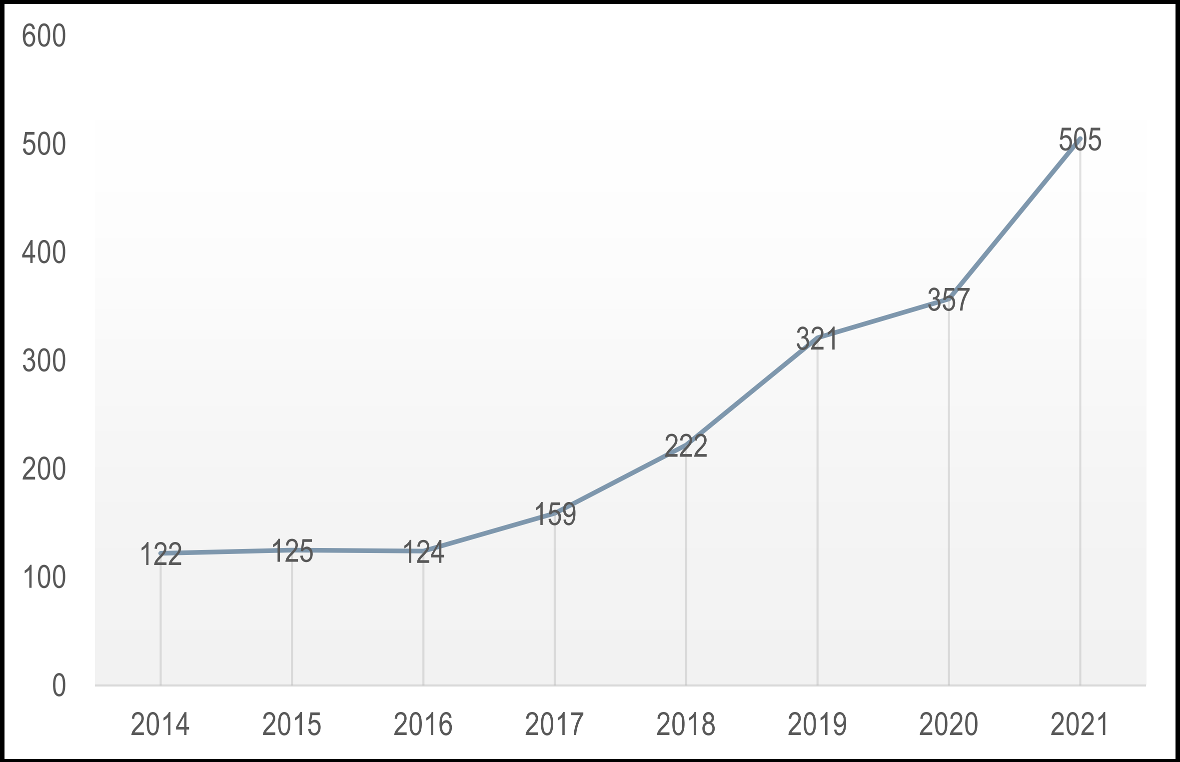
Decision Trees and SVMs

Is machine learning important for robotics?

- International Conference on Robotics and Automation (ICRA) (the mainstream conference in robotics)

Number of papers with the word "learning"



Year

Examples of Learning in Robotics

- Manipulation Example: **"**Learning Contact-Rich Manipulation Skills with Guided Policy Search**,"** Sergey Levine, Nolan Wagener, Pieter Abbeel, ICRA, 2015.
  - https://rll.berkeley.edu/icra2015gps/index.htm

- Perception Example: "DeepIM: Deep Iterative Matching for 6D Pose Estimation," Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang and Dieter Fox, ECCV, 2018
  - https://www.youtube.com/watch?v=61DM_WsigY4

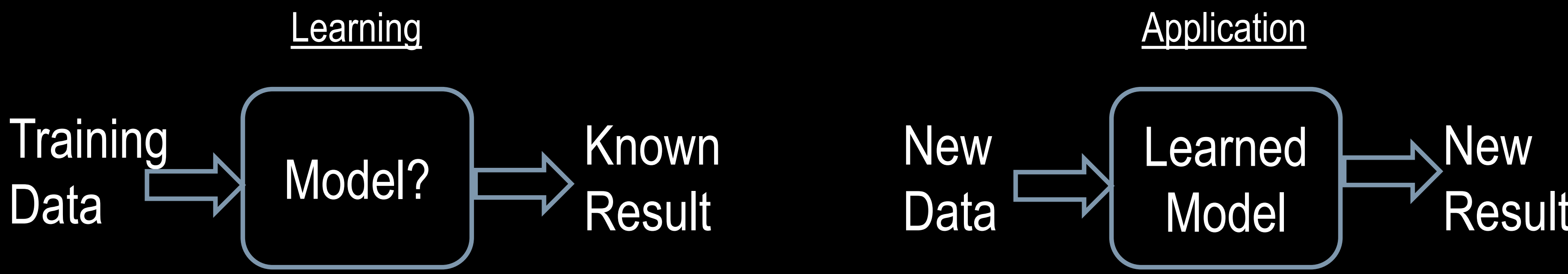
- Self-Driving Example: MIT 6.S094: Deep Learning for Self-Driving Cars

- A survey paper on deep learning in robotics from 2018 (already out-of-date):
  - https://arxiv.org/abs/1804.06557

- Many many more!

What is Machine Learning?

- Goal: Automatically compute models from data



Goal: Learn the model

- Three main types of learning:
  1. Supervised learning ← This lecture
  2. Unsupervised learning
  3. Reinforcement learning

Variants of Machine Learning Problems

- What is being learned?
  - Parameters, problem structure, hidden concepts,…
- What information do we learn from?
  - Labeled data, unlabeled data, rewards
- What is the goal of learning?
  - Prediction, diagnostics, summarization,…
- How do we learn?
  - Passive/active, online/offline
- Outputs
  - Binary, discrete, continuous

Outline

- Supervised learning
- Decision trees
- Support Vector Machines (SVM)

Supervised Learning

Supervised Learning

- Supervised learning: Given a set of data points with a label (discrete or continuous), create a model to describe them

- Classification: Label is a discrete variable
  - Usually binary but can extend to multiple classes

- Regression: Label is continuous

Training Data

$$\left.\begin{array}{cc} x_{11}x_{12}x_{13} & y_1 \\ x_{21}x_{22}x_{23} & y_2 \\ \ldots & \\ x_{m1}x_{m2}x_{m3} & y_m \end{array}\right\}$$

Training data
= Data set
= Training set
= Observations

Input   Label

- Each y_i was generated by a function $f(x_i) = y_i$, and more generally $f(x) = y$.
- Machine 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)

Learning is just optimization

- Recall the optimization problem from the beginning of class:

$$
\begin{array}{ll}
\text{minimize} & f_0(x) \\
\text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\
 & h_i(x) = 0, \quad i = 1, \ldots, p
\end{array}
$$

- $x \in \mathbf{R}^n$ is the optimization variable
- $f_0 : \mathbf{R}^n \to \mathbf{R}$ is the objective or cost function
- $f_i : \mathbf{R}^n \to \mathbf{R}$, $i = 1, \ldots, m$, are the inequality constraint functions
- $h_i : \mathbf{R}^n \to \mathbf{R}$ are the equality constraint functions

- In supervised learning:
  - x often describes a function
    - E.g. the entries of x are the coefficients of some polynomial
  - The objective function is some desired property of your solution
    - E.g. sparsity, simplicity, etc.
  - The training data are the constraints

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.

- 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

Inductive Learning

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

- What does “describes” mean?
    - Proposal: h is the same as the function f that generated them

How can we pick the right h?

- 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?

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?
  - ~~Proposal: h is the same as the function f that generated them~~
  - h models the observations well, and is *likely to predict future observations well*

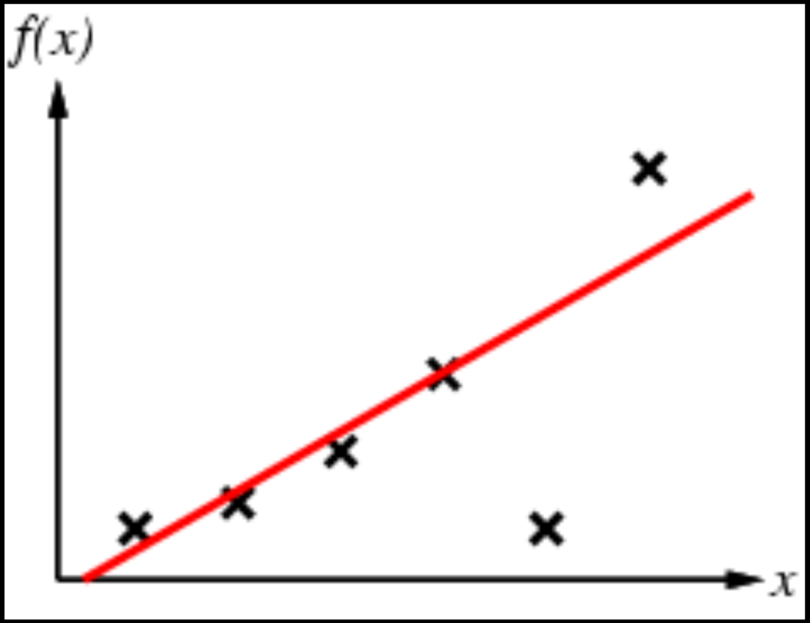
Inductive Learning

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

- E.g., curve fitting (AKA regression):

Inductive Learning

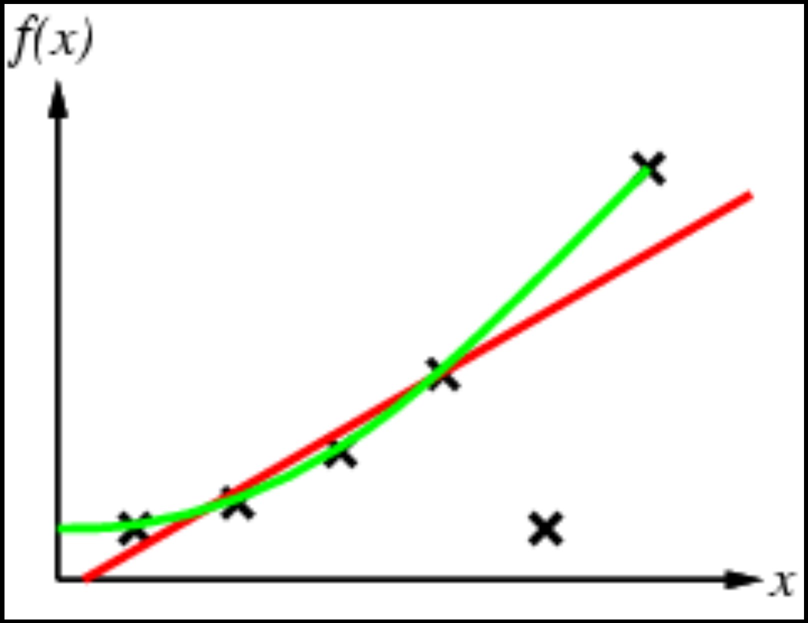
- Construct/adjust h to agree with f on training data
- E.g., curve fitting (AKA regression):



