# Artificial Intelligence

# Machine Learning Intro

Instructor: Qiang Yu

# Why Study Learning?

- Considered a hallmark of intelligence
- Viewed as way to reduce programming burden
  - Not enough programmers in the world to produce custom solutions to all problems – even if we knew how
  - Programmers are expensive!
- Many algorithms assume parameters that are difficult to determine exactly a priori
  - What is the right formula to filter spam?
  - When should your smart thermostat turn on the heat?

# What is Machine Learning?

- Learning Element
  - The thing that learns
- Performance Element
  - Objective measure of progress
- Learning is simply an increase in the ability of the learning element over time (with data) to achieve the task specified by the performance element

# ML vs. Statistics?

- Machine learning is:
  - Younger
  - More empirical
  - More algorithmic
  - (arguably) More practical
  - (arguably) More decision theoretic

Look at this cool result! Maybe somebody can explain why it works later?

- Statistics is:
  - More mature
  - (arguably) More formal and rigorous

Let's model this situation and prove that we converge to a consistent answer!

# Different kinds of learning… (Feedback)

- Supervised learning:
  - Someone gives us examples and the right answer (*label*) for those examples
  - We have to predict the right answer for unseen examples
- Unsupervised learning:
  - We see examples but get no feedback (no labels)
  - We need to find patterns in the data
- Semi-supervised learning:
  - Small amount of labeled data, large amount of unlabeled data
- Reinforcement learning:
  - We take actions and get rewards
  - Have to learn how to get high rewards

# Types of Supervised Learning

- Training input:
  - Feature vector for each datum: $x_1 \ldots x_n$
  - Target value: y

- Classification – assigning labels/classes
- Regression – assigning real numbers

# Features and Targets

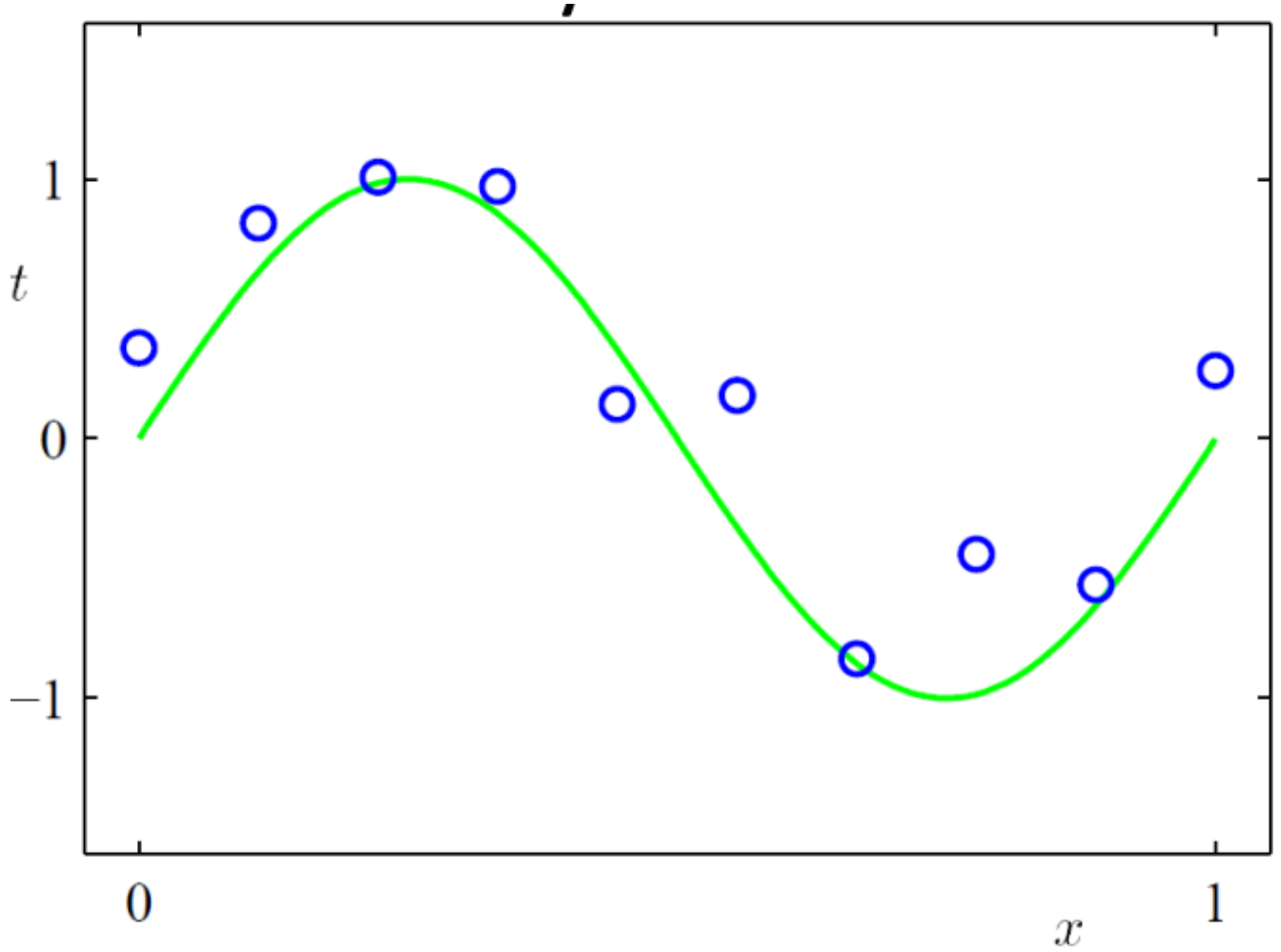
- Features can be anything
  - Images, sounds, text
  - Real values (height, weight)
  - Integers, or binaries
- Targets can be discrete classes:
  - Safe mushrooms vs. poisonous
  - Malignant vs. benign
  - Good credit risk vs. bad
  - Label of image
- Or numbers
  - Selling price of house
  - Life expectancy

# How Most Supervised Learning Algorithms Work

- Main idea: Minimize error on training set
- How this is done depends on:
  - Hypothesis space
  - Type of data

