FIXED INCOME ANALYSIS

FALL 2019

Agenda

- Ask the Fed
- Intro to Machine Learning

AI & Machine Learning

- Will AI replace us in the future?

- Two types of jobs:
  - Research, Design and develop AI tools
  - Help and maintenance of AI, e.g.
    - Mechanical engineers and maintenance technicians
    - Manufacturing and electrical engineers
    - Surgical technicians working with robotic tools

Machine Learning

- Machine learning is programming computers to optimize a performance criterion using example data or past experience.
- There is no need to “learn” to calculate payroll
- Learning is used when:
  - Human expertise does not exist (navigating on Mars)
  - Humans are unable to explain their expertise (speech recognition)
  - Solution changes in time (routing on a computer network)
  - Solution needs to be adapted to particular cases (user biometrics)

Machine Learning

	Supervised Learning	*Unsupervised Learning*
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

Machine Learning

- Apply a prediction function to a feature representation of the image to get the desired output:

f() = “apple”

f() = “tomato”

f() = “cow”

Machine Learning

$$y = f(\mathbf{x})$$

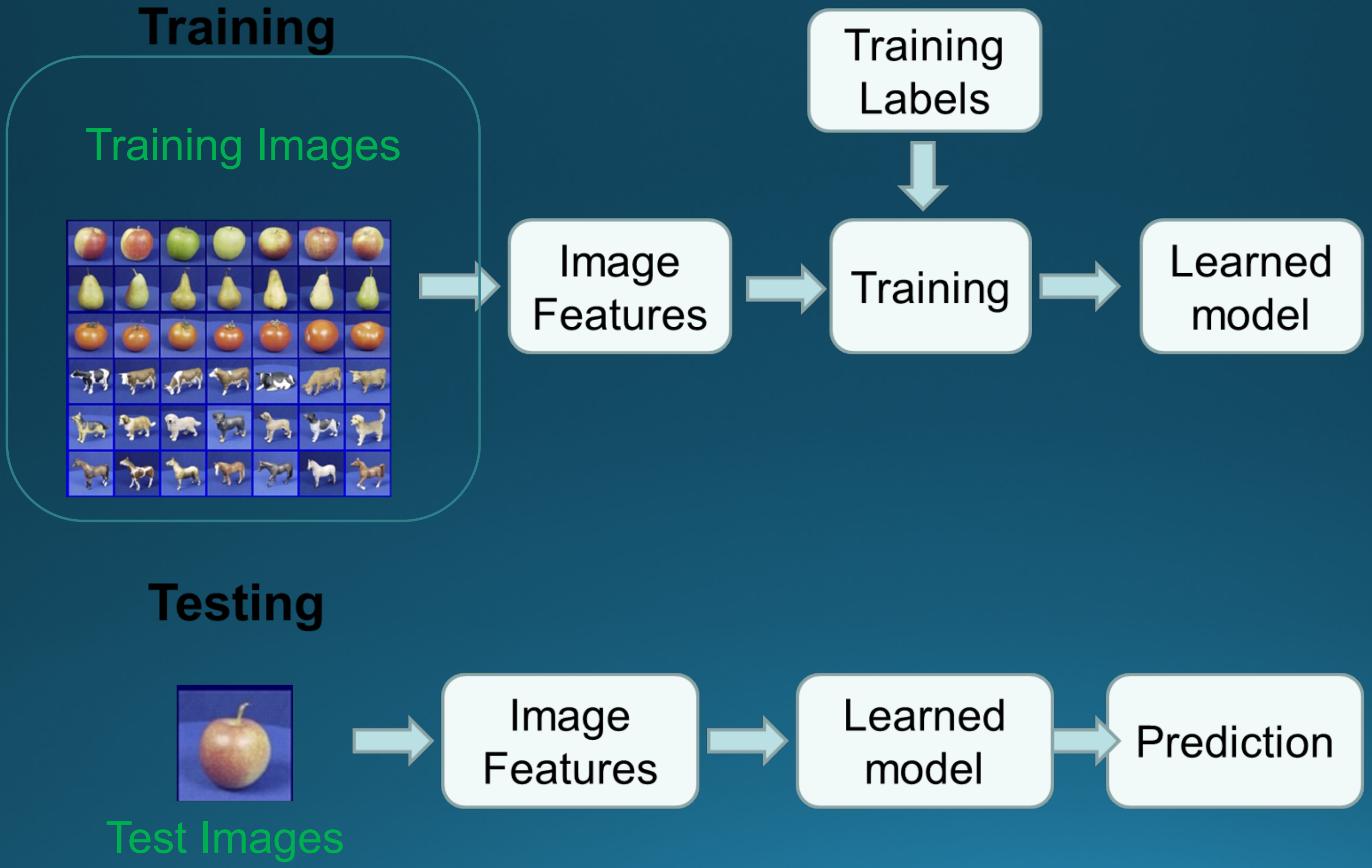
output

prediction function

Image feature

- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

Machine Learning



Generalization

Training set (labels known)

Test set (labels unknown)

- How well does a learned model generalize from the data it was trained on to a new test set?