Inductive Learning

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

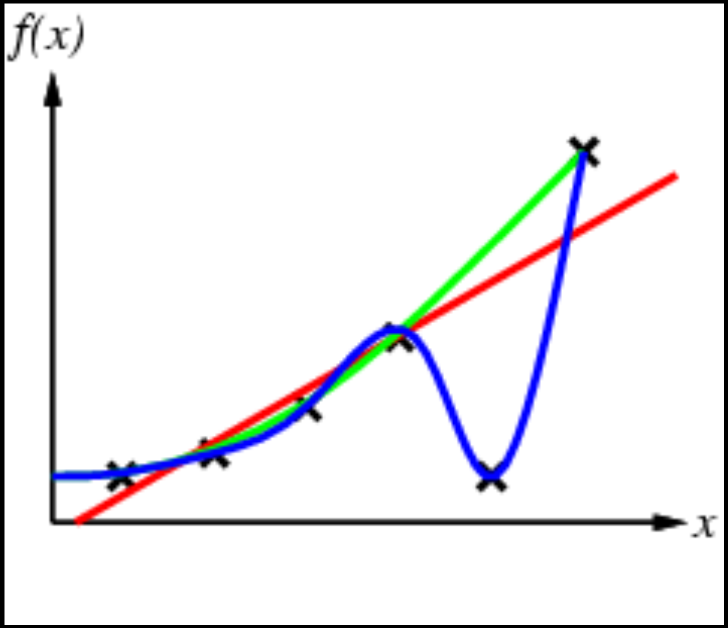
- E.g., curve fitting (AKA regression):



Inductive Learning

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

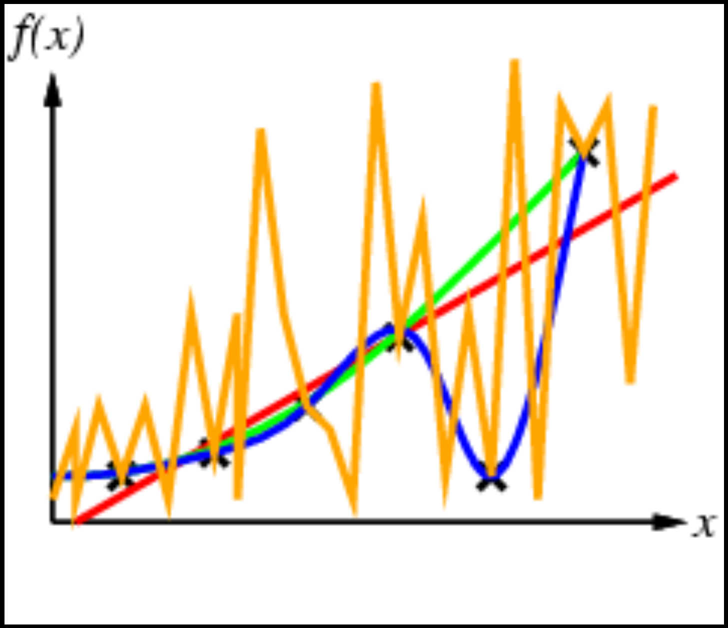
- E.g., curve fitting (AKA regression):



Inductive Learning

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

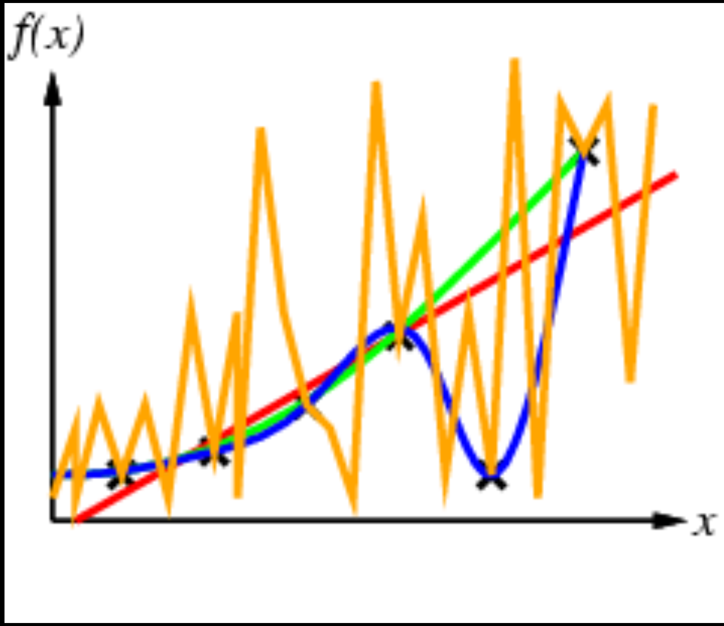
- E.g., curve fitting (AKA regression):



Inductive Learning

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

- E.g., curve fitting (AKA regression):



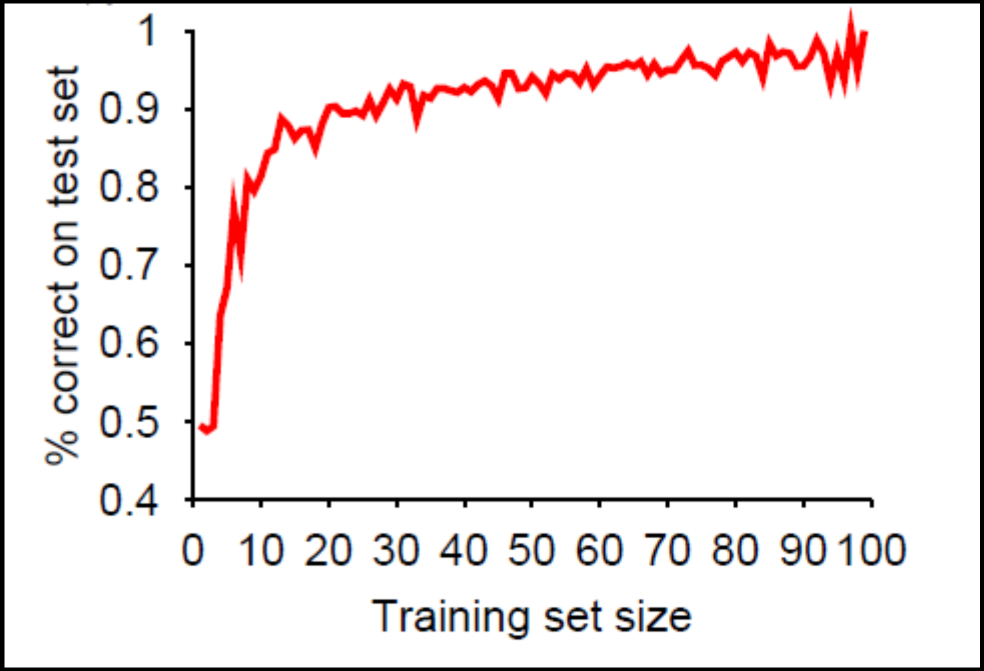
- Ockham's razor: prefer the simplest hypothesis consistent with data

Avoiding Overfitting the Model

1. Divide the data that you have into distinct training data and test data
2. Use only the training data to train your model
3. Verify performance using the test data
    - Measure error rate to evaluate method

- Drawback of this method: 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 Usually Produces Better Models



Decision Trees

Decision Trees

- …similar to a game of 20 questions
- Decision trees are simple, powerful and popular for classification and prediction
- Decision trees represent *rules*, which can be understood by humans and used in knowledge systems

Learning decision trees

Problem: decide whether to wait for a table at a restaurant, based on the following attributes:

1. Alternate: is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons: number of people in the restaurant (None, Some, Full)
6. Price: price range ($, $$, $$$)
7. Raining: is it raining outside?
8. Reservation: have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Attribute-based representations

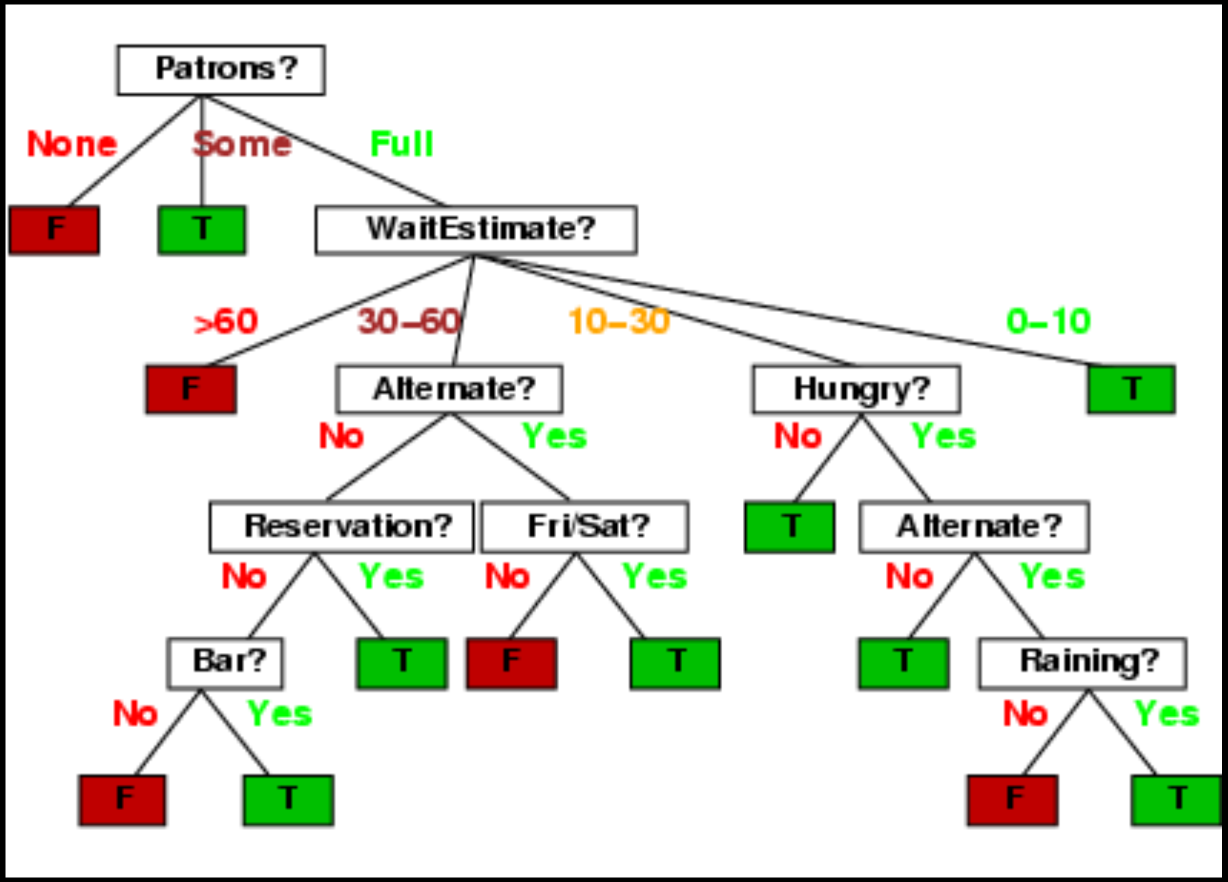
- Examples described by attribute values (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	Bar	Fri	Hun	Pat	$Price$	$Rain$	Res	$Type$	Est	$Wait$
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