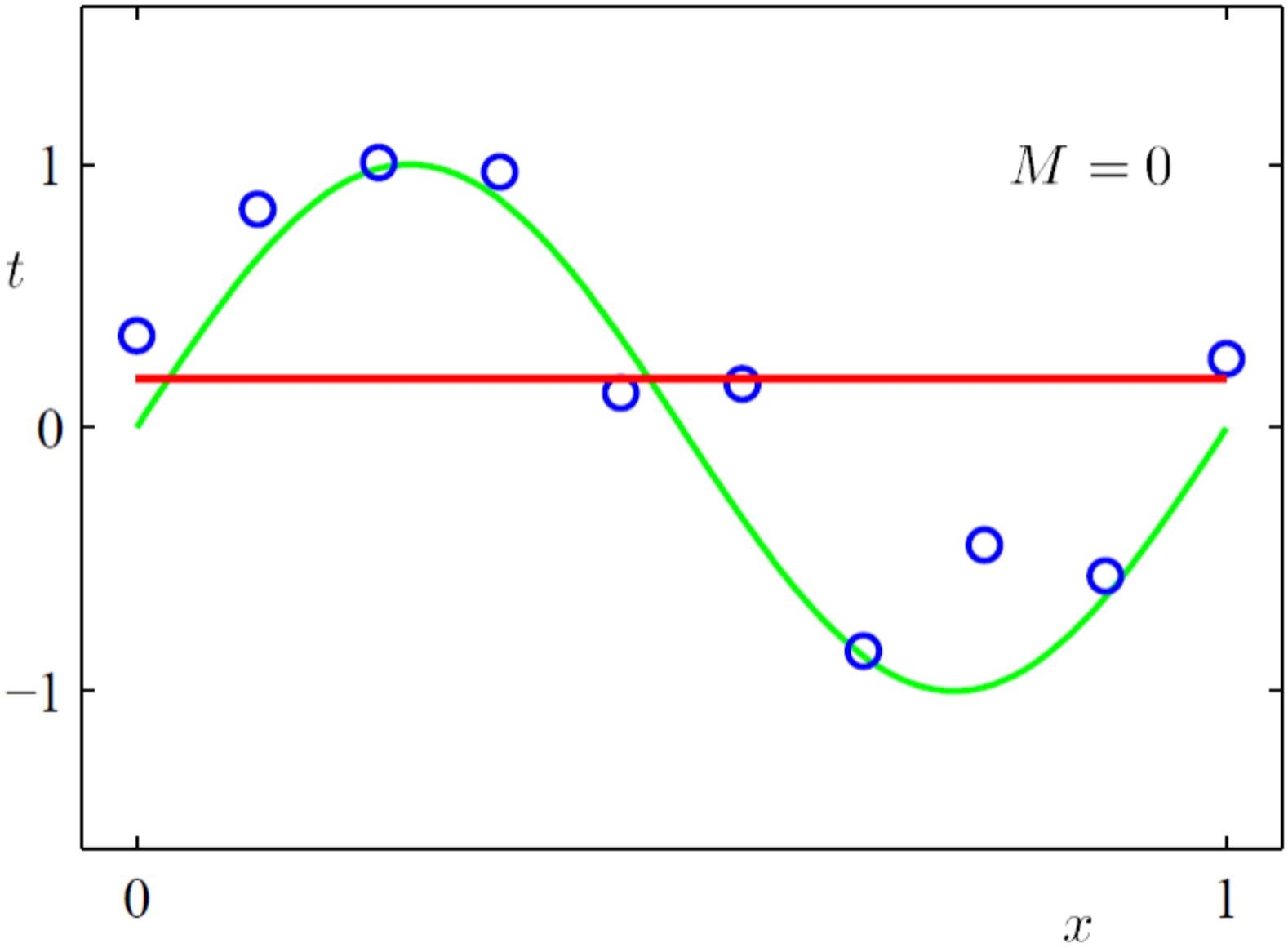
- Some approaches use a regularizer or prior to trade off training set error vs. hypothesis space complexity