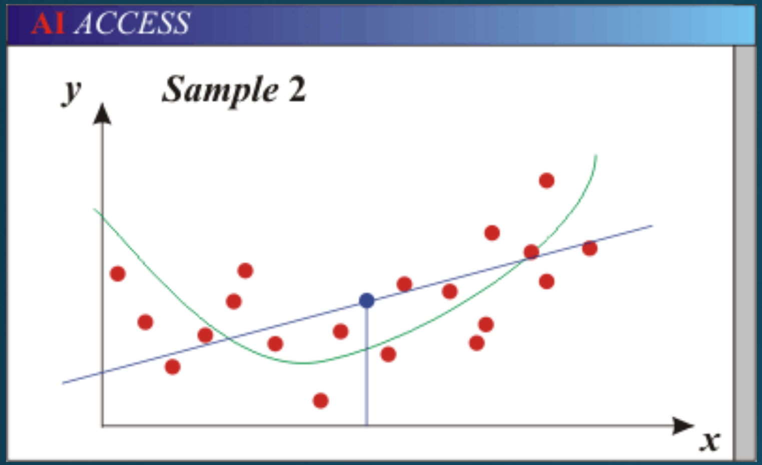
Slide credit: L. Lazebnik

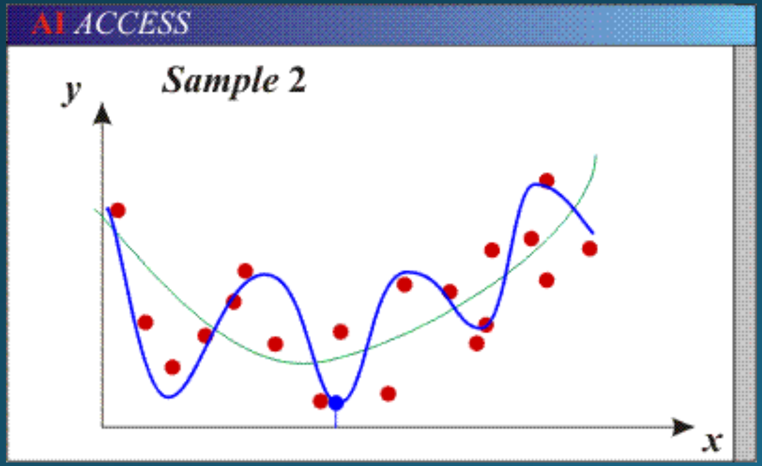
Generalization

- Components of generalization error
  - **Bias:** how much the average model over all training sets differ from the true model?
    - Error due to inaccurate assumptions/simplifications made by the model
  - **Variance:** how much models estimated from different training sets differ from each other
- **Underfitting:** model is too “simple” to represent all the relevant class characteristics
  - High bias and low variance
  - High training error and high test error
- **Overfitting:** model is too “complex” and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error