- Classification of examples is positive (T) or negative (F)

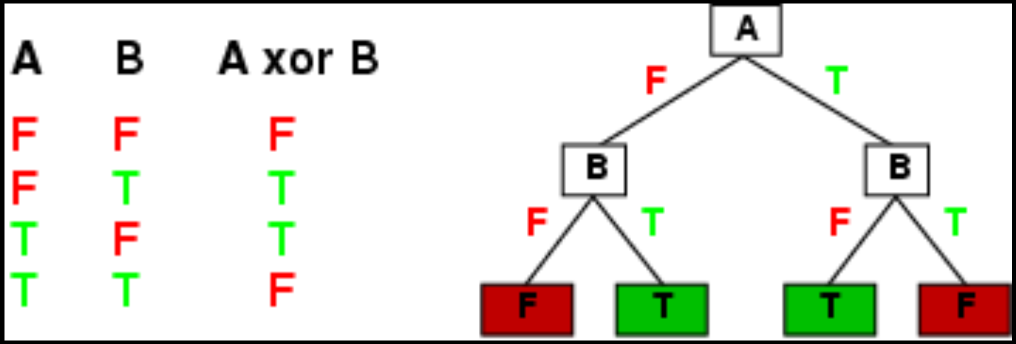
Decision trees

- One possible representation for hypotheses



Expressiveness

- Decision trees can express any function of the input attributes.
- E.g., for Boolean functions, truth table row → path to leaf:



- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless *f* is non-deterministic in *x*) but it probably won't generalize to new examples
- Prefer to find more compact decision trees
  - Remember Ockham’s razor

Hypothesis spaces

- How many distinct decision trees with n Boolean attributes?
  - = number of Boolean functions
  - = number of distinct truth tables with 2^n rows = 2^{2^n}
  - E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

- How many purely conjunctive hypotheses (e.g., Hungry ^ ¬Rain)??
  - Each attribute can be positive, negative, or not included
  - => 3^n distinct conjunctive hypotheses

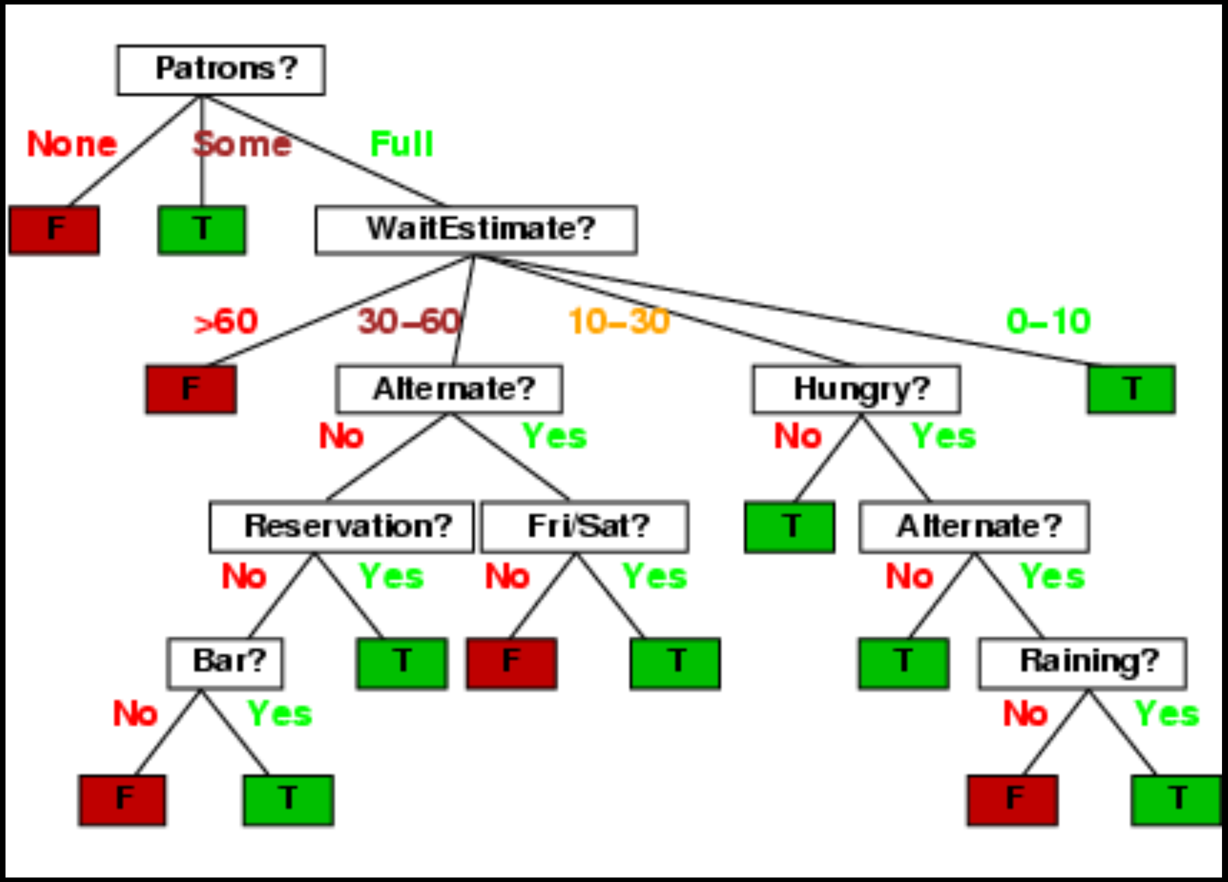
- More expressive hypothesis space
  - increases chance that target function can be expressed ☺
  - increases number of hypotheses consistent w/ training set
    - may get worse predictions on test data ☹

