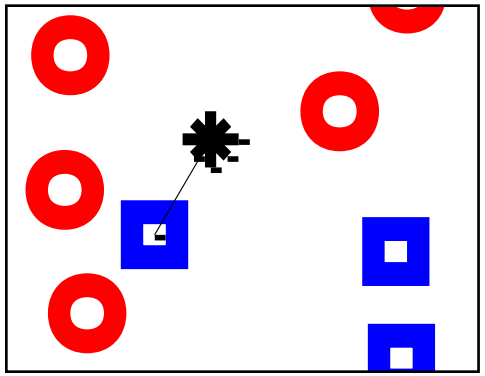
Noisy data



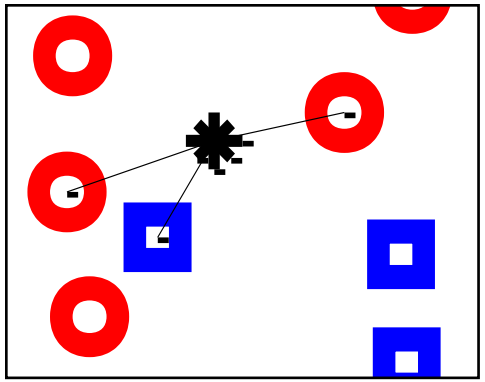
What do you do?

- 1-NN:
  - For a given query point q, assign the class of the nearest neighbour.
- K-NN
  - Compute the k nearest neighbours and assign the class by majority vote.
  - Increasing the value of k makes the algorithm more resilient to noise in the data... although it can also cause some unwanted side effects.

$k = 1$



$k = 3$



Calculating closest neighbors

Minkowski Distance (L^p norm):

$$L^p(x_j, x_q) = \left(\sum_i |x_{j,i} - x_{q,i}|^p \right)^{1/p}$$

Where i is the dimensionality of the data.

When $p = 2$, this equals Euclidean distance.

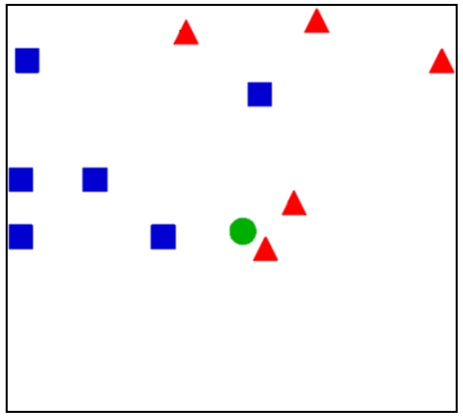
When $p = 1$, this equals Taxi-Cab or Manhattan distance.

Distance-Weighted k-NN

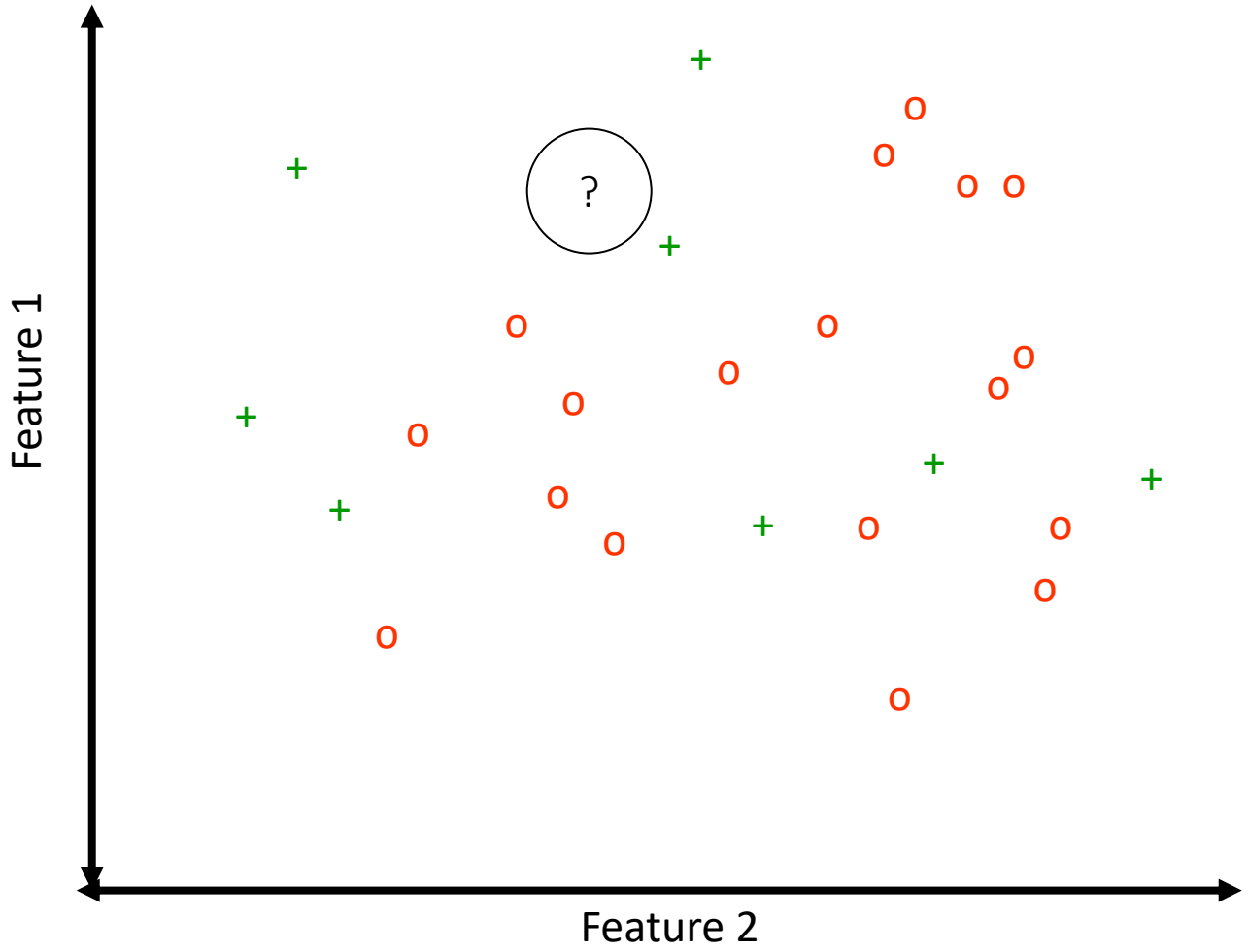
Insight: should points closer to the query have more influence than those far away?

Solution: Weigh the value that each of the k neighbors contributes to voting by a weight value corresponding to (or related to) the distance:

$$w_k = L^p(query, k)$$

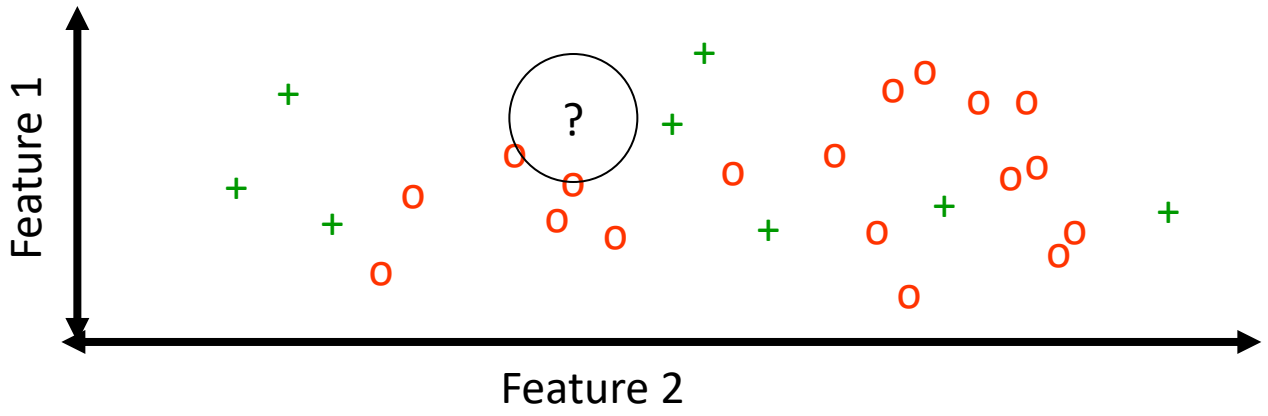


Not all features are not created equal!