Slide credit: L. Lazebnik

Bias-Variance Trade-off

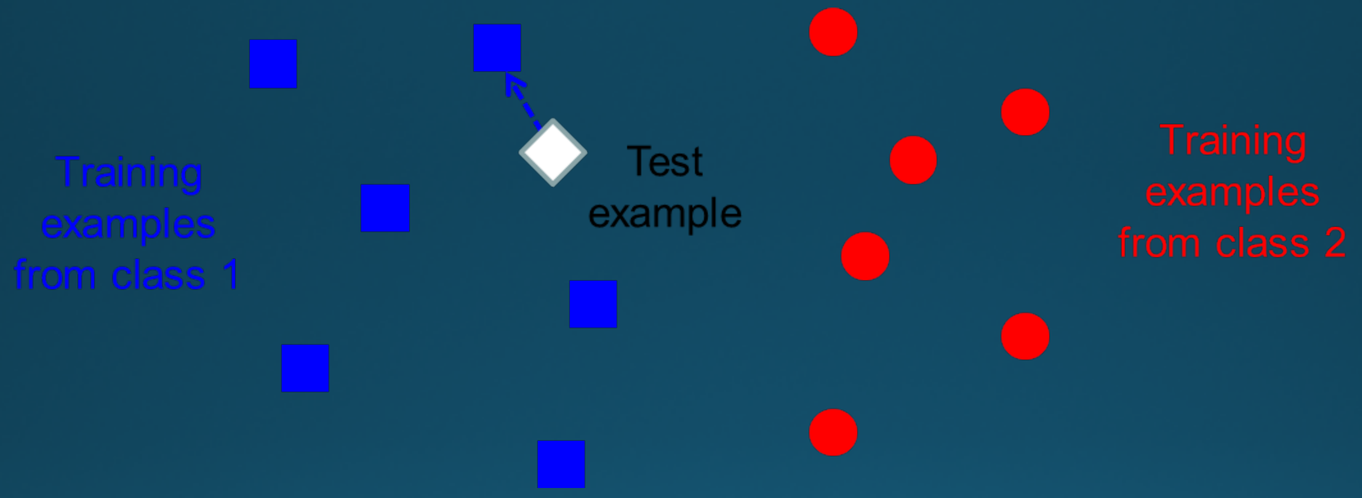




- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).
- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).

Slide credit: D. Hoiem

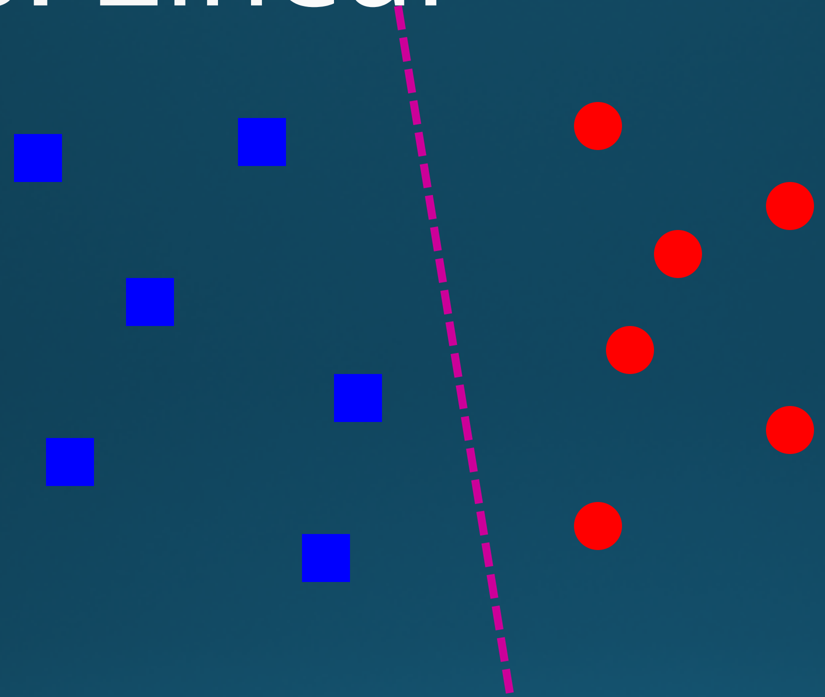
Classifiers: Nearest neighbor



$f(\mathbf{x})$ = label of the training example nearest to $\mathbf{x}$

- All we need is a distance function for our inputs
- No training required!

Classifiers: Linear



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Slide credit: L. Lazebnik

Many classifiers to choose from

- SVM
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- K-nearest neighbor
- RBMs
- Etc.

Which is the best one?