Decision tree learning

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

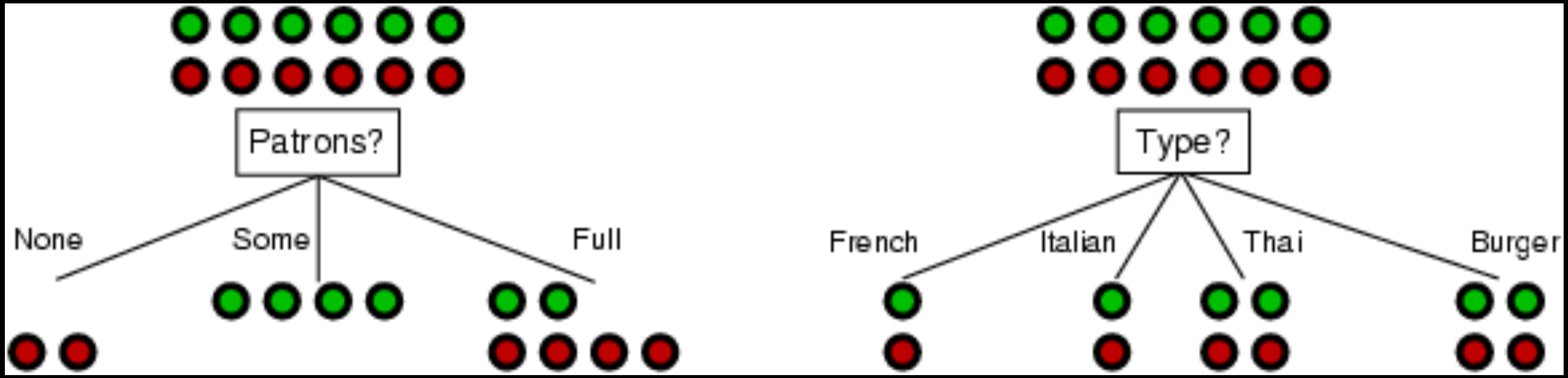
```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

One possible representation for hypotheses



Choosing an attribute

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



- Which is a better choice?
  - Patrons

Using information theory

- Implement `Choose-Attribute` in the DTL algorithm based on information content – measured by Entropy

- Entropy is the measure of uncertainty of a random variable
  - More uncertainty leads to higher entropy
  - More knowledge leads to lower entropy

Entropy

$$H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k) \log_2 P(v_k)$$

- For a training set containing *p* positive examples and *n* negative examples:

$$H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2 \frac{p}{p+n} - \frac{n}{p+n}\log_2 \frac{n}{p+n}$$

Entropy Examples: Flipping a coin

- Fair coin flip:
  - $H(heads, tails) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1$
- Biased coin flip:
  - $H(heads, tails) = -(0.99 \log_2 0.99 + 0.01 \log_2 0.01) = 0.08$

Decision Trees and Information Gain

- Choose the attribute with the largest Information Gain (IG)
- Consider the attributes Patrons and Type:

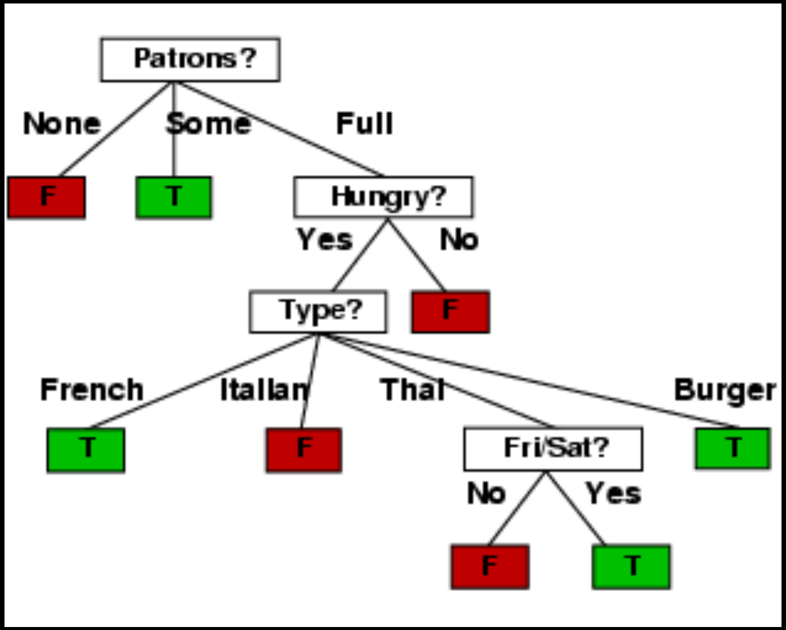
$$IG(Patrons) = 1 - [\frac{2}{12}H(0,1) + \frac{4}{12}H(1,0) + \frac{6}{12}H(\frac{2}{6},\frac{4}{6})] = .0541 \text{ bits}$$

$$IG(Type) = 1 - [\frac{2}{12}H(\frac{1}{2},\frac{1}{2}) + \frac{2}{12}H(\frac{1}{2},\frac{1}{2}) + \frac{4}{12}H(\frac{2}{4},\frac{2}{4}) + \frac{4}{12}H(\frac{2}{4},\frac{2}{4})] = 0 \text{ bits}$$

- Patrons has the highest IG of all attributes and so is chosen by the DTL algorithm as the root