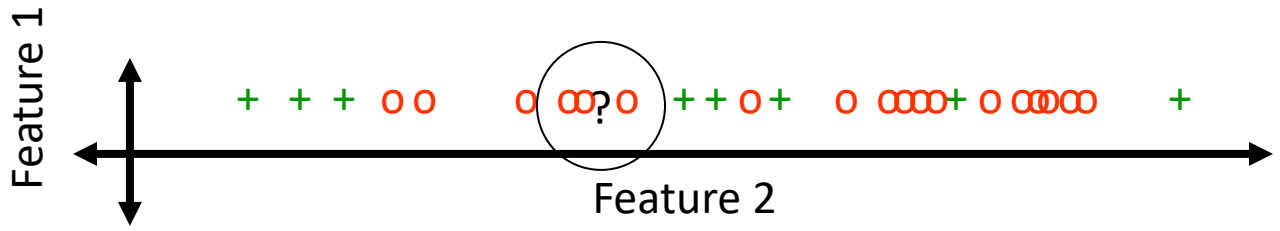
Not all features are not created equal!

Feature weighting!!



Why bother?

- Reduce noise, remove irrelevant information
- Ultimately, improve classification



Example: Training set

Example: Features

- Shape
- Size
- Color

Example: Query

Training set

Shape > Size >> Color!

Disadvantages of K-NN

- Determining distance function can be hard if feature values are scaled differently
  - Solution: normalize all feature values

- Generally, K-NN does not perform well when the dimensionality of the feature space is very high.

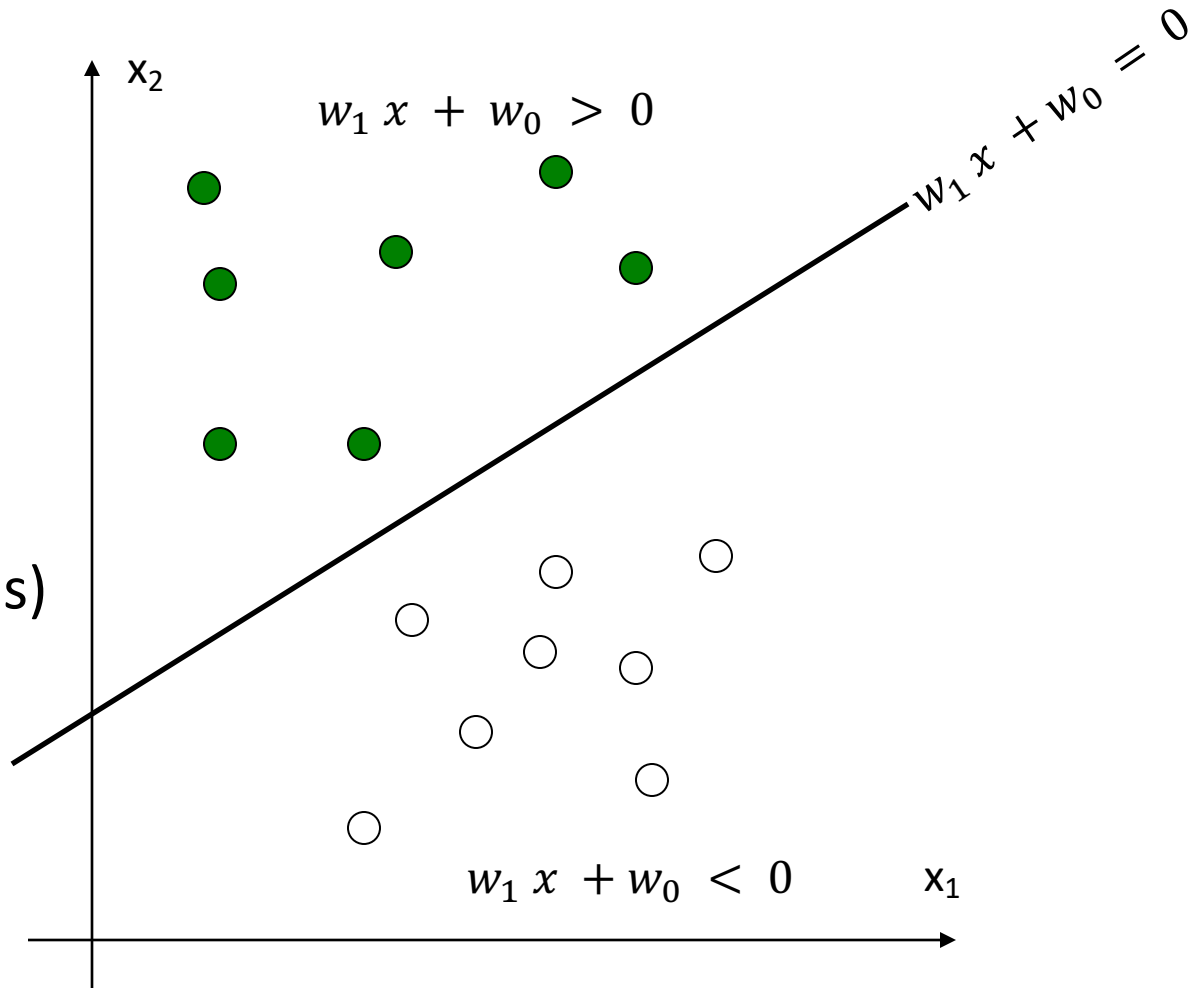
Support Vector Machines (SVM)

Linear Discriminant Function

f(x) is a linear function:

$$f(x) = w_1 x + w_0$$

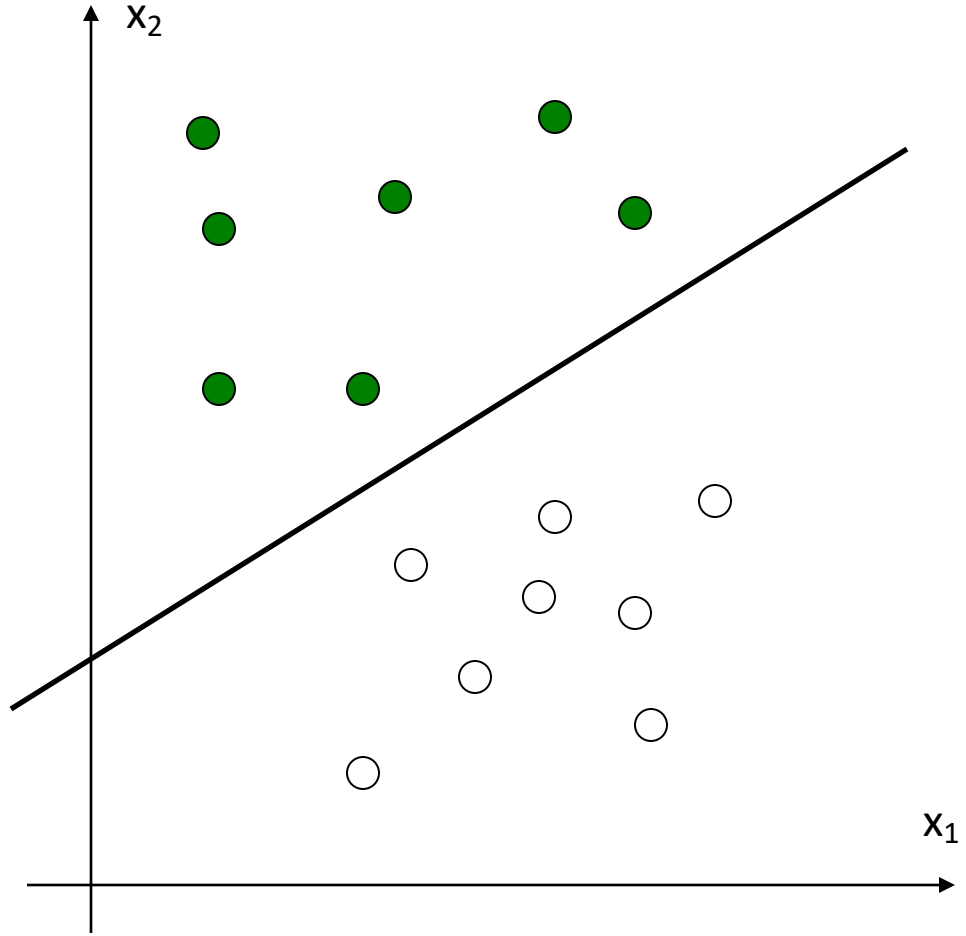
(a hyper-plane in >2 dimensions)



Linear Discriminant Function

How many linear discriminant functions will minimize the error rate?

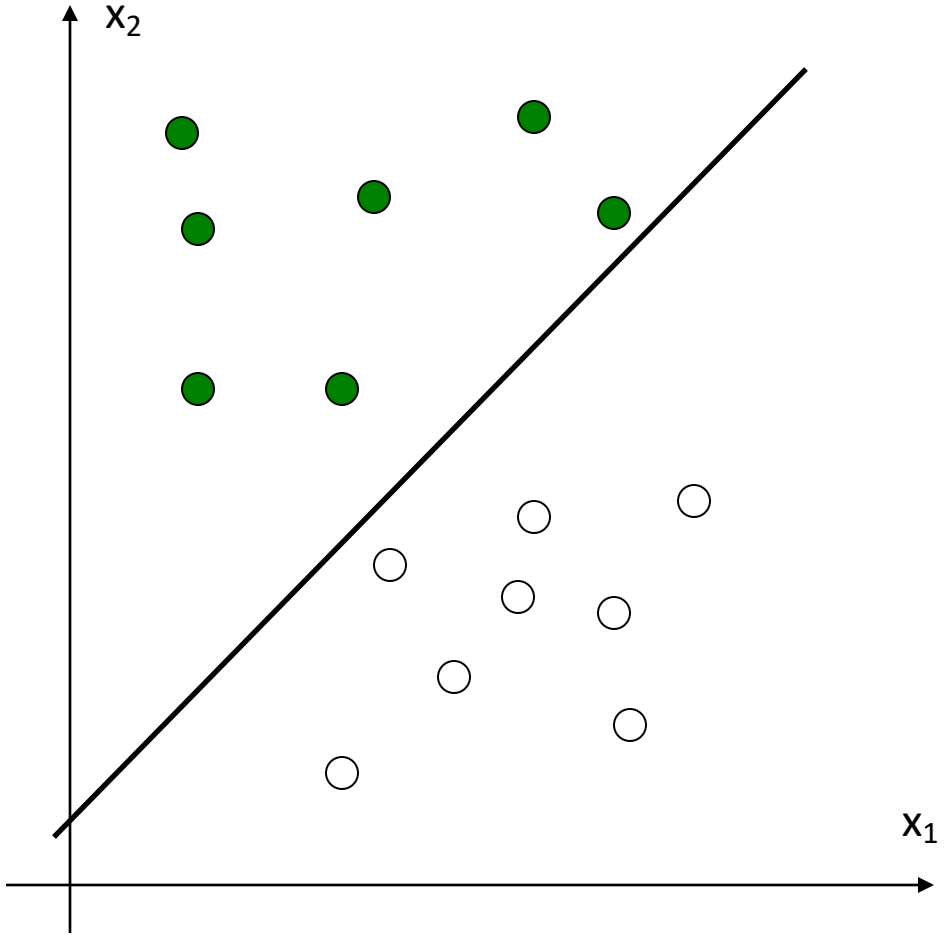
Infinite number of answers!



Linear Discriminant Function

How many linear discriminant functions will minimize the error rate?

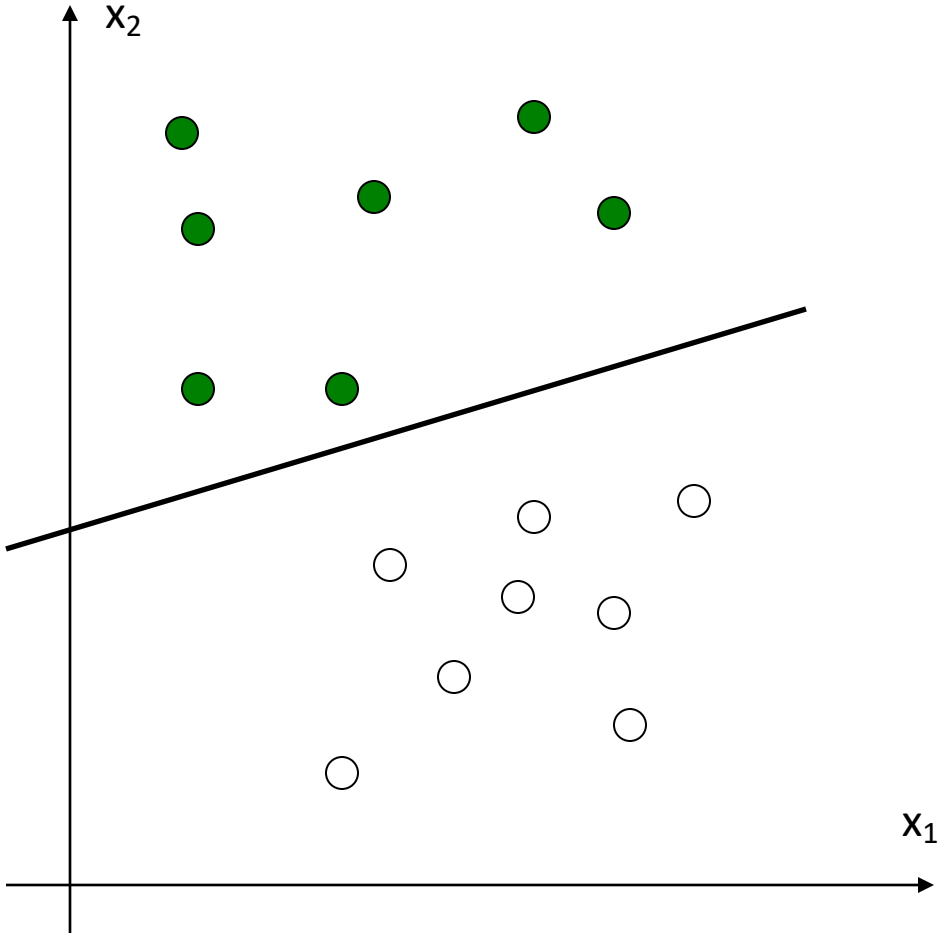
Infinite number of answers!