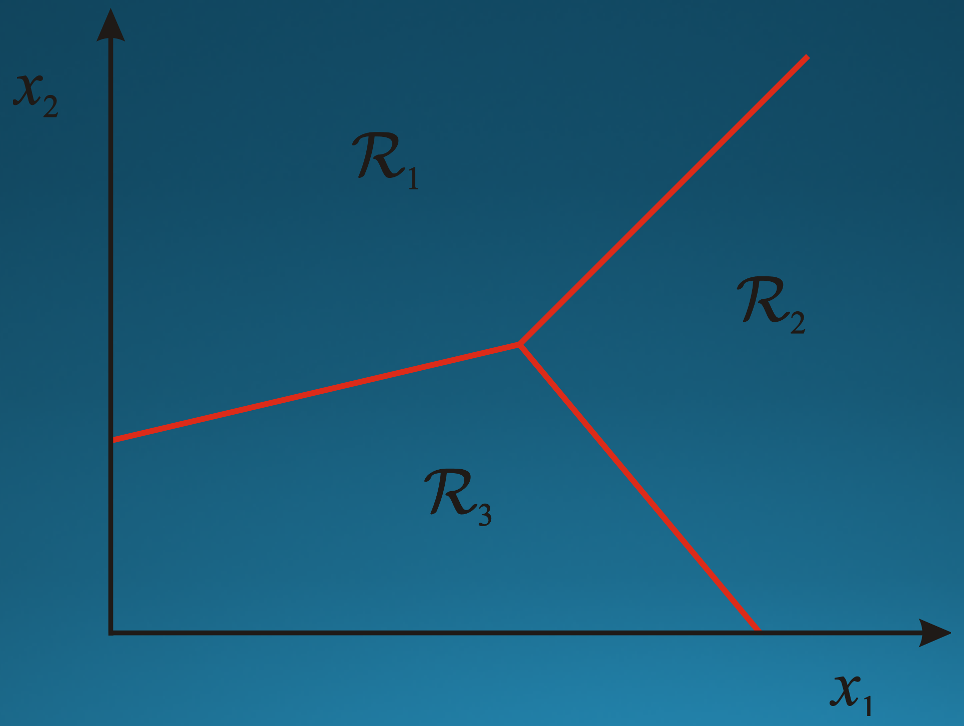
Slide credit: D. Hoiem

Very brief tour of some classifiers

- **K-nearest neighbor**
- **SVM**
- **Boosted Decision Trees**
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- RBMs
- Etc.

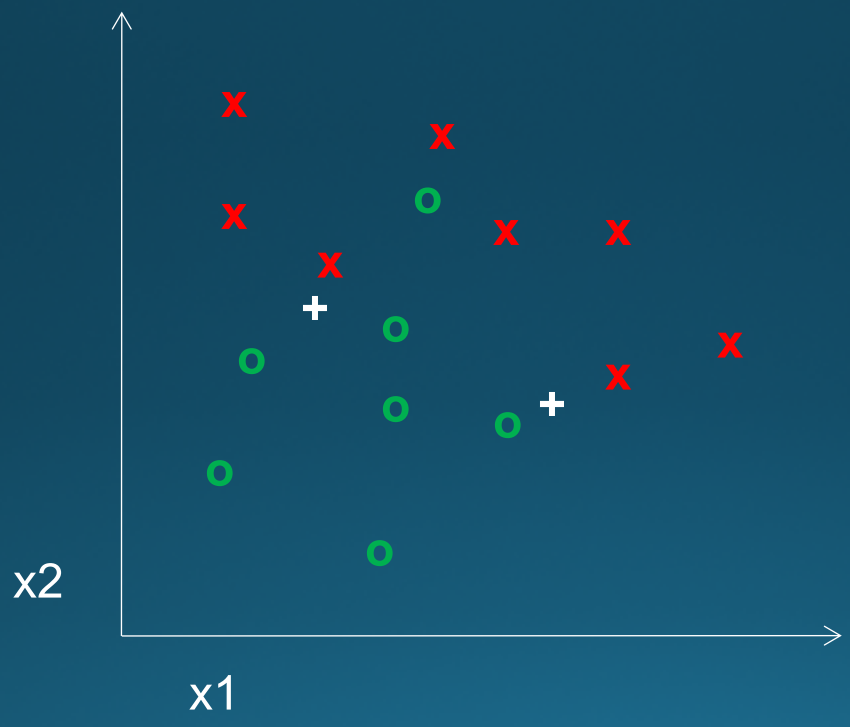
Classification

- Assign input vector to one of two or more classes
- Any decision rule divides input space into *decision regions* separated by *decision boundaries*