Learned Restaurant Tree

- Decision tree learned from the 12 examples:



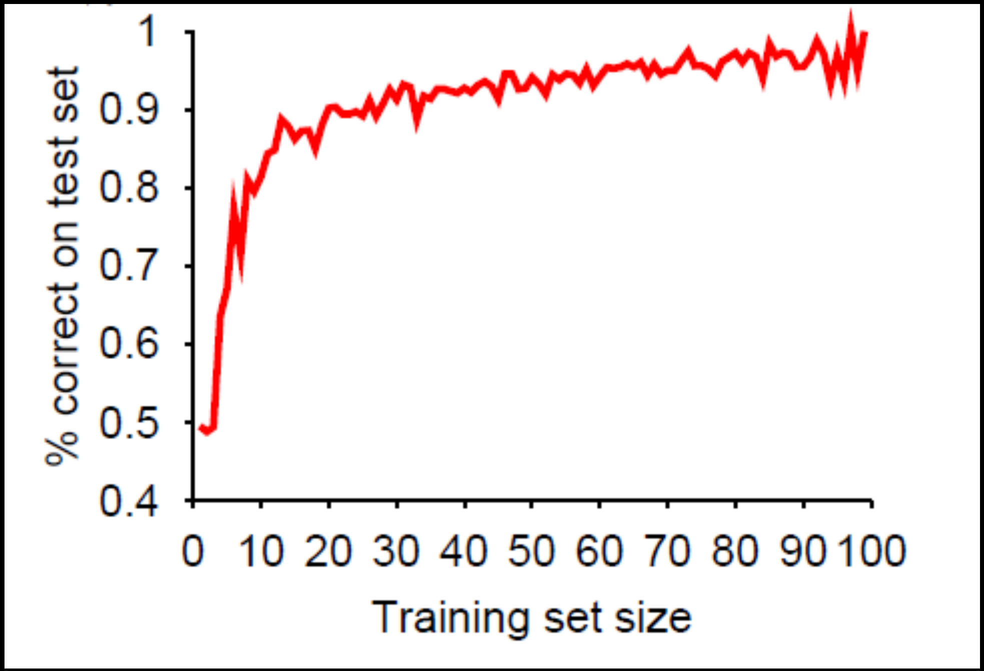
- Substantially simpler than the full tree
  - Raining and Reservation were not necessary to classify all the data.

Performance measurement

- How do we know that our function $h \approx f$?
  1. Use theorems of computational/statistical learning theory
  2. Try h on a new test set of examples

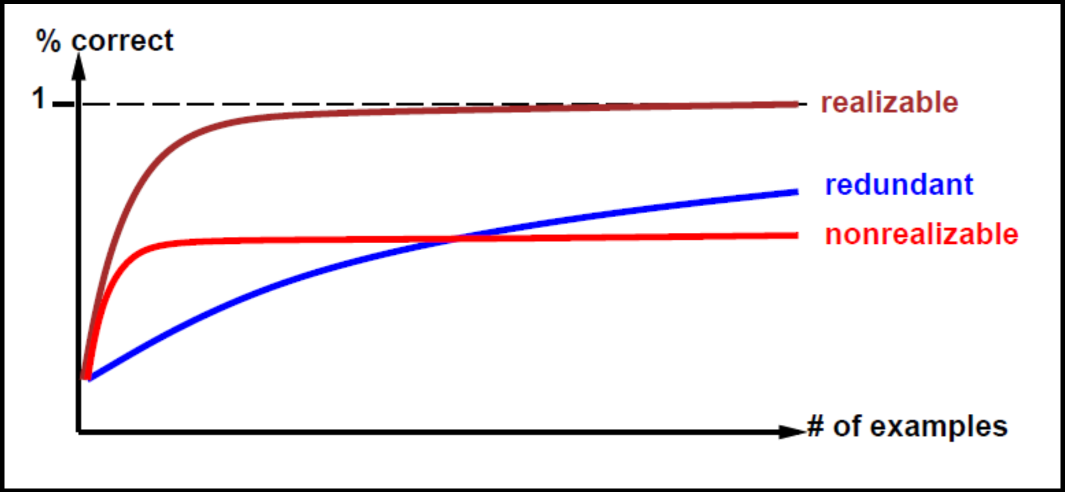
     (assumes *same* distribution over example space as training set)

Learning curve = % correct on test set as a function of training set size



Realizability and expressiveness

- Learning curve depends on realizability of your hypothesis class
  - realizable (can express target function) vs. non-realizable
- Non-realizability can be due to missing attributes or restricted hypothesis class (e.g., thresholded linear function)
- Redundant expressiveness (e.g., lots of irrelevant attributes) may require many more examples



Break

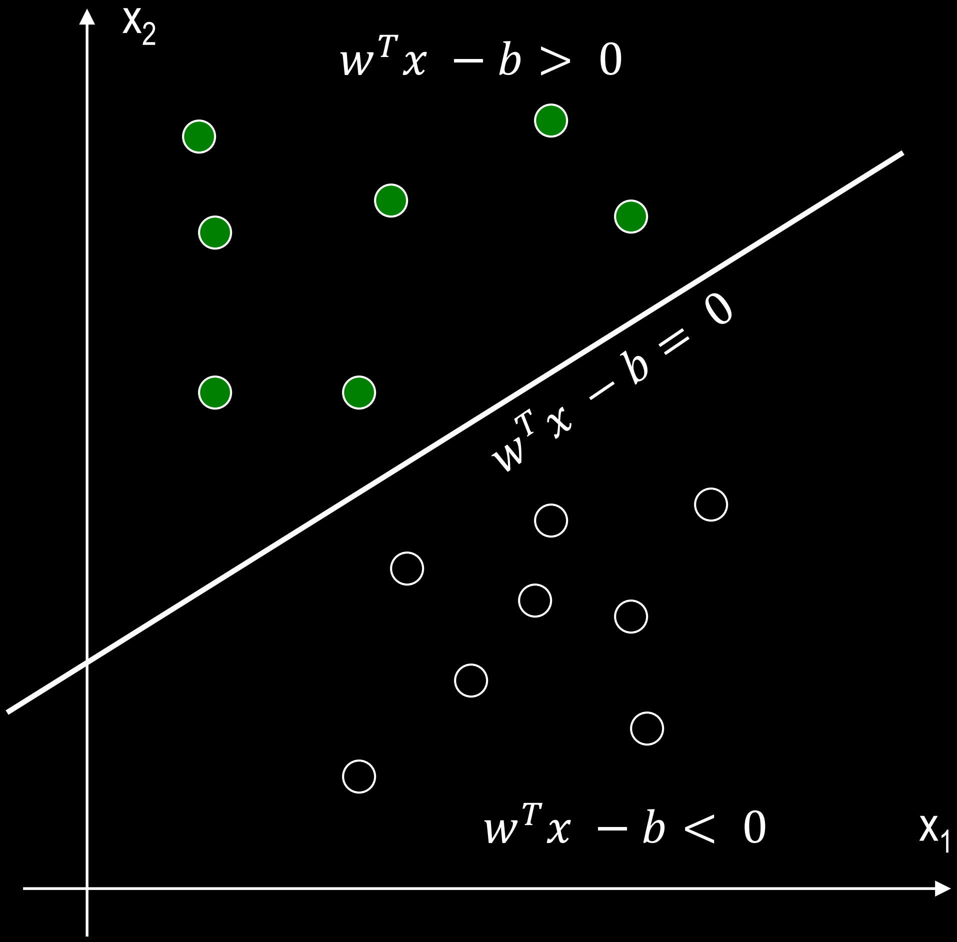
Support Vector Machines

Linear Discriminant Function

- A discriminant function f(x) is a function that separates two sets of data according to their labels
- Assume f(x) is a linear function:

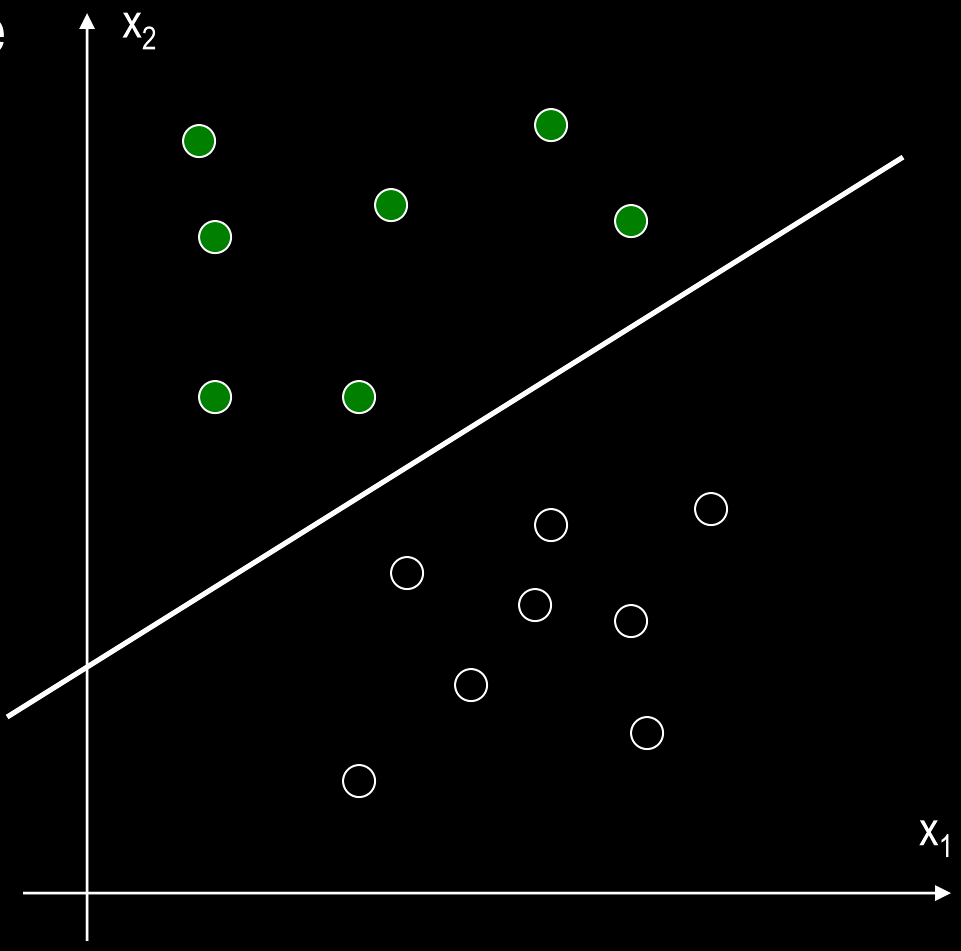
$$f(x) = w^T x - b$$

(a hyper-plane)



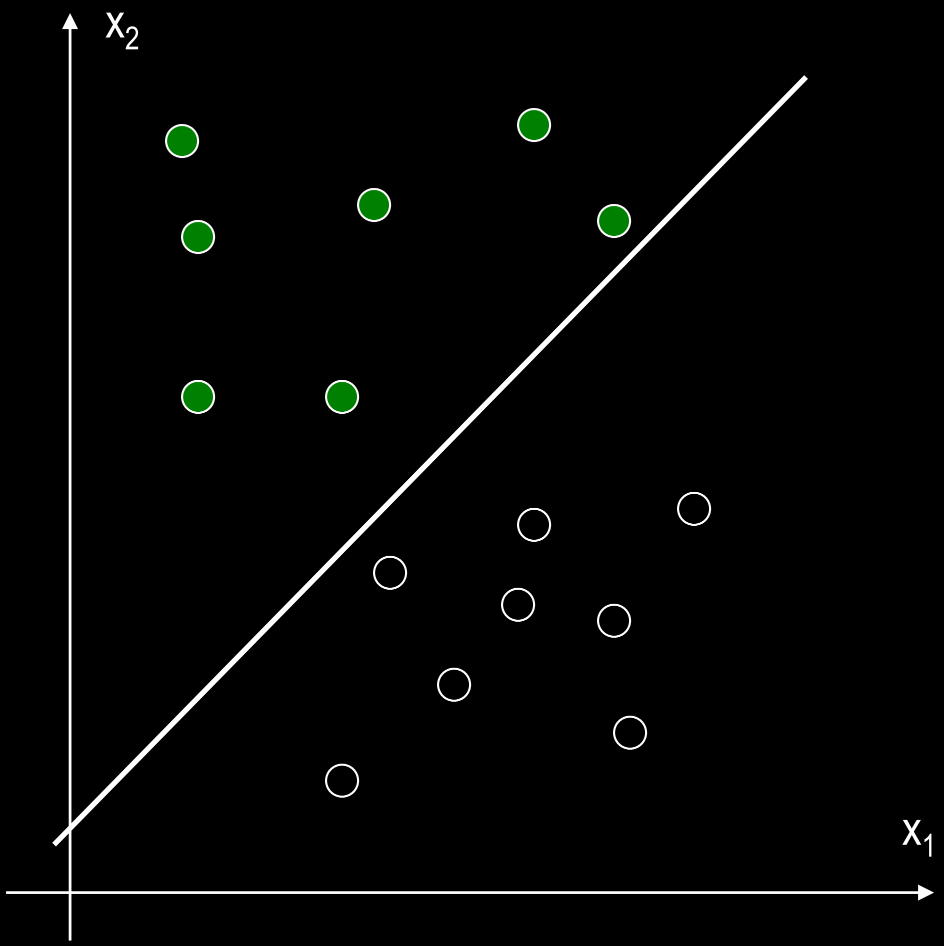
Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?



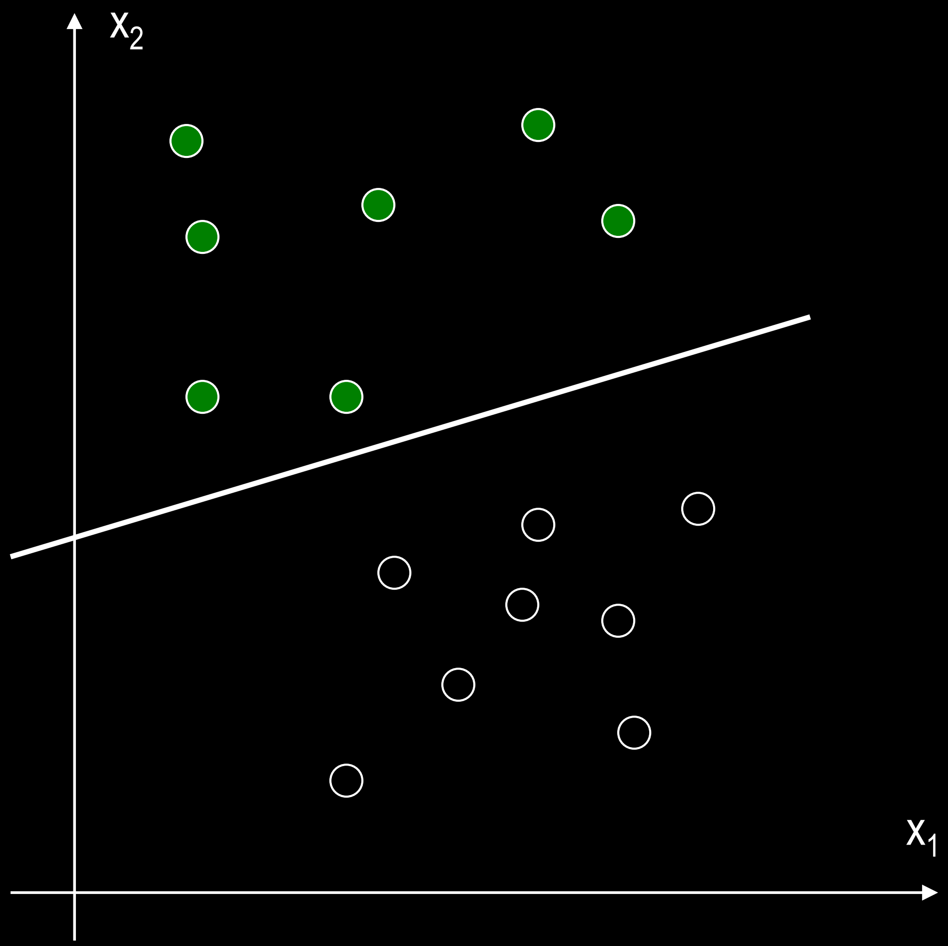
Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?



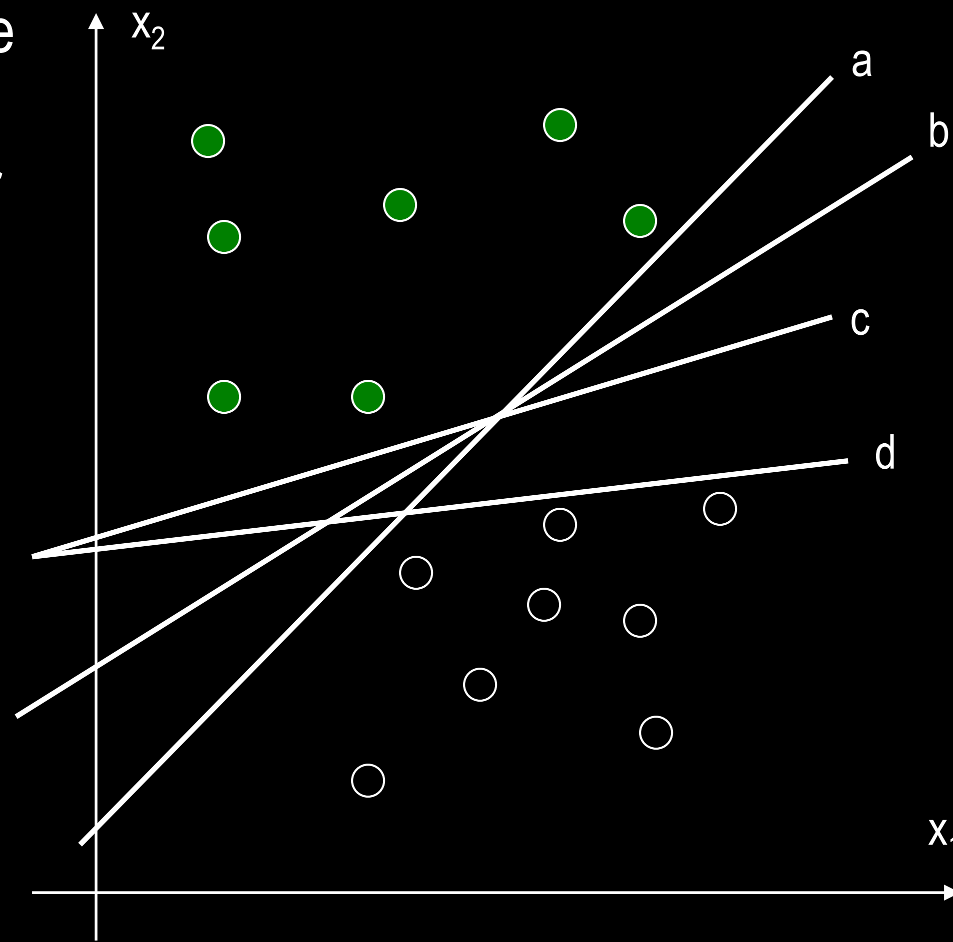
Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?



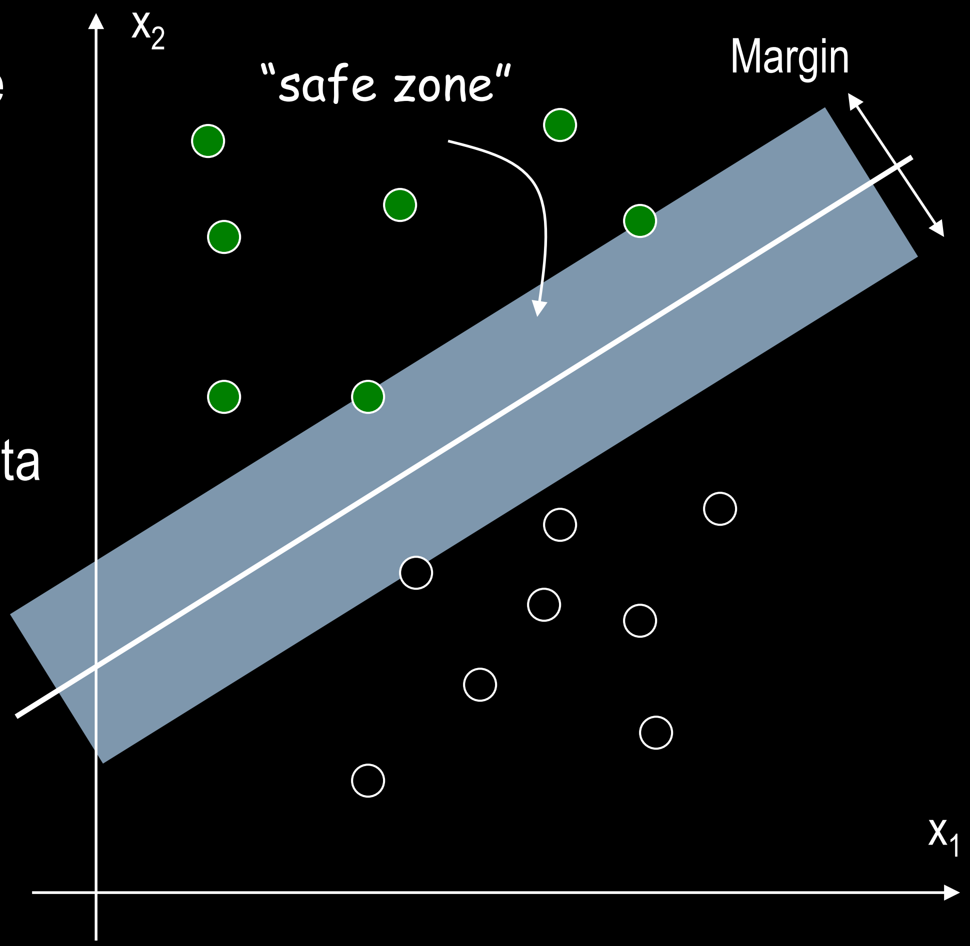
Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?
- Infinite number of answers!
- Which one is best?