Linear Discriminant Function

How many linear discriminant functions will minimize the error rate?

Infinite number of answers!

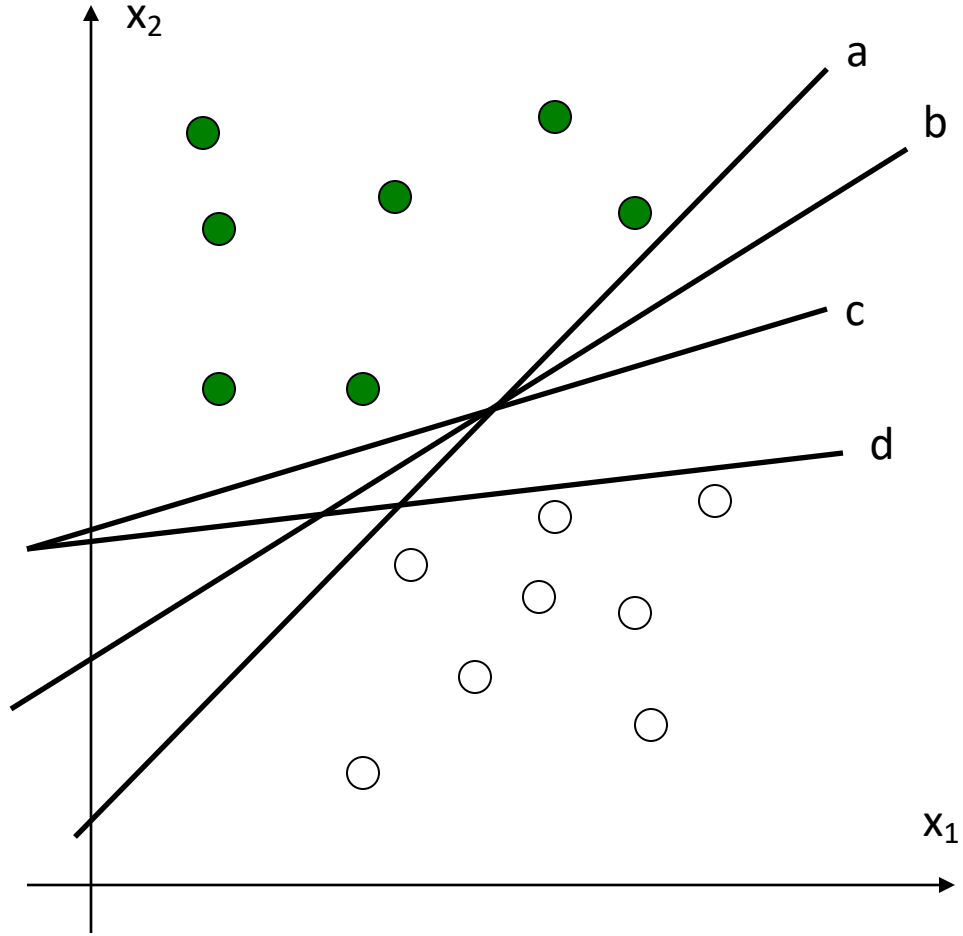


Linear Discriminant Function

How many linear discriminant functions will minimize the error rate?

Infinite number of answers!

Which one is best?



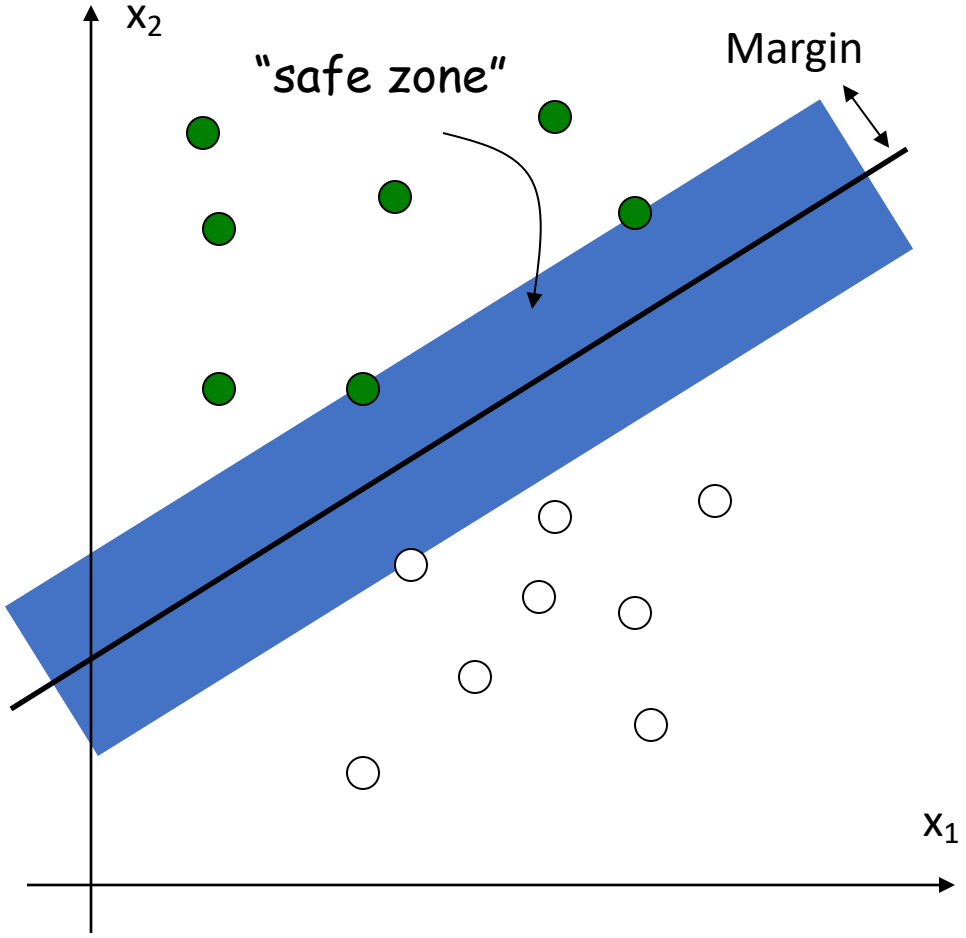
Maximum-Margin Linear Classifier

- The linear function with the maximum margin is the best!