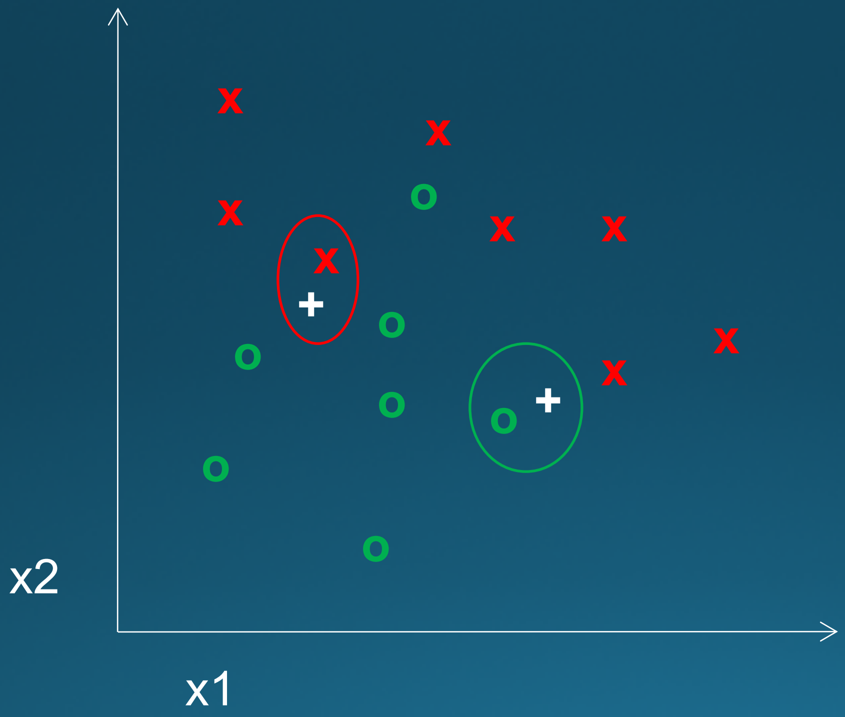


Slide credit: L. Lazebnik

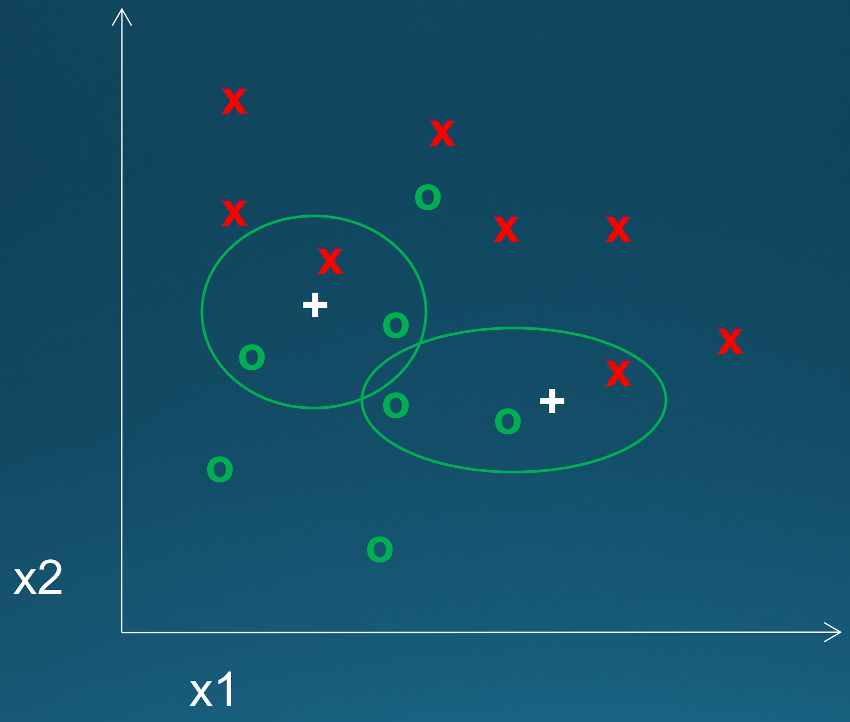
K-nearest neighbor



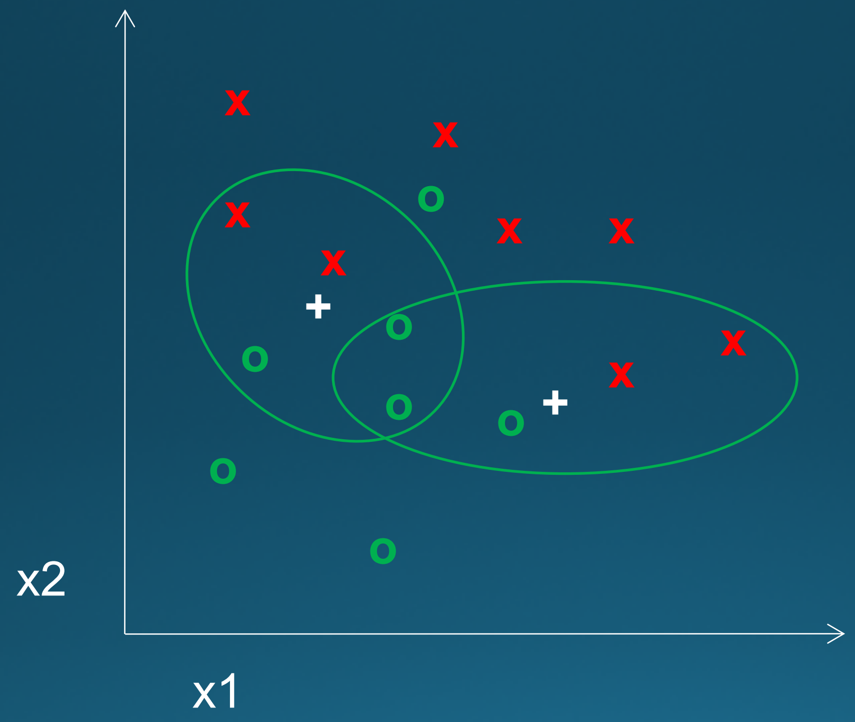
1-nearest neighbor



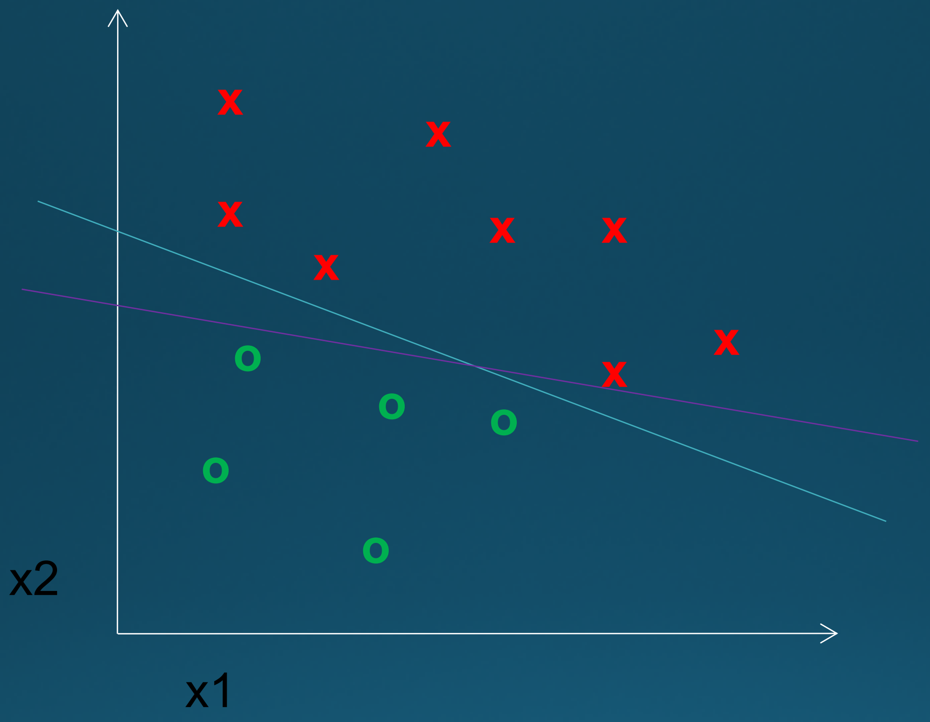
3-nearest neighbor



5-nearest neighbor



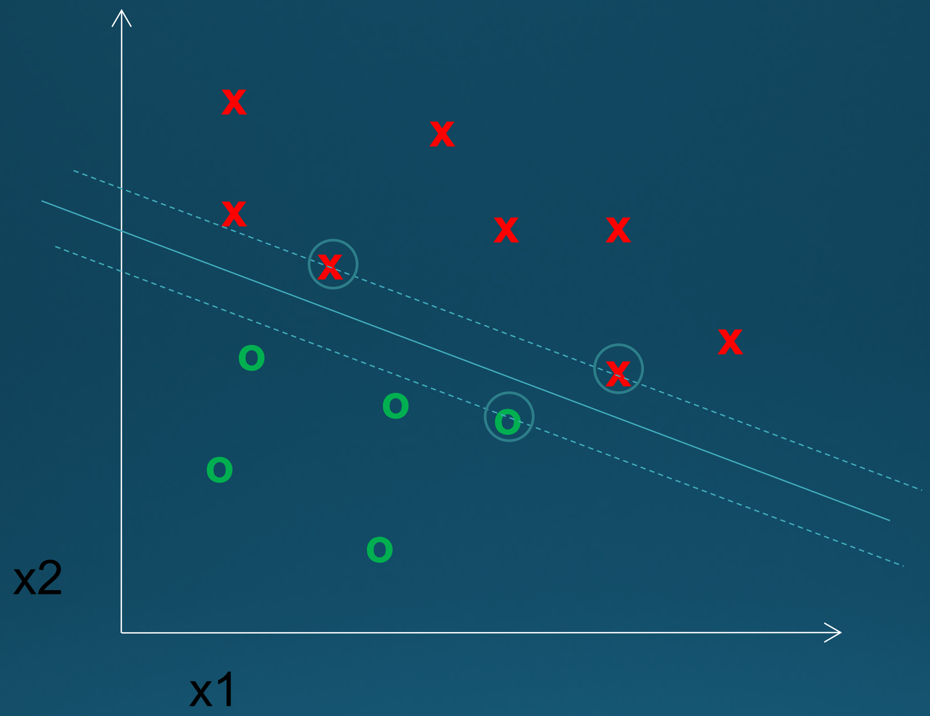