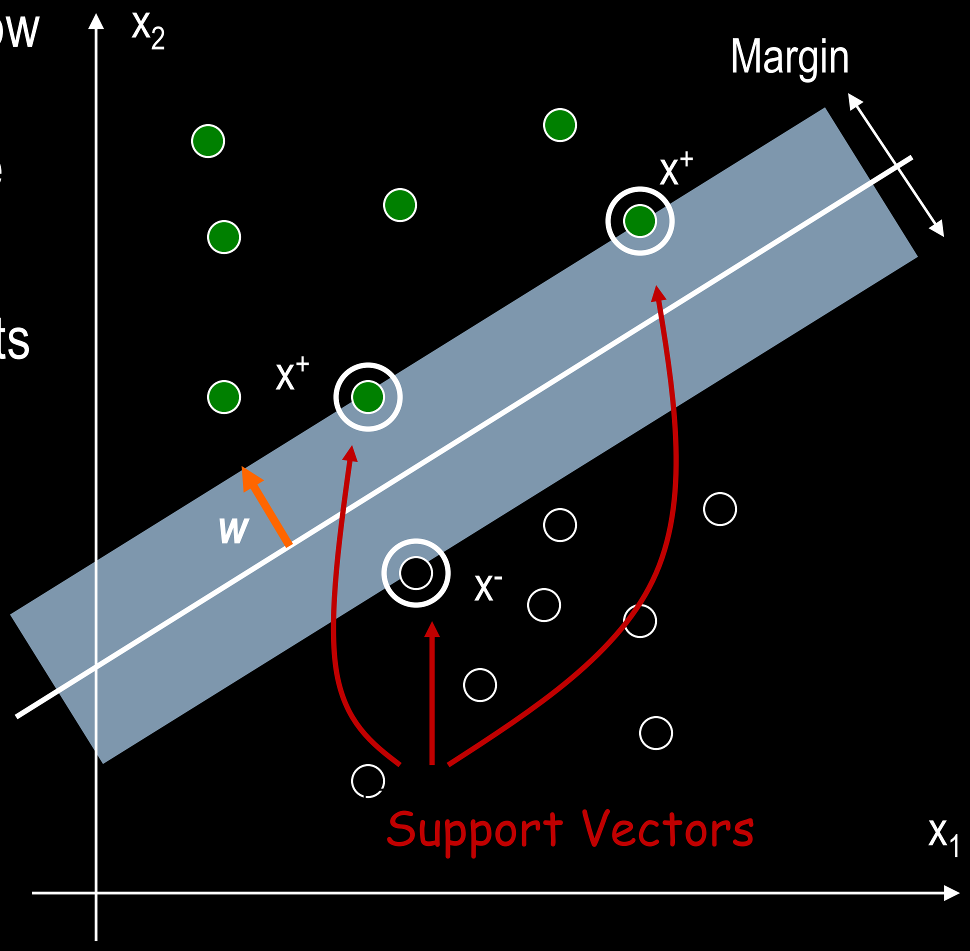
Maximum Margin Linear Classifier

- For an SVM, 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 is it the best?
  - Robust to outliners and thus strong generalization ability



Maximum Margin Linear Classifier

- Observation: All we need to know to compute the hyperplane is a subset of the points that are the support vectors
  - None of the other data points matter!
- This idea leads to the linear Support Vector Machine (SVM) classifier



SVM: How to compute the hyperplane

- Remember, machine learning is just optimization
- The function we want is $f(x) = w^T x - b$ and we need to compute w and b given a data set of xs
- The objective function is:

$$\underset{(w,b)}{\operatorname{argmin}} \quad \frac{1}{2}\|w\|^2$$

- Subject to the constraints:

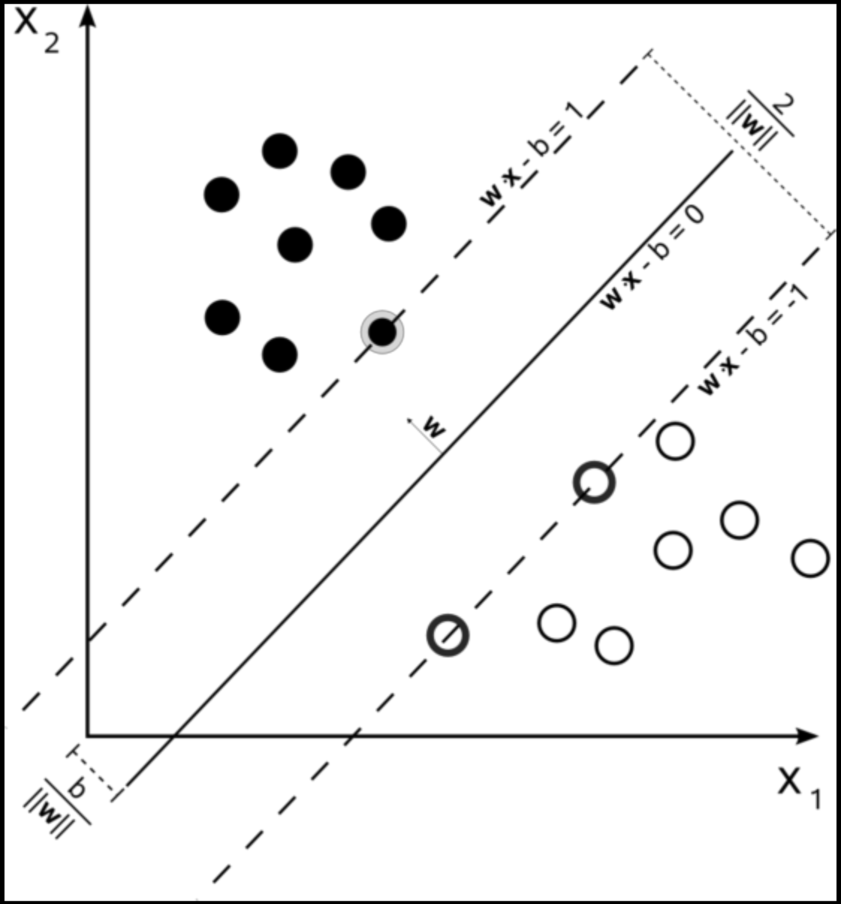
$$y_i(w^T x_i - b) \geq 1 \qquad \text{for } i = \{1, \ldots, n\}$$

This is the margin

- y_i is the label for data point x_i
  - y_i is either +1 or -1 (remember we are deciding between two classes)

SVM: Why minimize $\frac{1}{2}||w||^2$?

- $\frac{2}{||w||}$ is the width of margin
- To maximize margin, minimize $||w||$
- We actually minimize $\frac{1}{2}||w||^2$ instead b/c of mathematical convenience
  - The result will be the same as minimizing $||w||$



The SVM optimization problem

- The primal problem is a quadratic program
- Convert to the dual form using lagrange multipliers α:

$$\arg\min_{\mathbf{w},b}\max_{\boldsymbol{\alpha}\geq 0}\left\{\frac{1}{2}\|\mathbf{w}\|^2-\sum_{i=1}^{n}\alpha_i\underbrace{[y_i(\mathbf{w}\cdot\mathbf{x_i}-b)-1]}\right\}$$

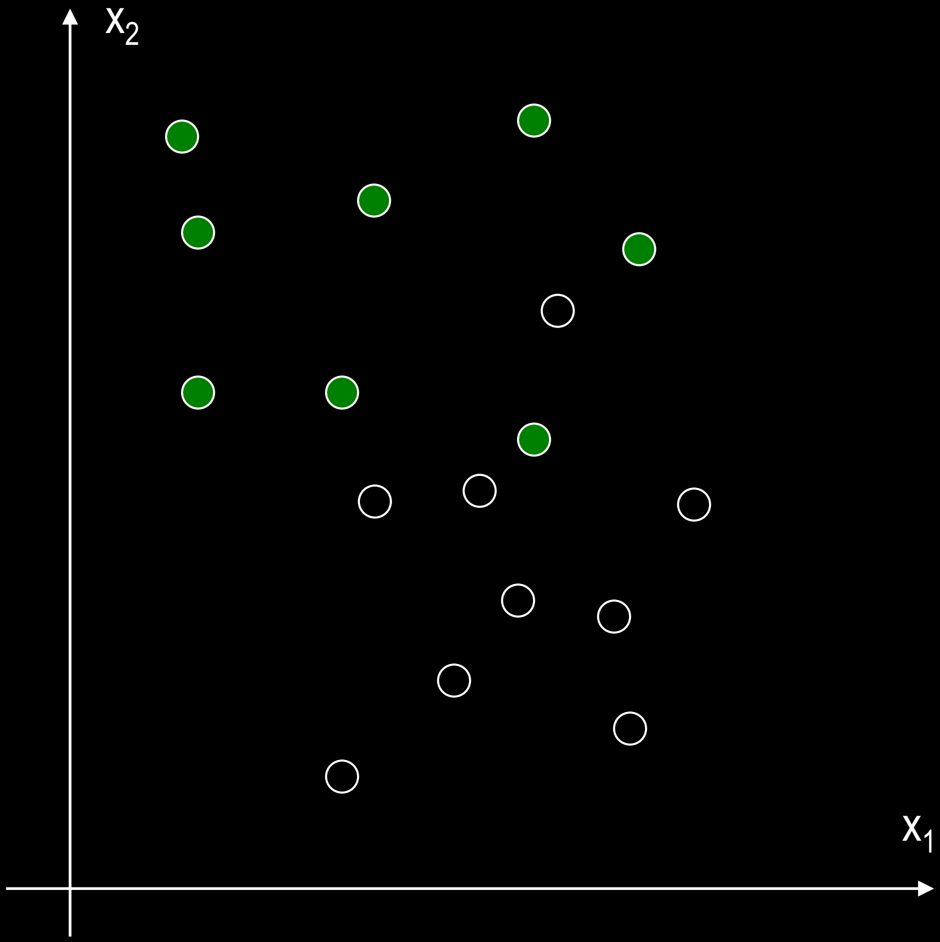
For any i where this is positive
α_i will be set to 0