# What is the Best Choice of Polynomial?

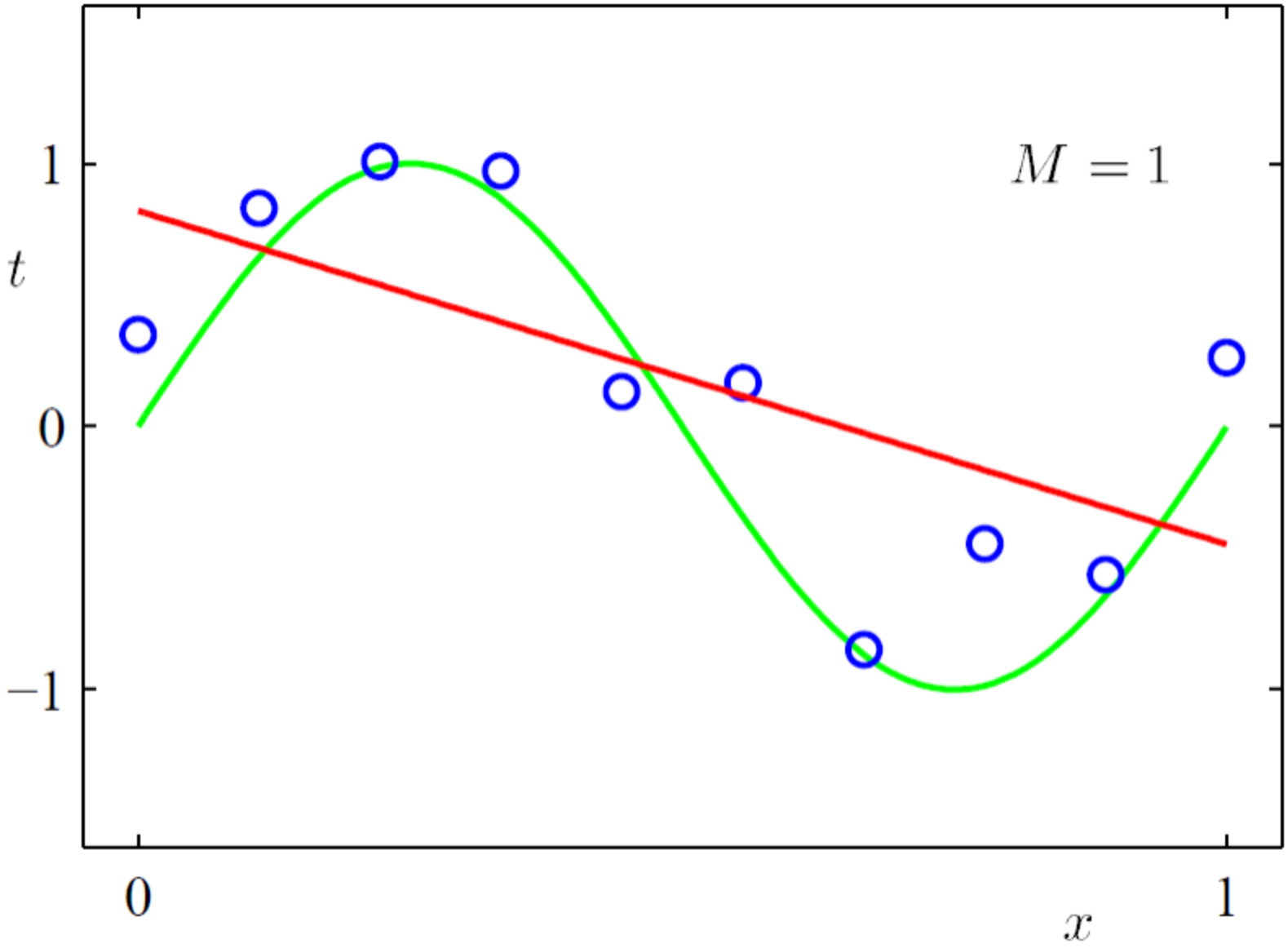



Noisy Source Data

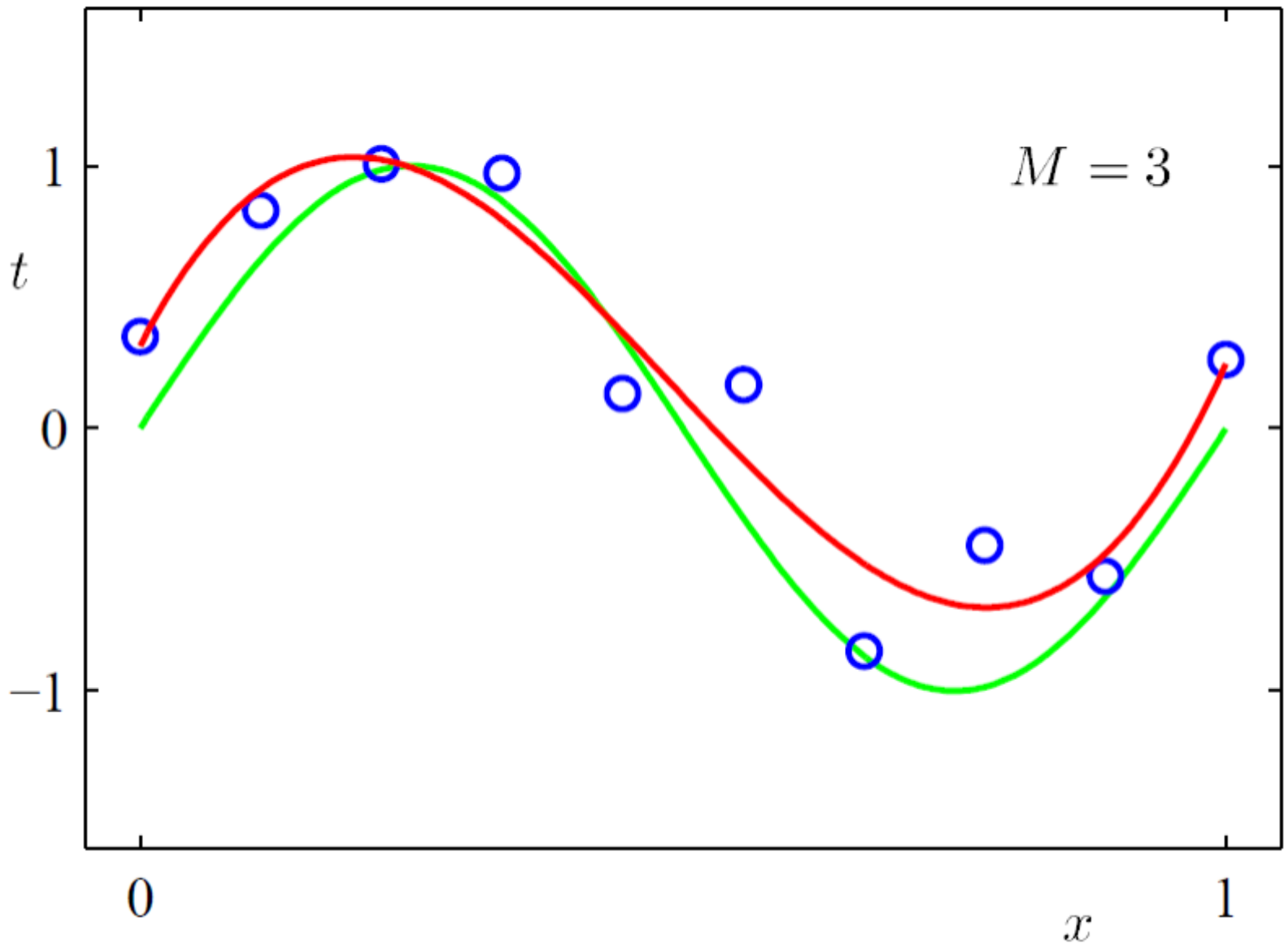
# Degree 0 Fit



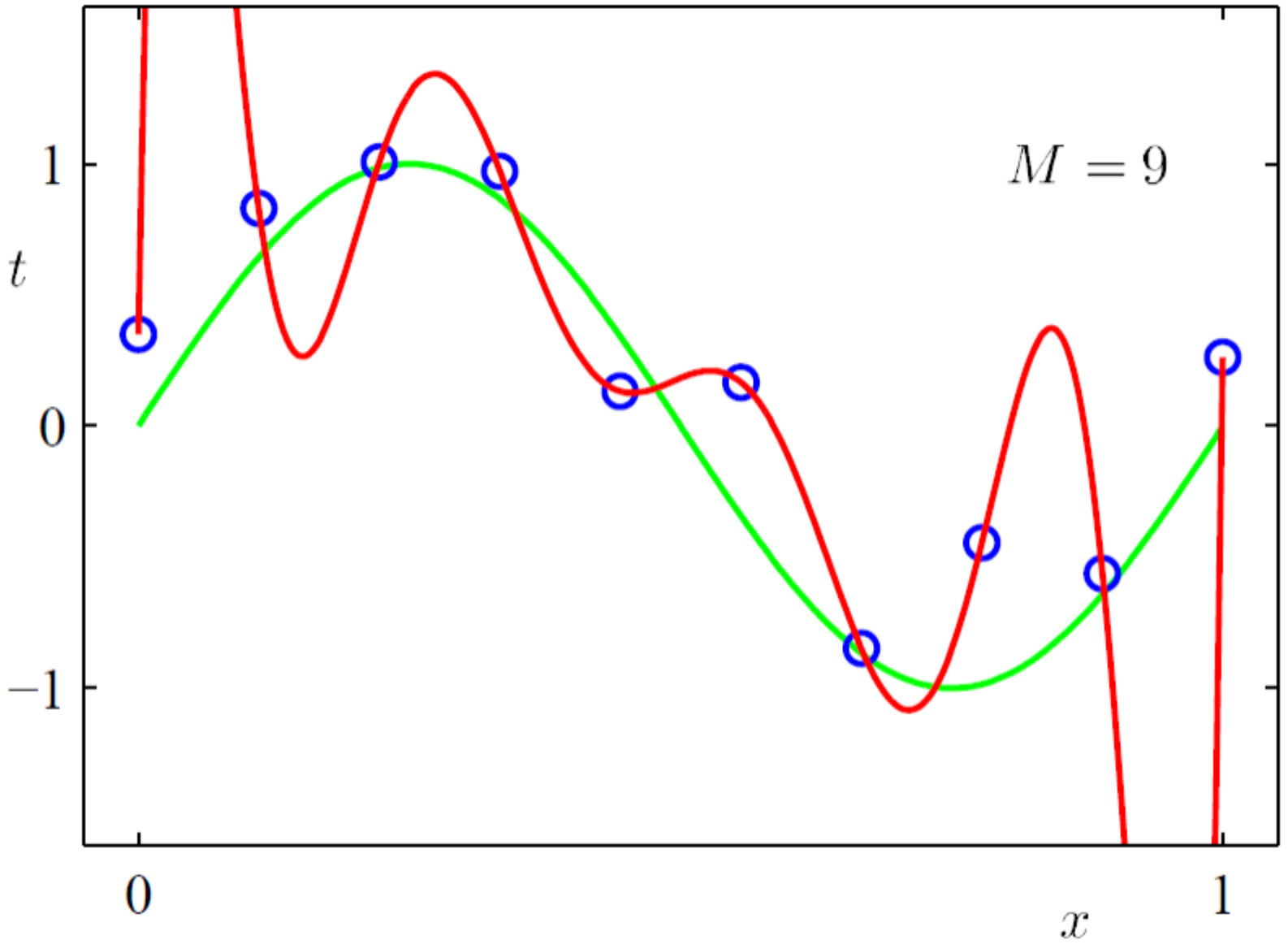
# Degree 1 Fit



# Degree 3 Fit

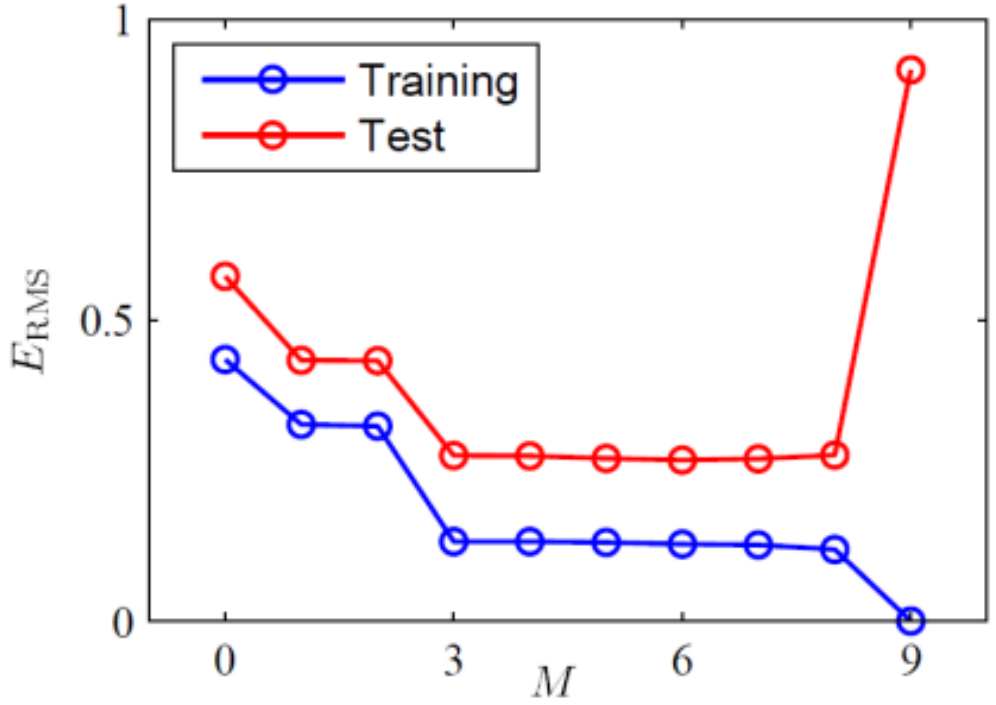