Classifiers: Linear SVM



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

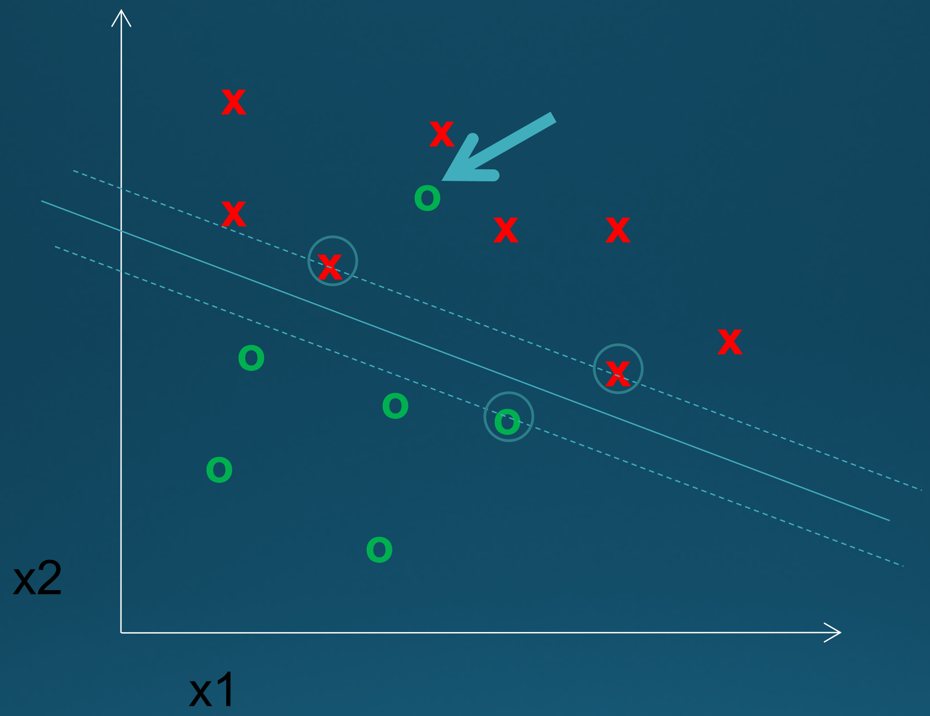
Classifiers: Linear SVM



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \operatorname{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Classifiers: Linear SVM