α_i will only be nonzero where
$y_i(w^T x_i - b) - 1 = 0$
These data points are the support vectors

- You can solve this problem with standard quadratic programming

But what about noisy data?

- Sometimes there is no hyperplane that gives perfect separation



Soft Margin SVM for Noisy Data

- Change the constraints of the SVM by introducing a slack variable:

$$y_i(\mathbf{w}\cdot\mathbf{x_i}-b)\geq 1-\xi_i, \qquad \xi_i\geq 0 \qquad 1\leq i\leq n.$$

Slack variable

- This allows data points to be on the “wrong” side of the hyperplane
- The objective function then seeks to minimize this violation:

$$\underset{\mathbf{w},\xi,b}{\arg\min}\left\{\frac{1}{2}\|\mathbf{w}\|^2+C\sum_{i=1}^{n}\xi_i\right\}$$

Soft Margin SVM

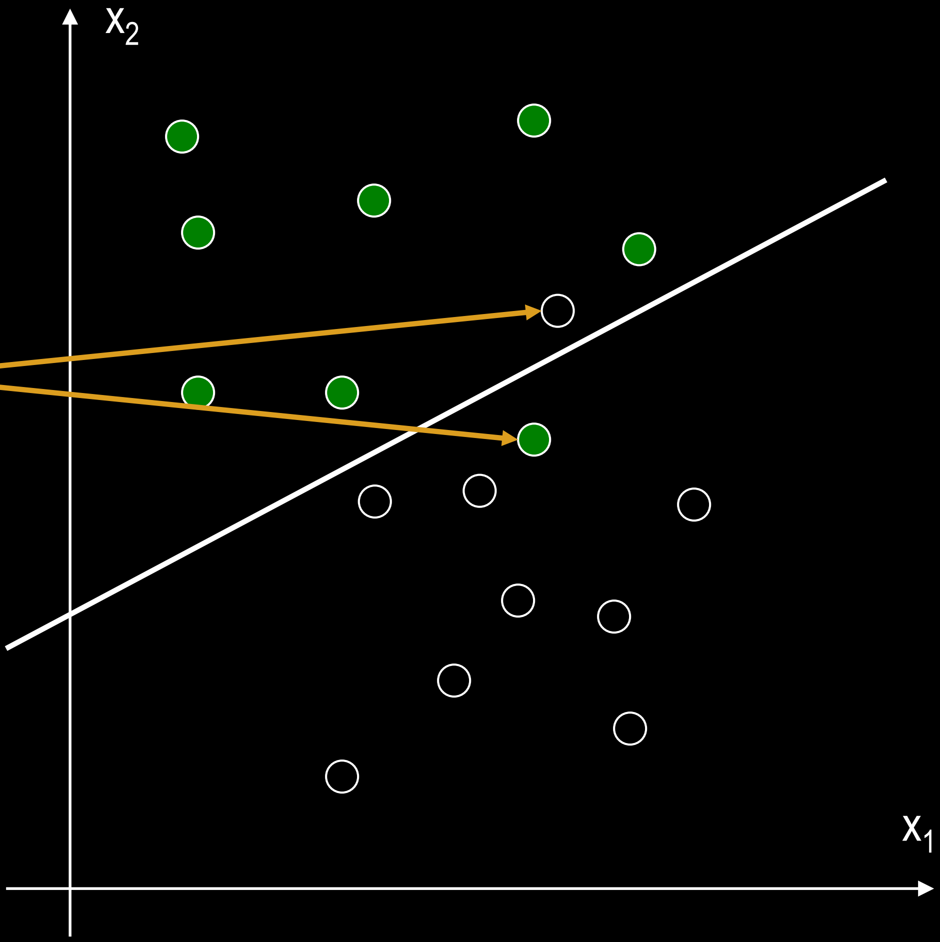
- The dual form is then:

$$\arg\min_{\mathbf{w},\xi,b}\max_{\boldsymbol{\alpha},\boldsymbol{\beta}}\left\{\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i(\mathbf{w}\cdot\mathbf{x_i} - b) - 1 + \xi_i] - \sum_{i=1}^{n}\beta_i\xi_i\right\}$$

$\alpha, \beta \geq 0$

Soft Margin SVM

- $\xi = 0$ for correctly-classified points
- $\xi_i > 0$ for misclassified points



SVMs for Regression

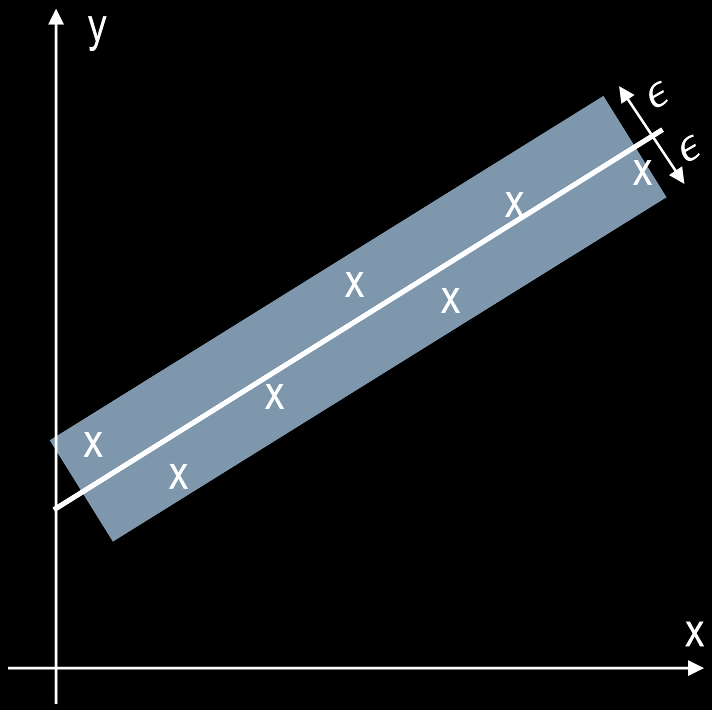
- Recall that in regression, we have a continuous-valued label y_i
- The SVM optimization problem for regression is:

$$\underset{(w,b)}{\operatorname{argmin}} \ \frac{1}{2}\|w\|^2$$

Subject to: $\quad y_i - w^T x_i - b \leq \epsilon$

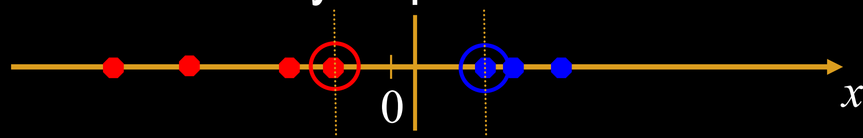
$$w^T x_i + b - y_i \leq \epsilon$$

- Assumes we don't care about approximation error as long as it is less than tolerance ϵ
- Alternatively, could also use a soft margin approach for noisy data



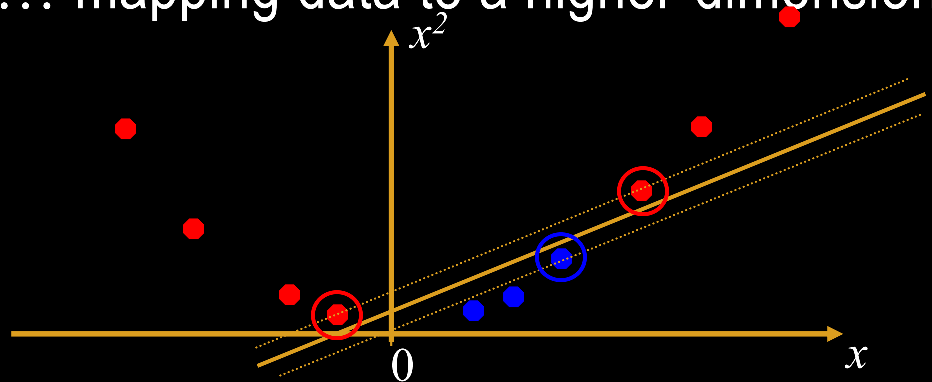
Non-linear SVMs