# Degree 9 Fit



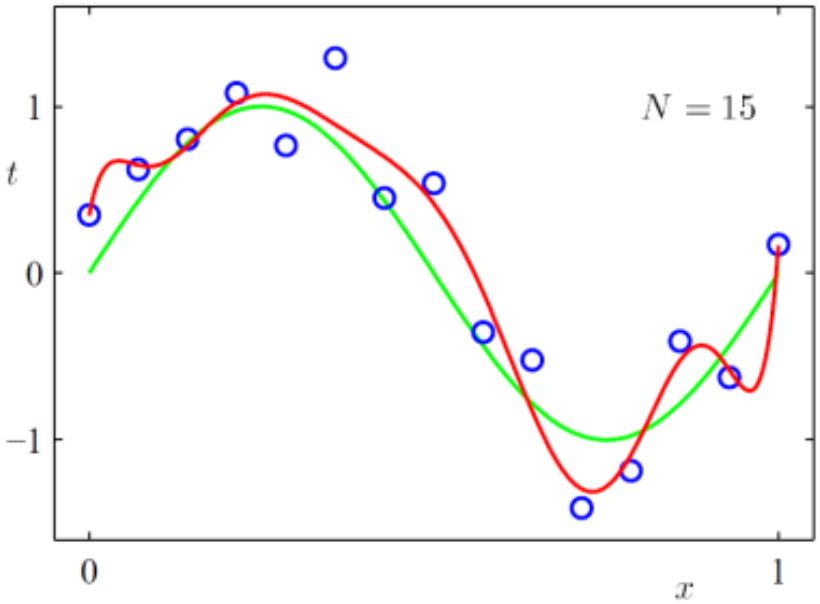
# Observations

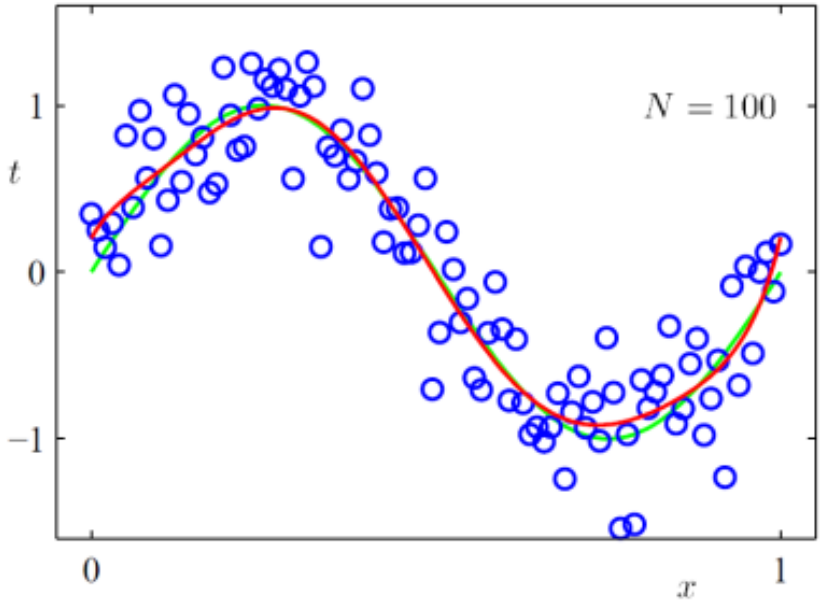
- Degree 3 is the best match to the source
- Degree 9 is the best match to the samples
- We call this **over-fitting**
- Performance on test data:



# What went wrong?

- Is the problem a bad choice of polynomial?
- Is the problem that we don't have enough data?
- Answer: Yes



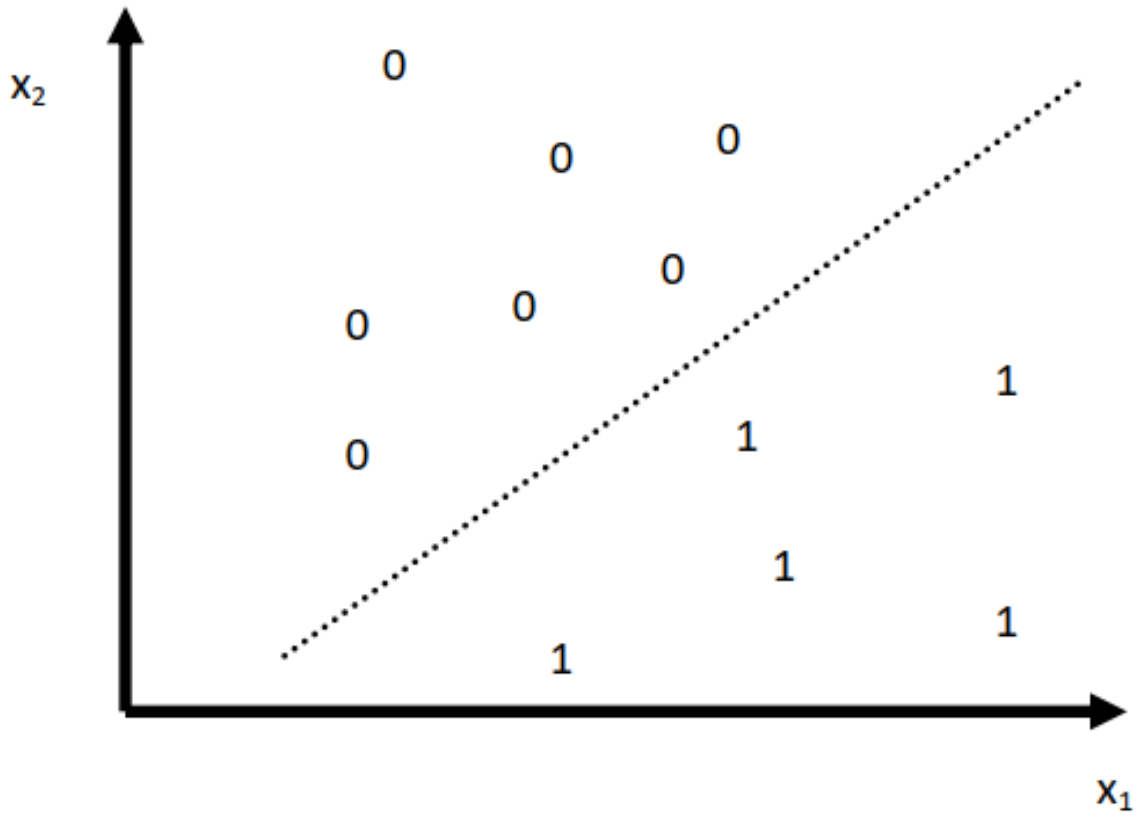
# Classification vs. Regression

- Regression tries to hit the target values with the function we are fitting

- Classification tries to find a function that separates the classes

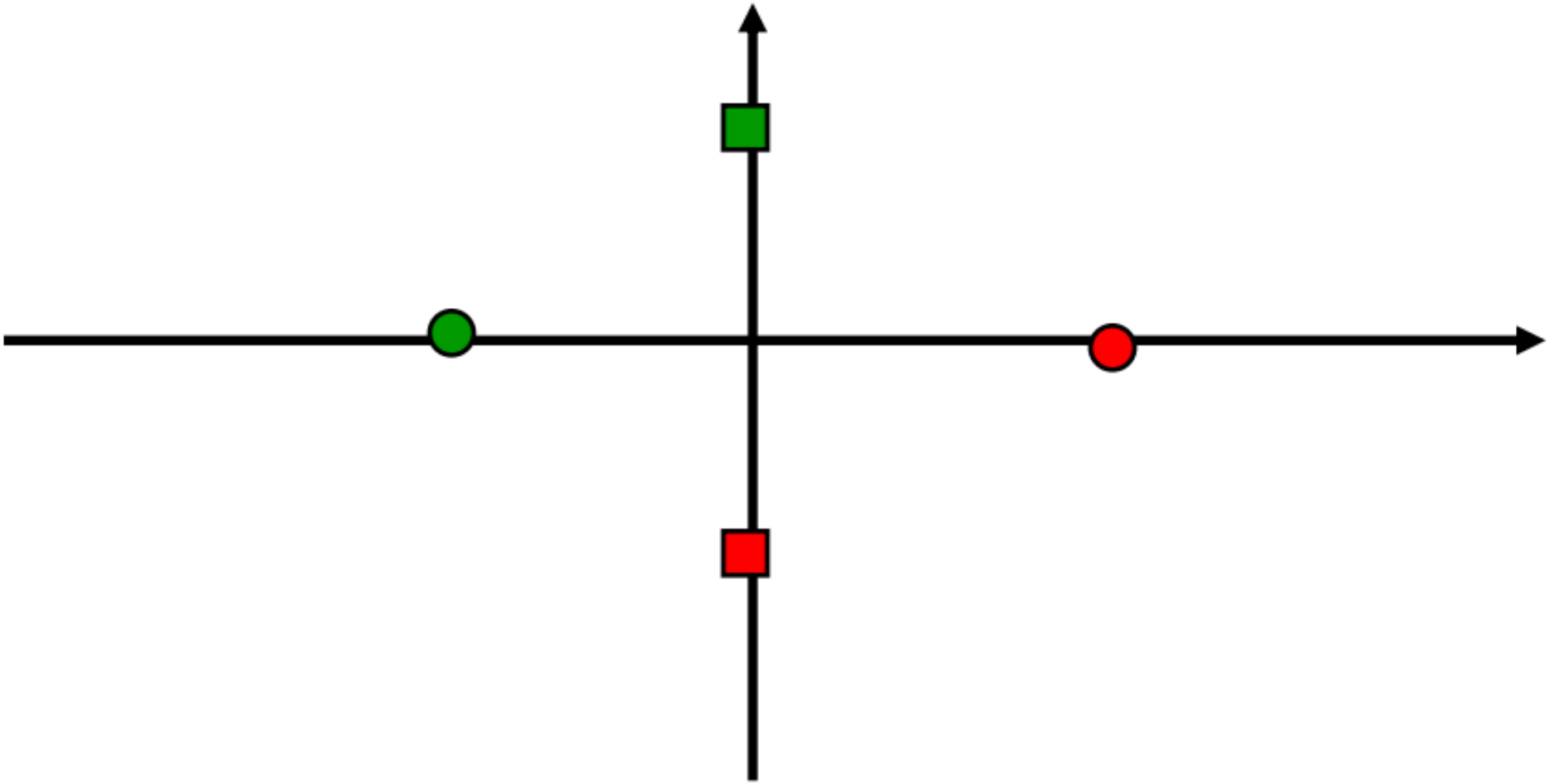
# Decision Boundaries

- A classifier can be viewed as partitioning the input space or feature space X into decision regions




- A linear threshold unit always produces a linear decision boundary. A set of points that can be separated by a linear decision boundary is **linearly separable**.