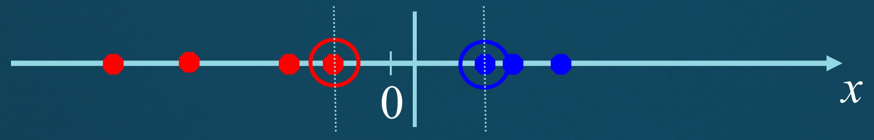


- Find a *linear function* to separate the classes:

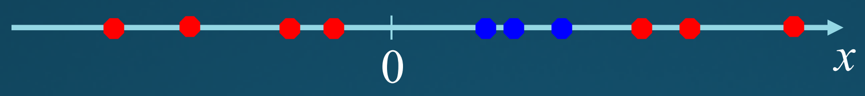
$$f(\mathbf{x}) = \operatorname{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Nonlinear SVMs

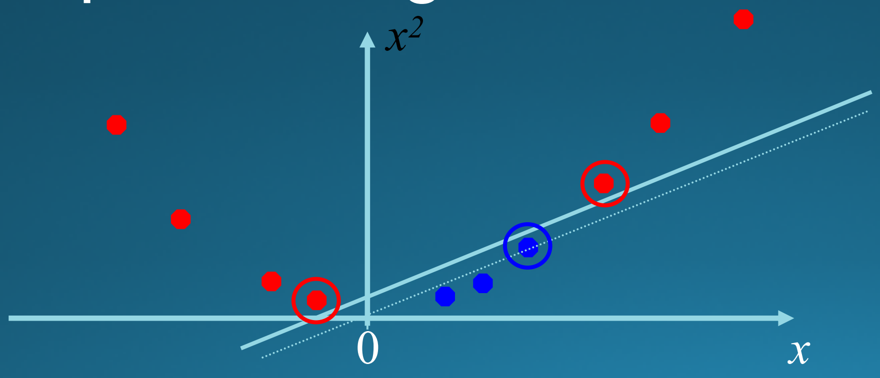
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?