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



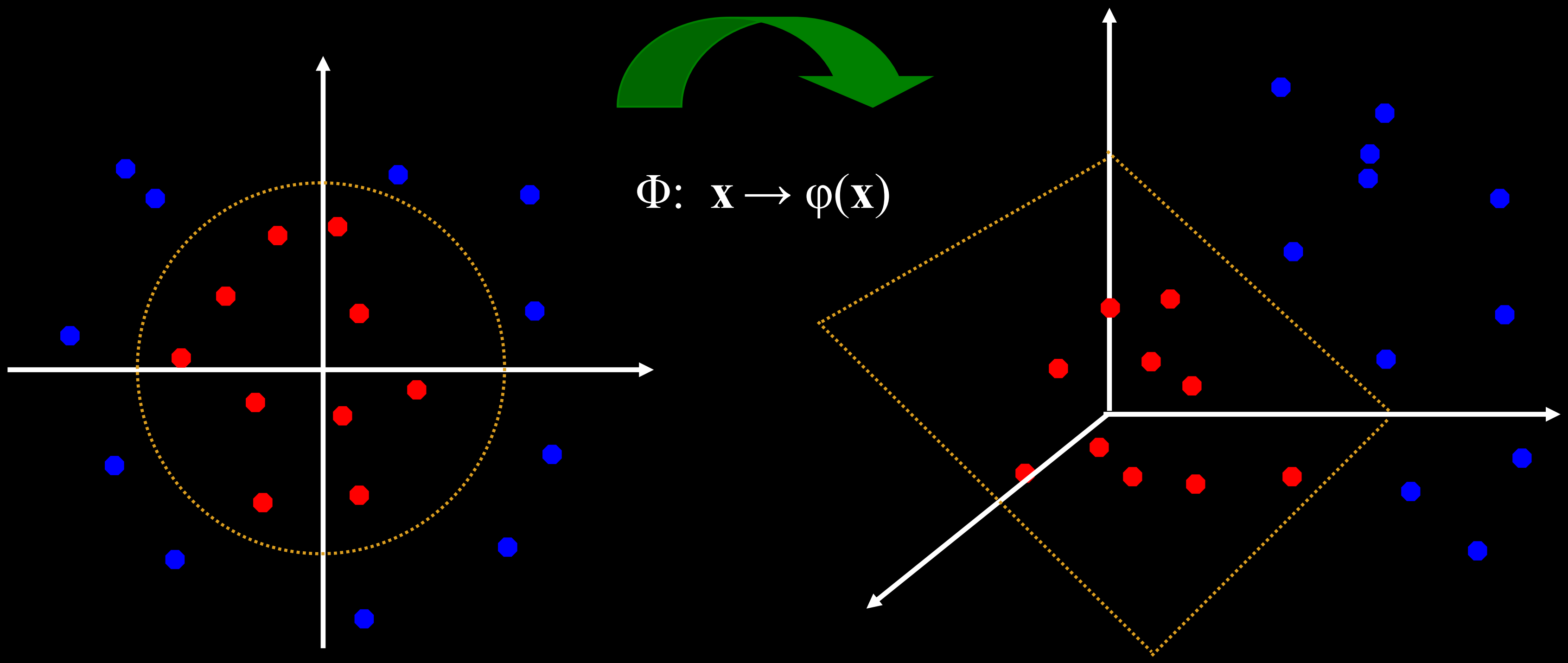
- But what are we going to do if the dataset is just too hard?

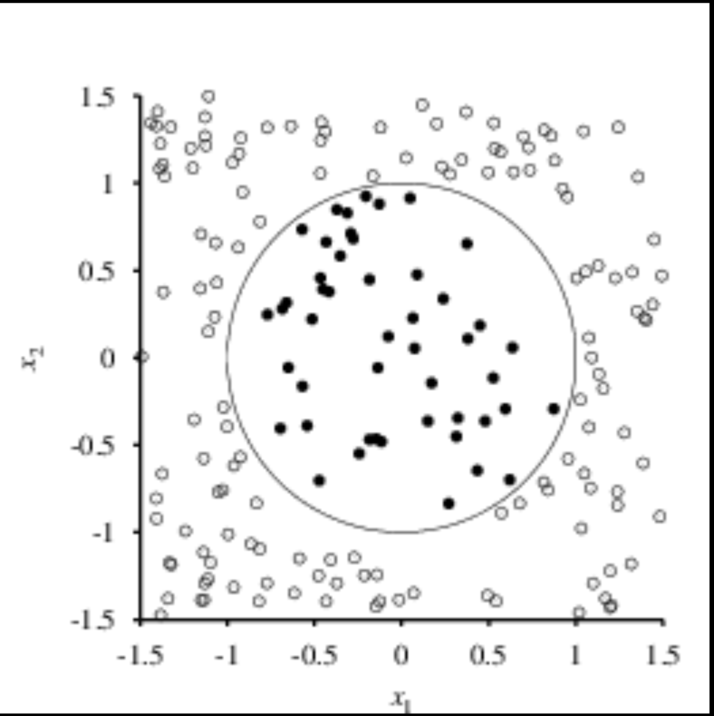
- How about… mapping data to a higher-dimensional space:

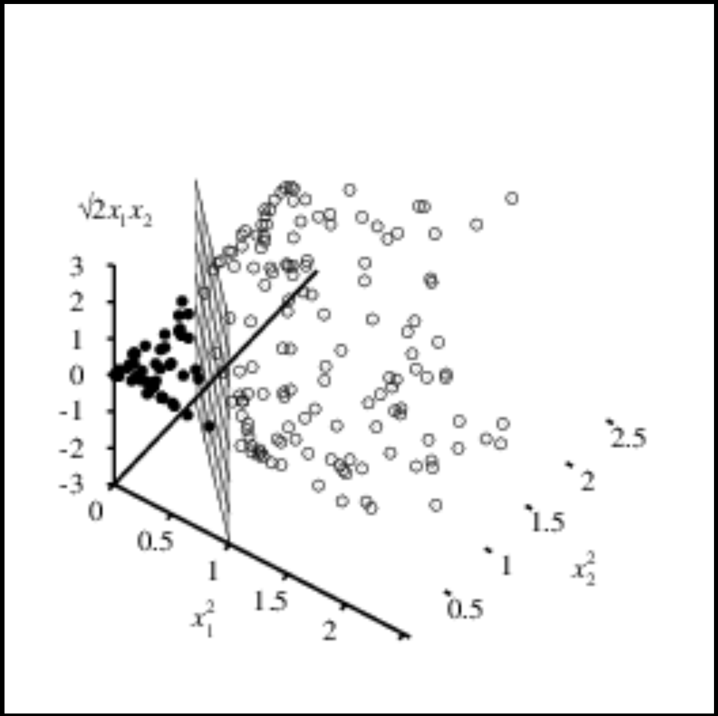


SVM Feature Space: The Kernel Trick

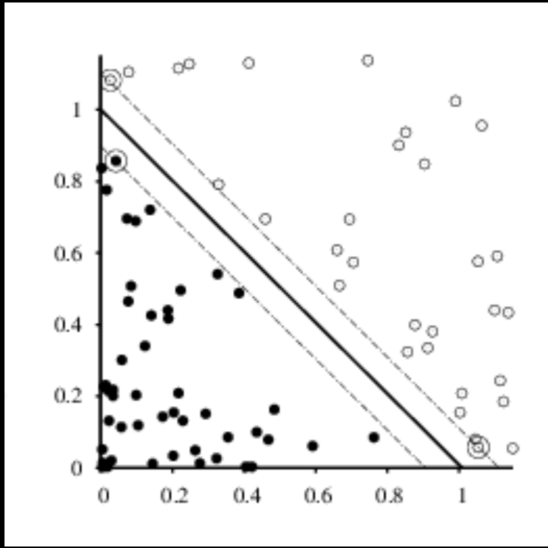
- General idea: the original input space can be mapped to some higher-dimensional feature space where the training set is linearly separable:







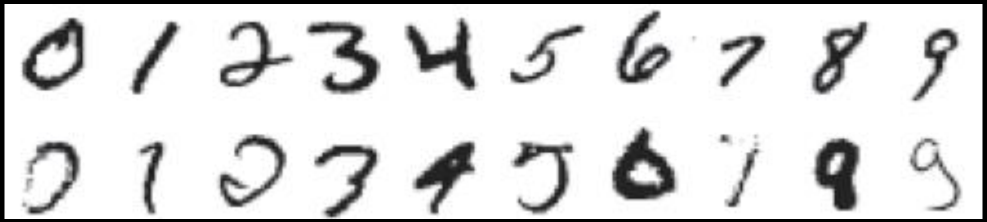
$$x_1^2 + x_2^2 \leq 1 \xrightarrow{\Phi} (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$



Closeup of the decision boundary

SVM Applications

- SVM has been used successfully in many real-world problems
  - text (and hypertext) categorization
  - image classification
  - bioinformatics (Protein classification, Cancer classification)
  - hand-written character recognition:



- Often the first algorithm you try for a learning problem

SVM Tools

- **SVM*light***
  - Written in C (should be fast!)
  - Run from command line
  - http://svmlight.joachims.org/

- LibSVM
  - Python interface
  - http://www.csie.ntu.edu.tw/~cjlin/libsvm/

- SciKit Learn
  - Python
  - https://scikit-learn.org/stable/modules/svm.html

Summary

- For supervised learning, the aim is to find a simple* hypothesis approximately consistent with training data
- Decision tree learning uses information gain to decide next variable on which to split
- Learning performance = prediction accuracy measured on test set
- Support vector machines learn a linear classifier, robust to some noise with slack variables
  - Can also be used for regression

Homework

- Read AI book Ch. 18.6-18.9