- Margin is defined as the width that the boundary could be increased by before hitting a data point

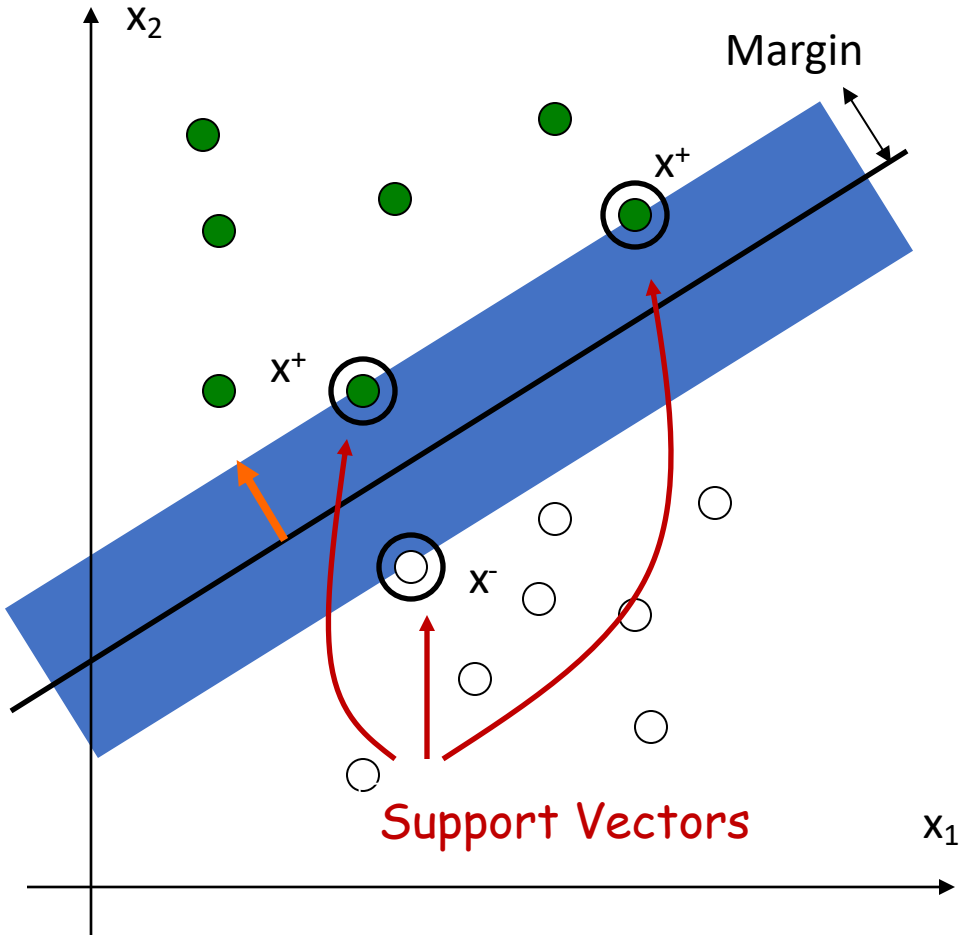
- Why large margin?

  - Of all the possible linear functions, this one is most robust to outliners and thus has the best generalization ability

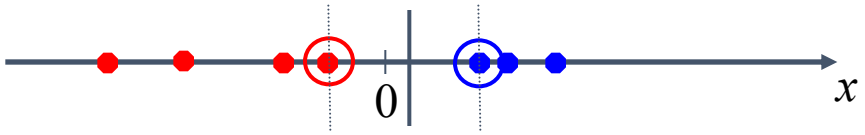


Maximum Margin Linear Classifier

Linear Support Vector Machine (SVM) classifier

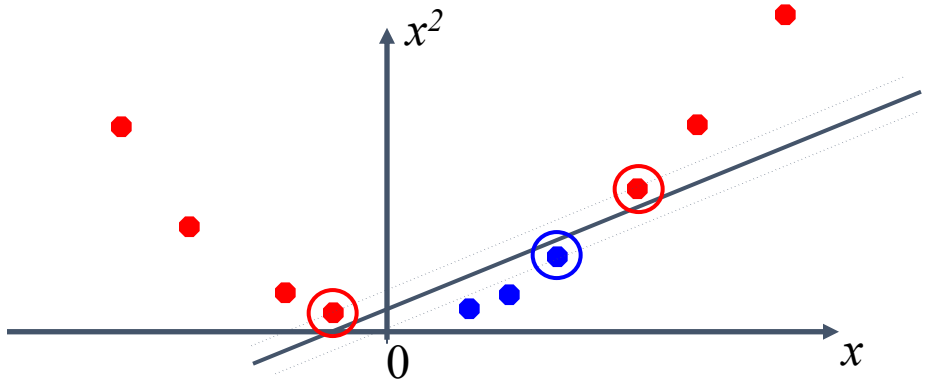


Datasets that are *linearly separable* with noise work out great:



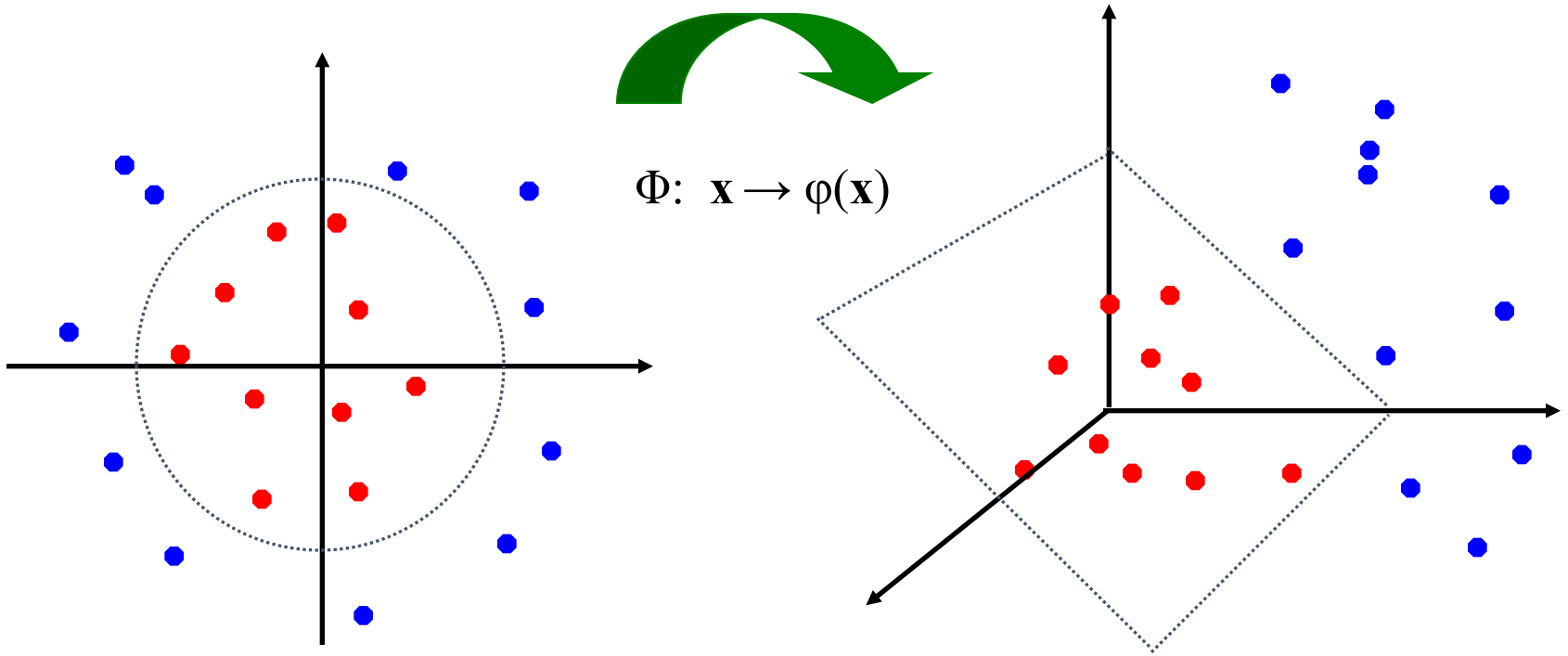
But what do we do if the dataset is not linearly separable? Can an SVM help us here?