# Limitations of Linearly Separable Functions



Is red linearly separable from green?

Are the circles linearly separable from the squares?

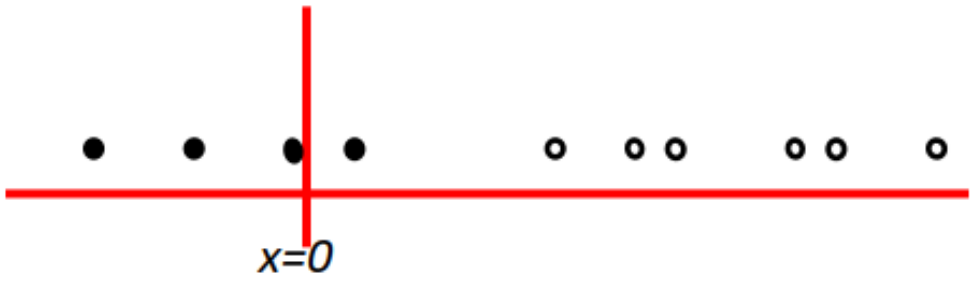
# Feature Engineering

- All data are represented in “feature space”- the space spanned by all possible values of all features
- Feature space is largely a choice, like the degree of your polynomial

- If you don’t like your performance, you can change your feature space – but don’t forget peril of overfitting

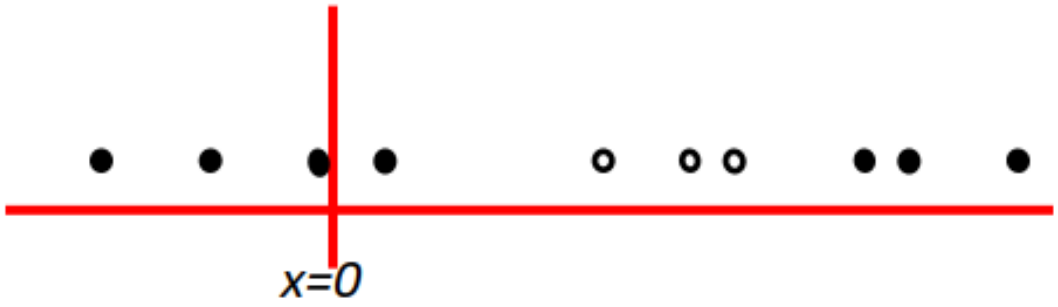
# Suppose we're in 1-dimension

Easy to find a
linear separator

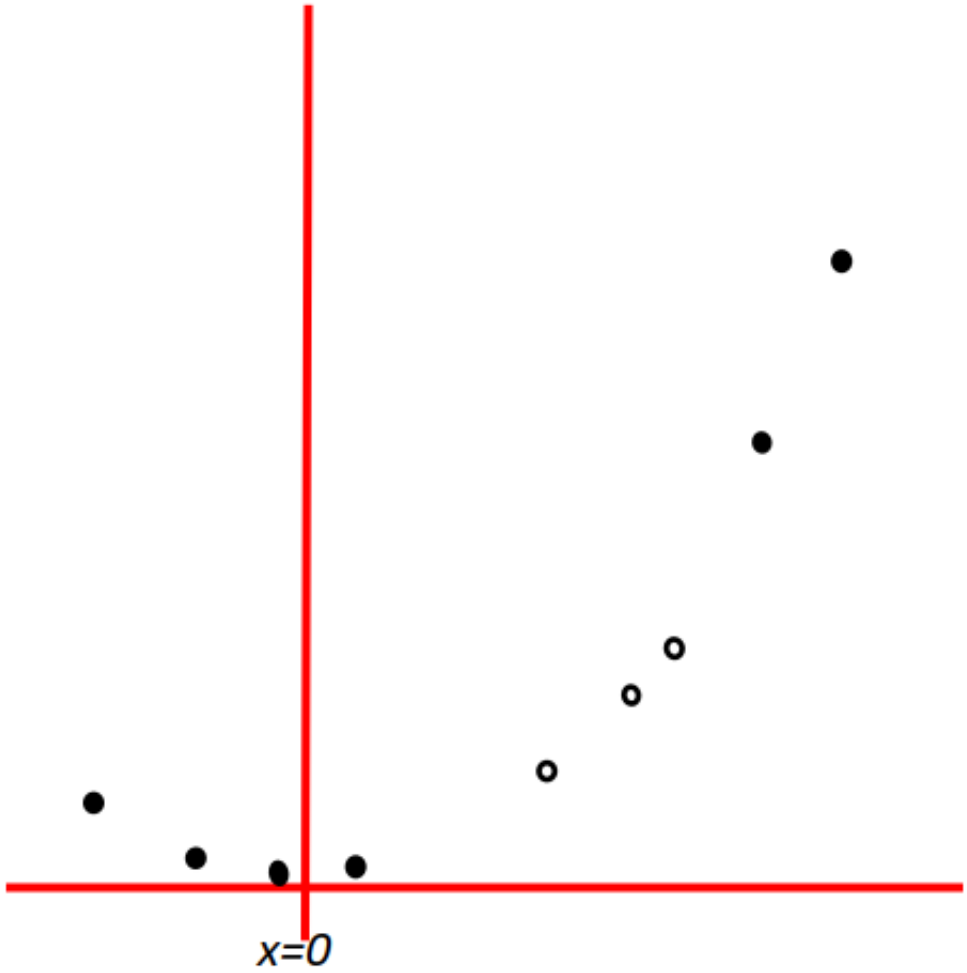


# Harder 1-dimensional dataset

What can be done about this?