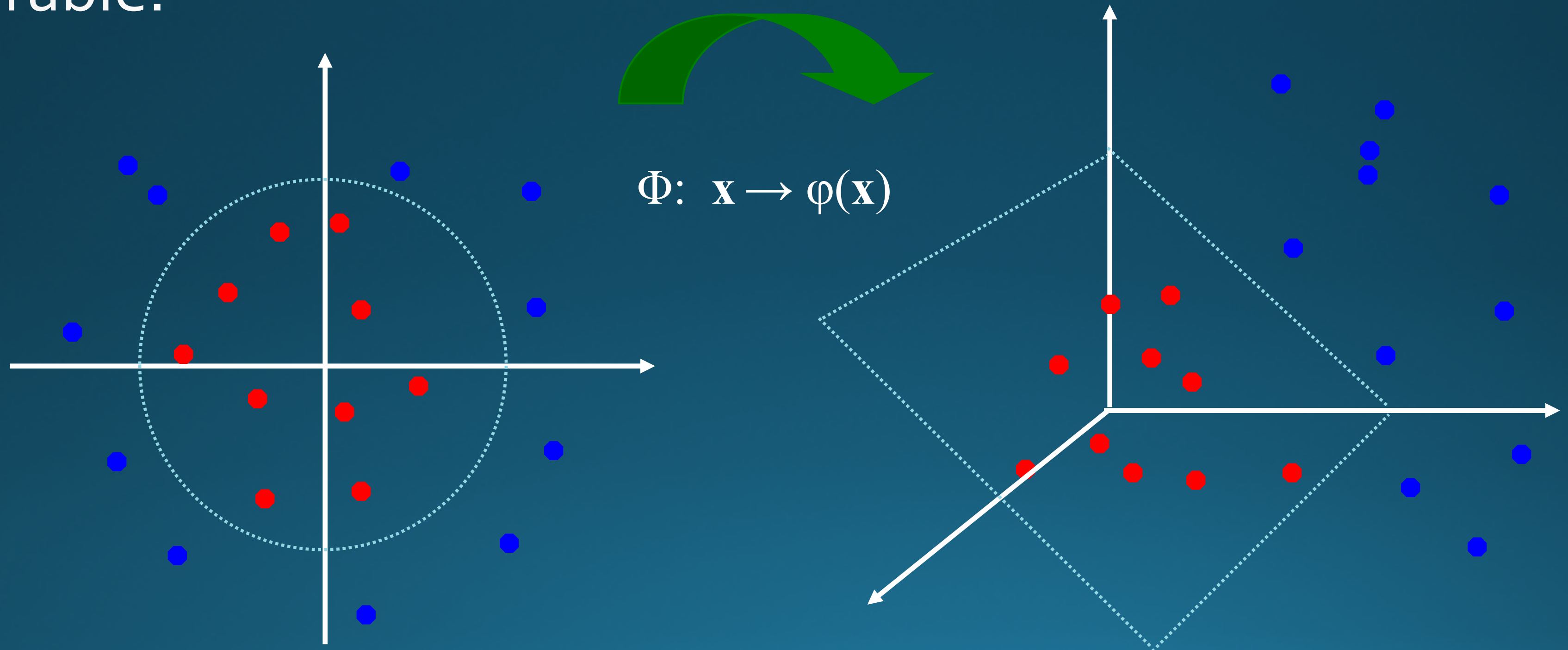


- We can map it to a higher-dimensional space:



Nonlinear SVMs

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



Nonlinear SVMs

- *The kernel trick*: instead of explicitly computing the lifting transformation $\varphi(\mathbf{x})$, define a kernel function K such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$$

(to be valid, the kernel function must satisfy *Mercer's condition*)

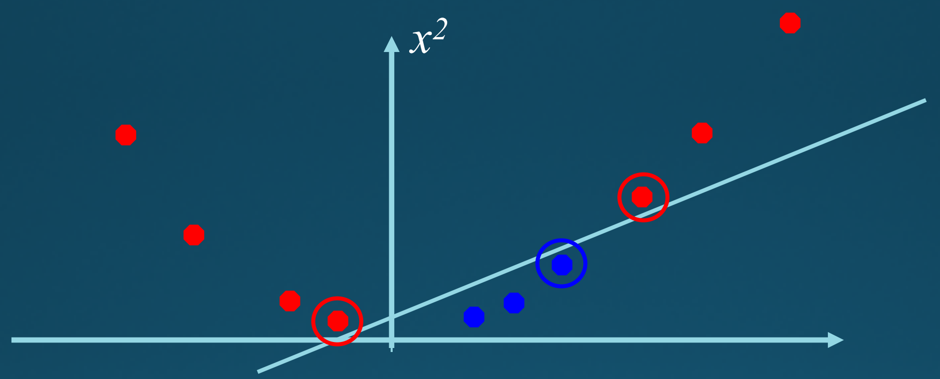
- This gives a nonlinear decision boundary in the original feature space:

$$\sum_i \alpha_i y_i \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}) + b = \sum_i \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Nonlinear kernel: Example

- Consider the mapping $\varphi(x) = (x, x^2)$



$$\varphi(x) \cdot \varphi(y) = (x, x^2) \cdot (y, y^2) = xy + x^2 y^2$$

$$K(x, y) = xy + x^2 y^2$$

SVMs: Pros and cons

- Pros
  - Many publicly available SVM packages: http://www.kernel-machines.org/software
  - Kernel-based framework is very powerful, flexible
  - SVMs work very well in practice, even with very small training sample sizes

- Cons
  - No “direct” multi-class SVM, must combine two-class SVMs
  - Computation, memory
    - During training time, must compute matrix of kernel values for every pair of examples
    - Learning can take a very long time for large-scale problems