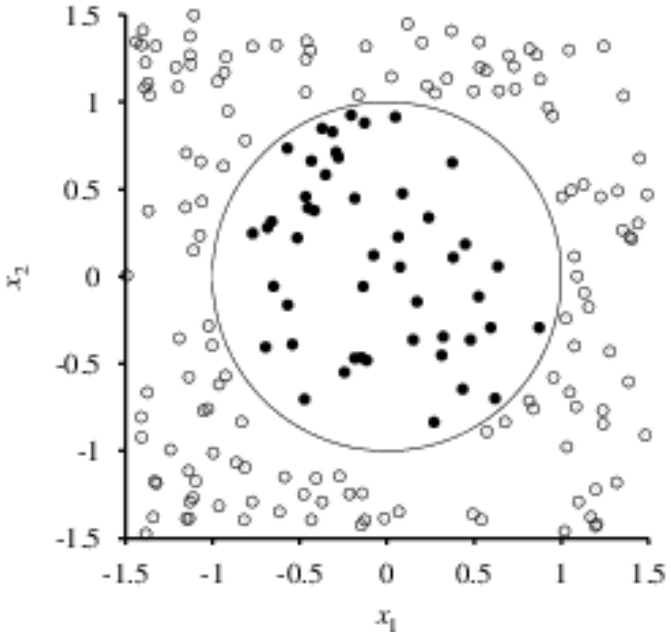
Key insight: mapping data to a higher-dimensional space can make it linearly separable:



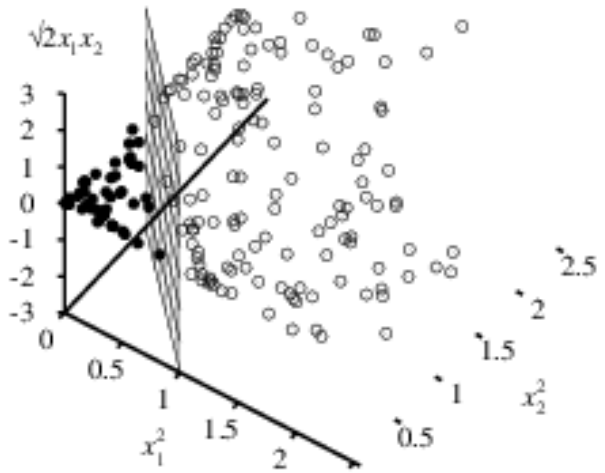
SVM Feature Space: The Kernel Trick

General idea: the original input space can be mapped to some higher-dimensional feature space where the training set is separable:



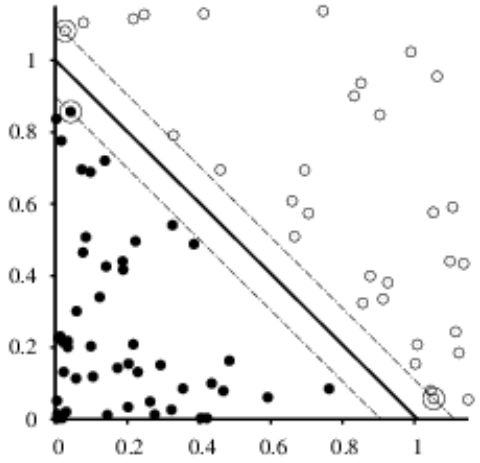


$$x_1^2 + x_2^2 \leq 1$$



$$(x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

Closeup of the decision boundary



Multiclass SVMs?

- SVMs are a binary classifier.
- How can we apply SVMs to multiclass problems?

Multiclass SVMs

One-against-all (one-against-rest) classification

- Train a single classifier per class, with the samples of that class as positive samples and all other samples as negatives
- N total classifiers for N classes
- This strategy requires the base classifiers to produce a real-valued confidence score for its decision of that class, and the class with the highest confidence is chosen.

One-against-one (pairwise) classification

- Train binary classifiers for a N-way multiclass problem. Each receives the samples of a pair of classes from the original training set and must learn to distinguish these two classes.
- Total of $N\ (N\ -\ 1)\ /\ 2$ binary classifiers for N classes (N choose 2)
- At prediction time, a voting scheme is applied: all $N\ (N\ -\ 1)\ /\ 2$ classifiers are applied to the new sample and the class that got the highest number of "+1" predictions gets predicted by the combined classifier.

Additional Resources

Watch “How Support Vector Machines work” by Brandon Rohrer at Meta https://brohrer.github.io/blog.html

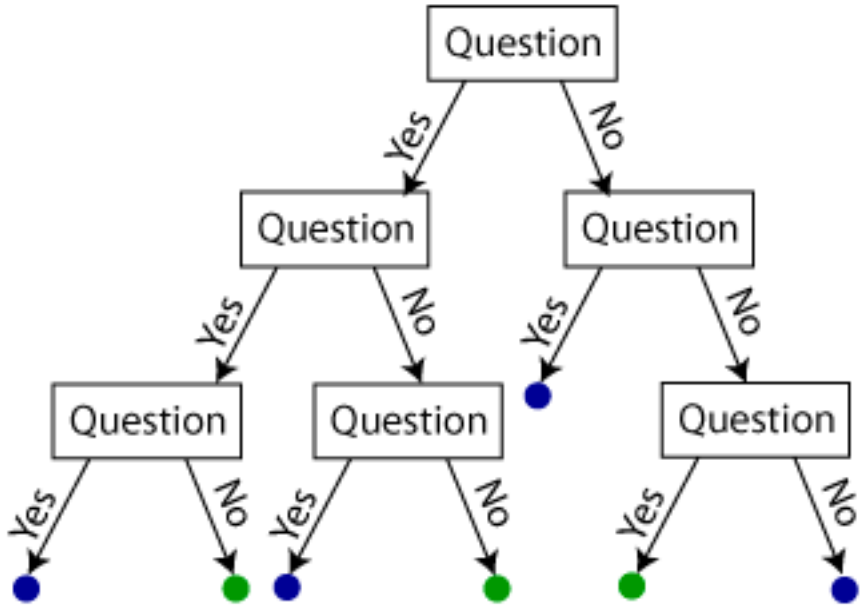
Sklearn:

http://scikit-learn.org/stable/modules/svm.html

http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Decision Trees

- ...similar to a game of 20 questions



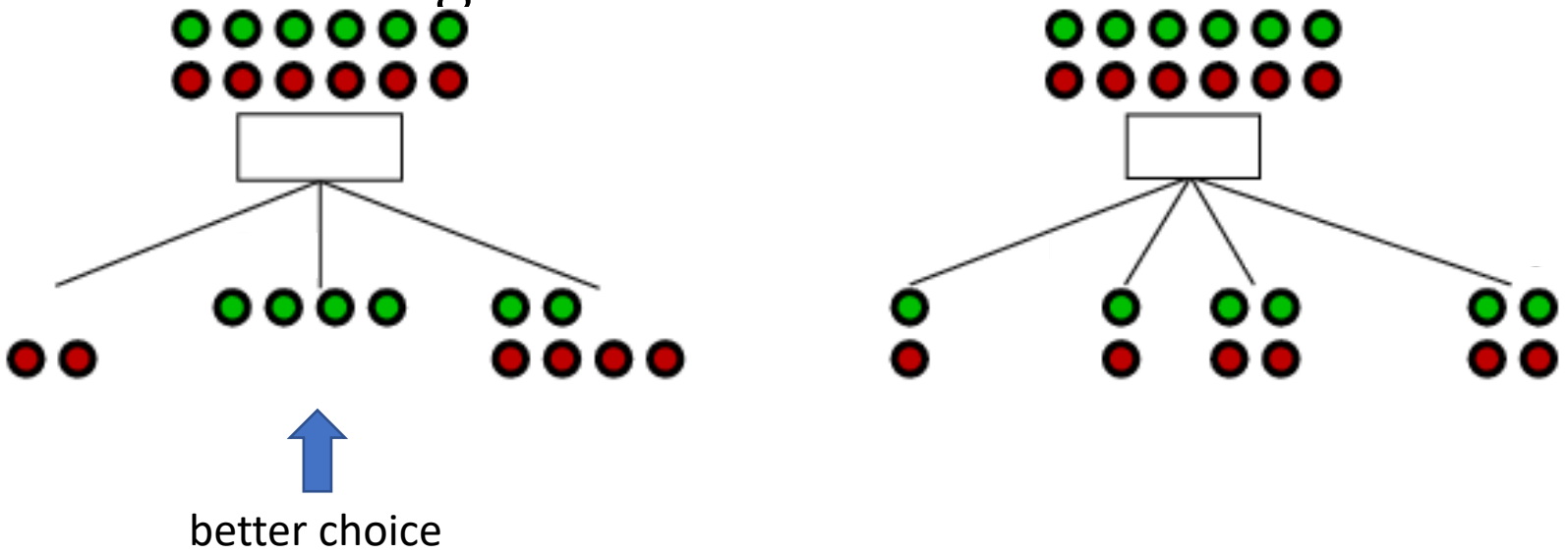
Example: road signs

- Determine feature/attribute vector used to represent signs
- Extract features for each sign
- Construct tree by splitting on features/attributes in order of importance

Example features: color, bounding box ratio, OCR output, color histogram values, etc.

Choosing an attribute

a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



Decision Tree algorithms choose splits based on the concepts of *entropy* and information gain, which we will not discuss but you can read about here: https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8 .

Over-fitting

- Over-fitting results in decision trees that are more complex than necessary
- Training error does not provide a good estimate of how well the tree will perform on previously unseen records (need a test set)
- Solutions:
  - Pruning
  - Early stopping

Disadvantage of Decision Trees

- Can be sensitive to data, small changes causing significant changes in the model
- Balancing between overfitting and generalization can be tricky

Random Forest

(ensemble learning method)

- Create many (hundreds, thousands) of *simpler* trees, to avoid overfitting any one tree. Here's how:
  - Given training set X, select a random subset of *features*, and fit a tree to these samples using only those features
  - Repeat until desired number of trees is reached
- Report the result obtained from majority vote by the collection of trees

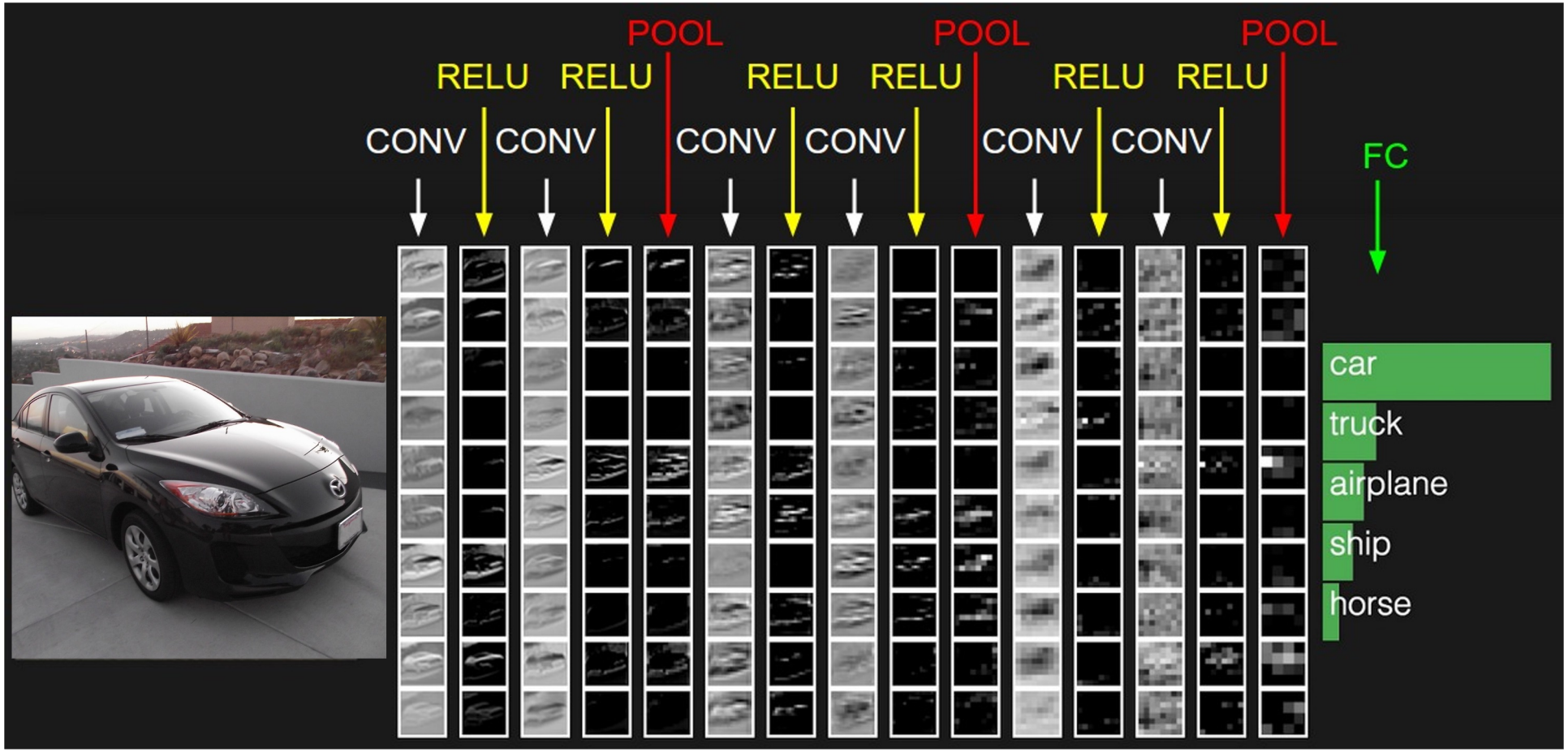
Boosting

Build many trees (hundreds, thousands), so that the weighted average over all trees is insensitive to fluctuations