# Harder 1-dimensional dataset

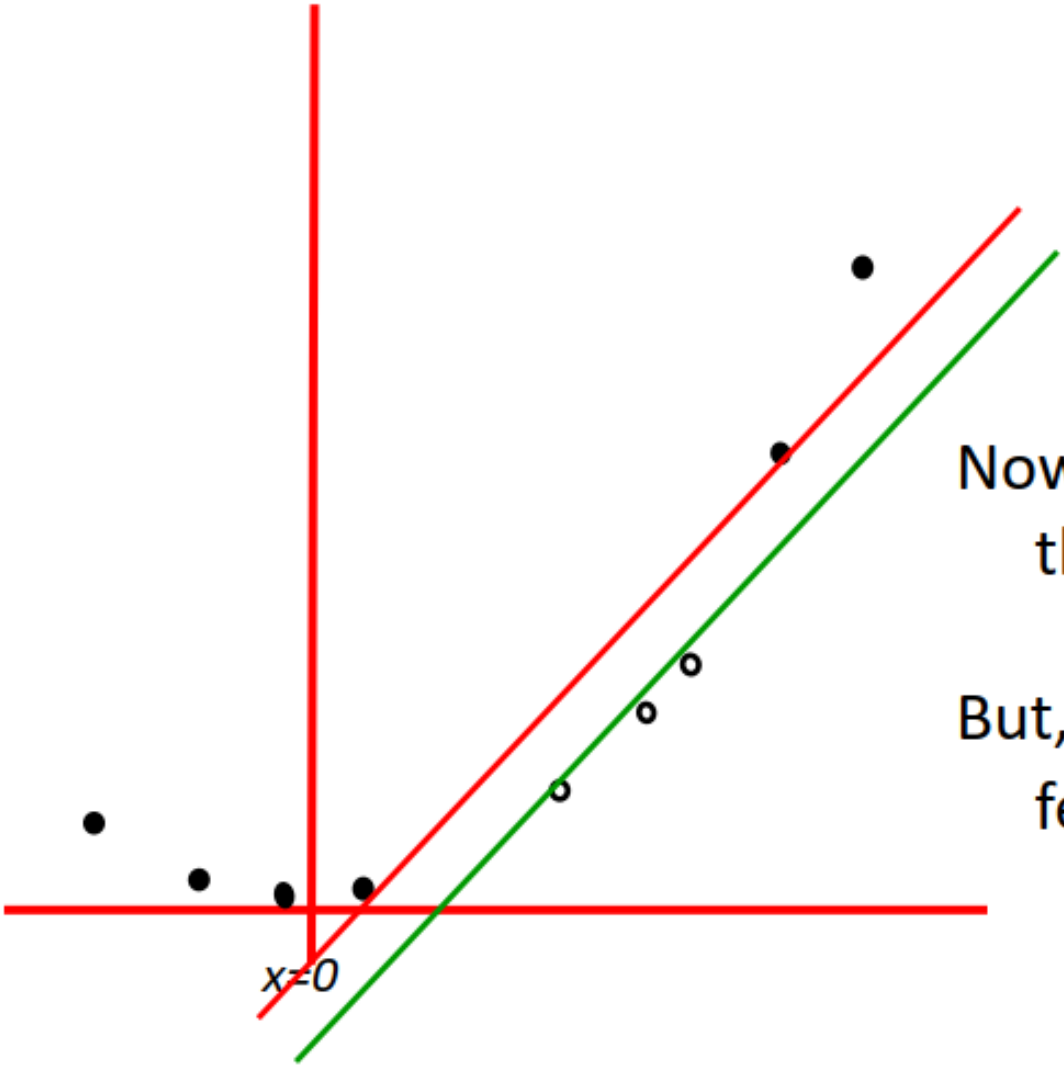



Remember how permitting non-linear basis functions made linear regression so much nicer?

Let's permit them here too

$$\Phi = (x, x^2)$$

# Harder 1-dimensional dataset




Now linearly separable in the new feature space

But, what if the right feature set isn't obvious

$$\Phi = (x, x^2)$$

# Motivation for non-linear Classifiers

- Linear methods are “weak”
  - Make strong assumptions
  - Can only express relatively simple functions of inputs
- Coming up with good features can be hard
  - Requires human input
  - Knowledge of the domain
- Role of neural networks
  - Neural networks started as linear models of single neurons
  - Combining ultimately led to non-linear functions that don’t necessarily need careful feature engineering

# Neural Network Motivation

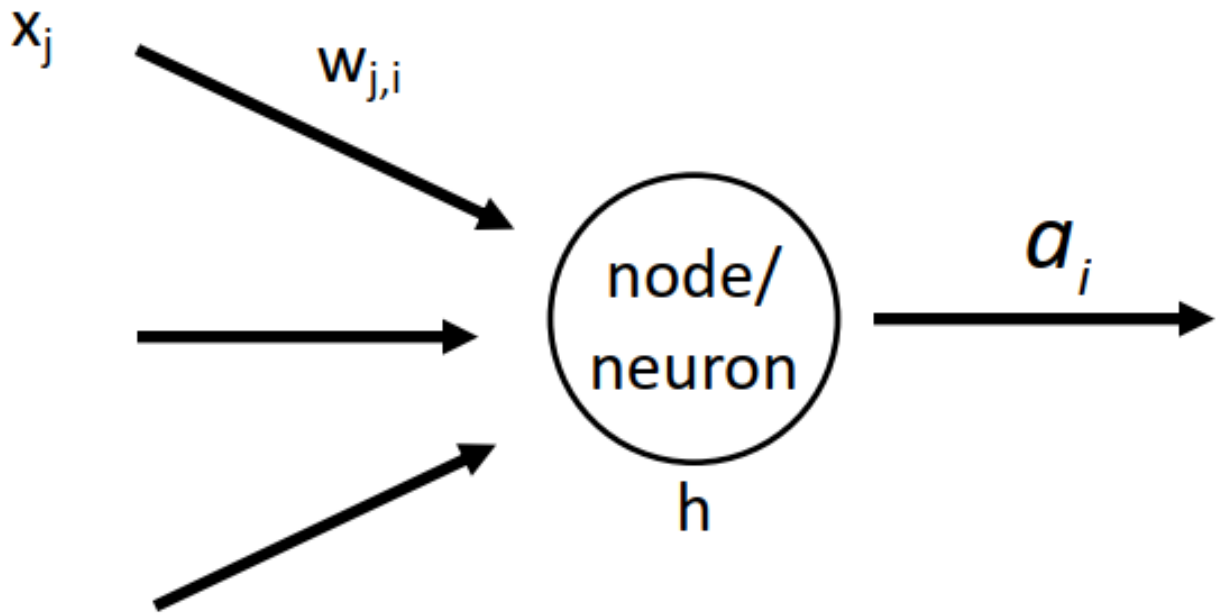
- Human brains are only known example of actual intelligence
- Individual neurons are slow, boring
- Brains succeed by using massive parallelism
- Idea: Copy what works

| | Computers | Brains |
|---|---|---|
| Computational Units | $10^{10}$ transistors/CPU | $10^{11}$ neurons/brain |
| Storage Units | $10^{11}$ bits RAM<br>$10^{13}$ bits HD | $10^{11}$ neurons<br>$10^{14}$ synapses |
| Cycle Time | $10^{-9}$ S | $10^{-3}$ S |
| Bandwidth | $10^{10}$ bits/s* | $10^{14}$ bits/s |
| Compute Power | $10^{10}$ Ops/s | $10^{14}$ Ops/s |