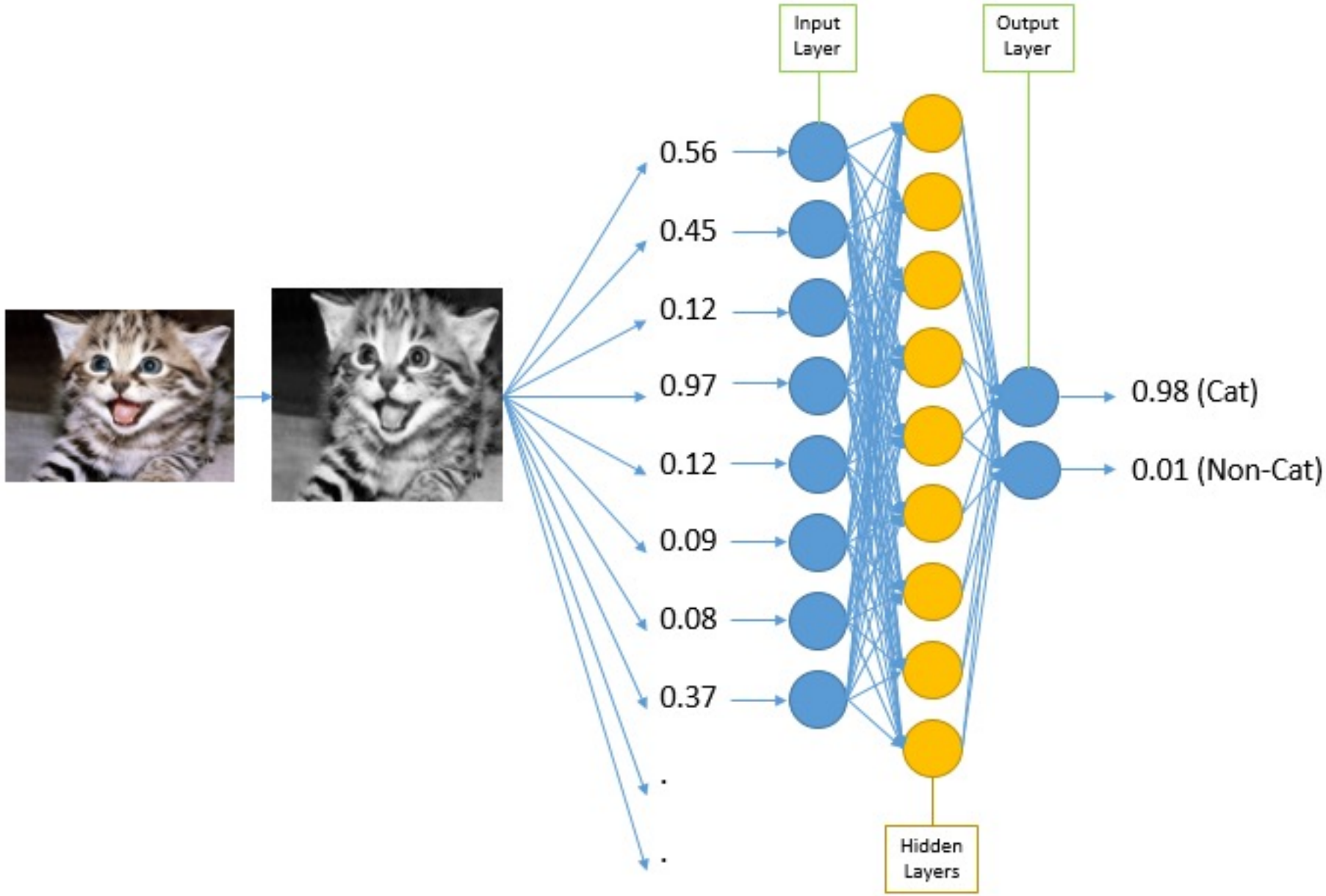
1. Create one tree
2. If there is misclassified data, create a second tree, giving more weight/importance to any misclassified examples
3. Score performance of each tree
4. Repeat to create more trees
5. Average over all trees, using the tree-scores as weights

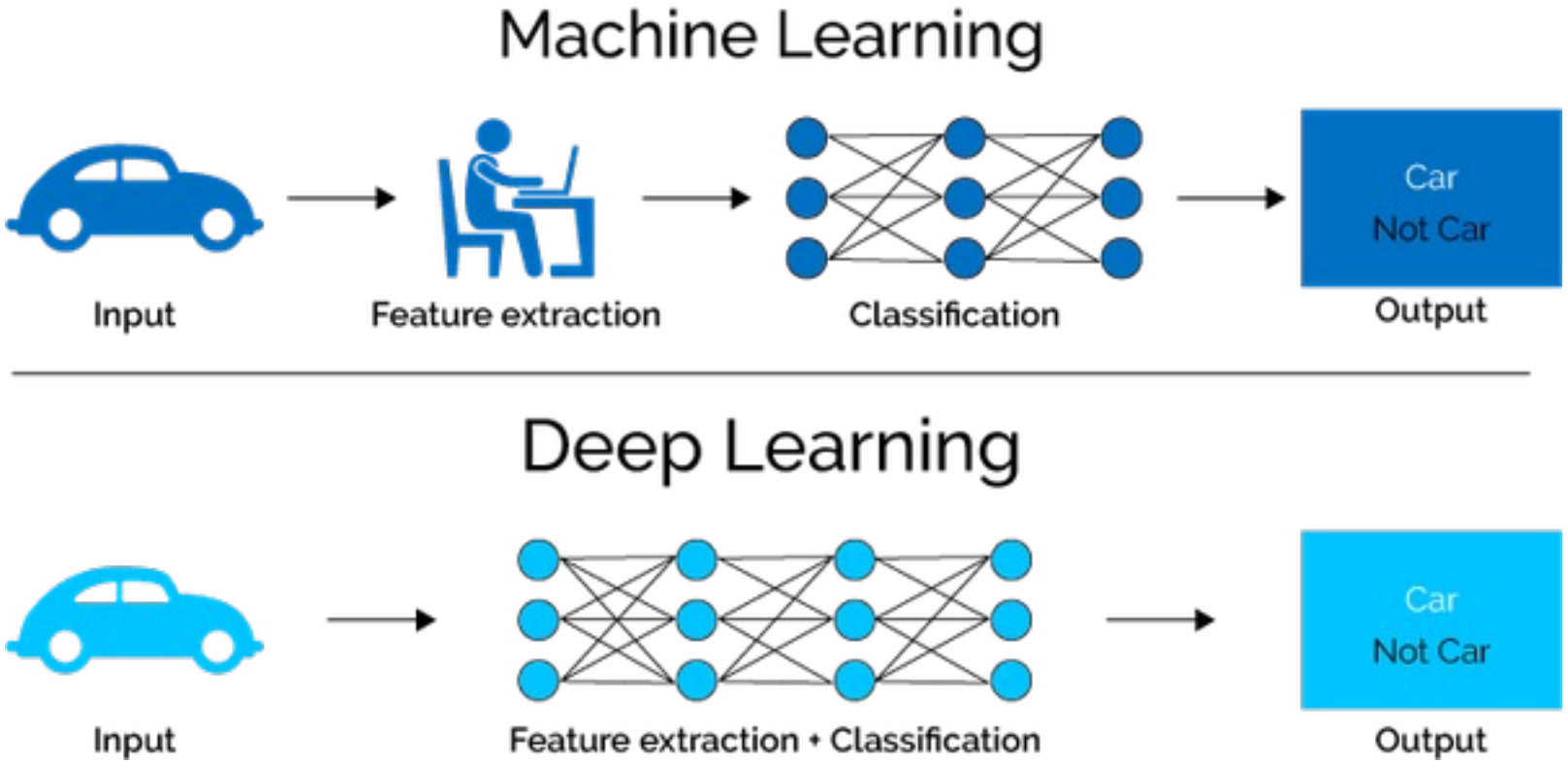
… and of course we've already talked about convolutional neural networks…

two more layers to go: POOL/FC



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n


Input Layer
Output Layer
0.56
0.45
0.12
0.97
0.12
0.09
0.08
0.37
.
.
0.98 (Cat)
0.01 (Non-Cat)
Hidden Layers



How complex is your problem?
How much data do you have?
How important is accuracy vs. interpretability?
How much time and resources are you willing to allocate?

https://blog.dataiku.com/when-and-when-not-to-use-deep-learning

Classical Machine Learning:

- Interpretability and explainability are paramount
- Smaller amounts of relatively simple data
- Straightforward feature engineering
- Limited computational power
- Limited time, need for faster prototyping and operationalization
- Need for varied algorithm choices
- Accuracy of test dataset results is acceptable

Deep Learning:

- Very high accuracy is a priority (and primes over straightforward interpretability and explainability)
- Large amounts of precisely labeled data
- Complex feature engineering
- Powerful compute resources available (GPU acceleration)
- Augmentation and other transformations of the initial dataset will be necessary

https://blog.dataiku.com/when-and-when-not-to-use-deep-learning