# Neural Network Lore

- Neural nets have been adopted with an almost religious fervor within the AI community – several times
  - First coming: Perceptron
  - Second coming: Multilayer networks
  - Third coming (present): Deep networks

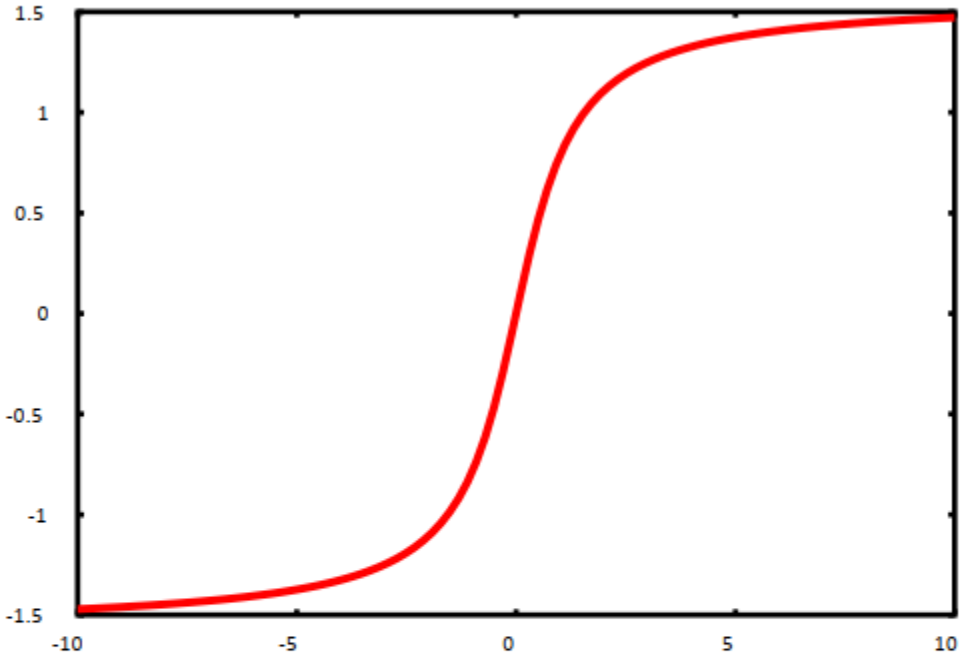
# Artificial Neurons




$$a_i = h\left(\sum_j w_{j,i} x_j\right)$$

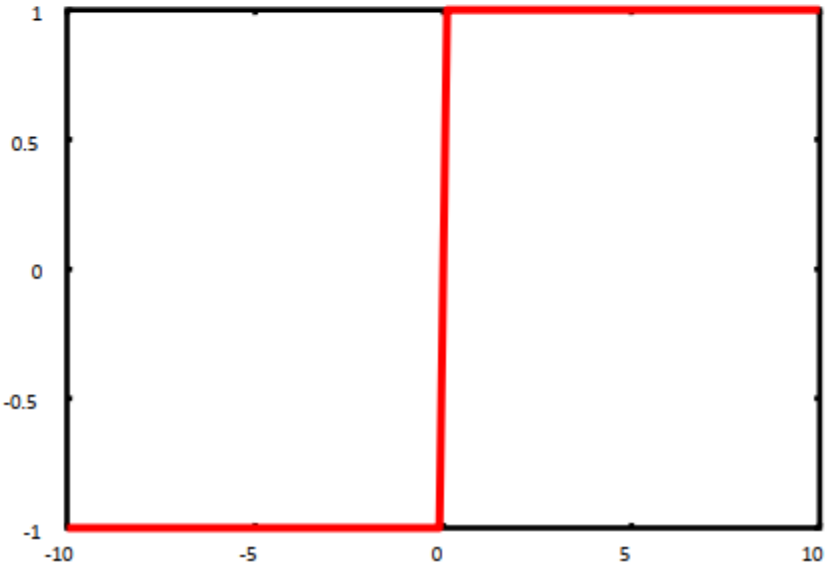
h can be any function, but usually a smoothed step function

# Threshold Functions

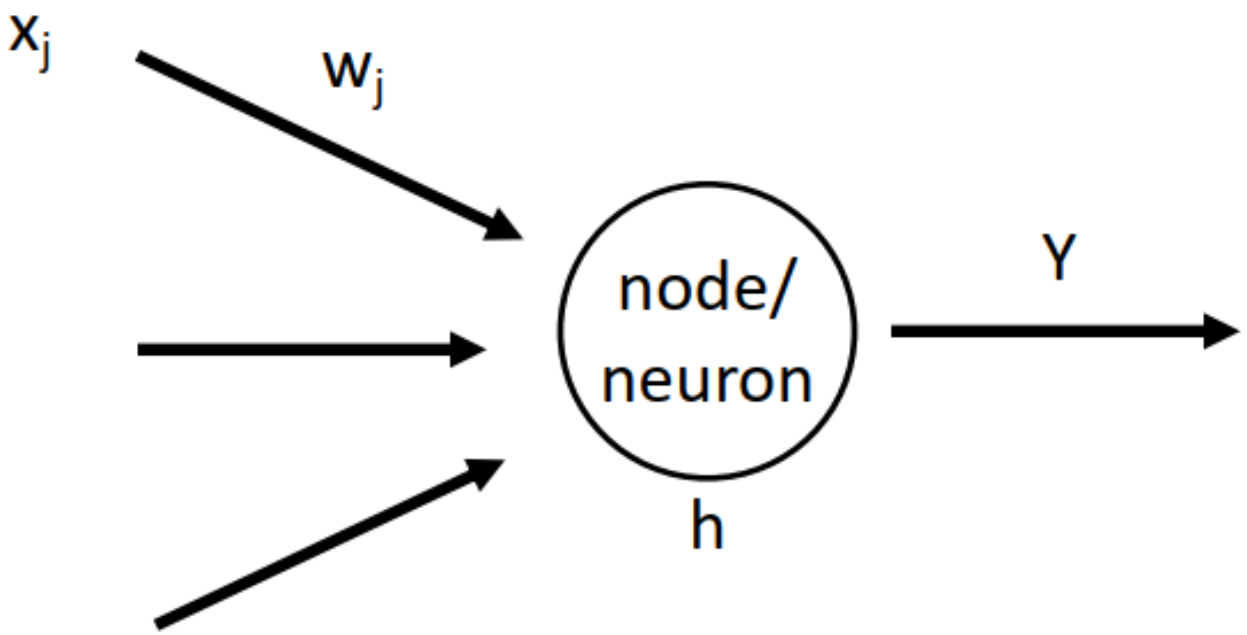



h(x)=tanh(x) or 1/(1+exp(-x))
(logistic sigmoid)

h(x)=sgn(x)
(perceptron)

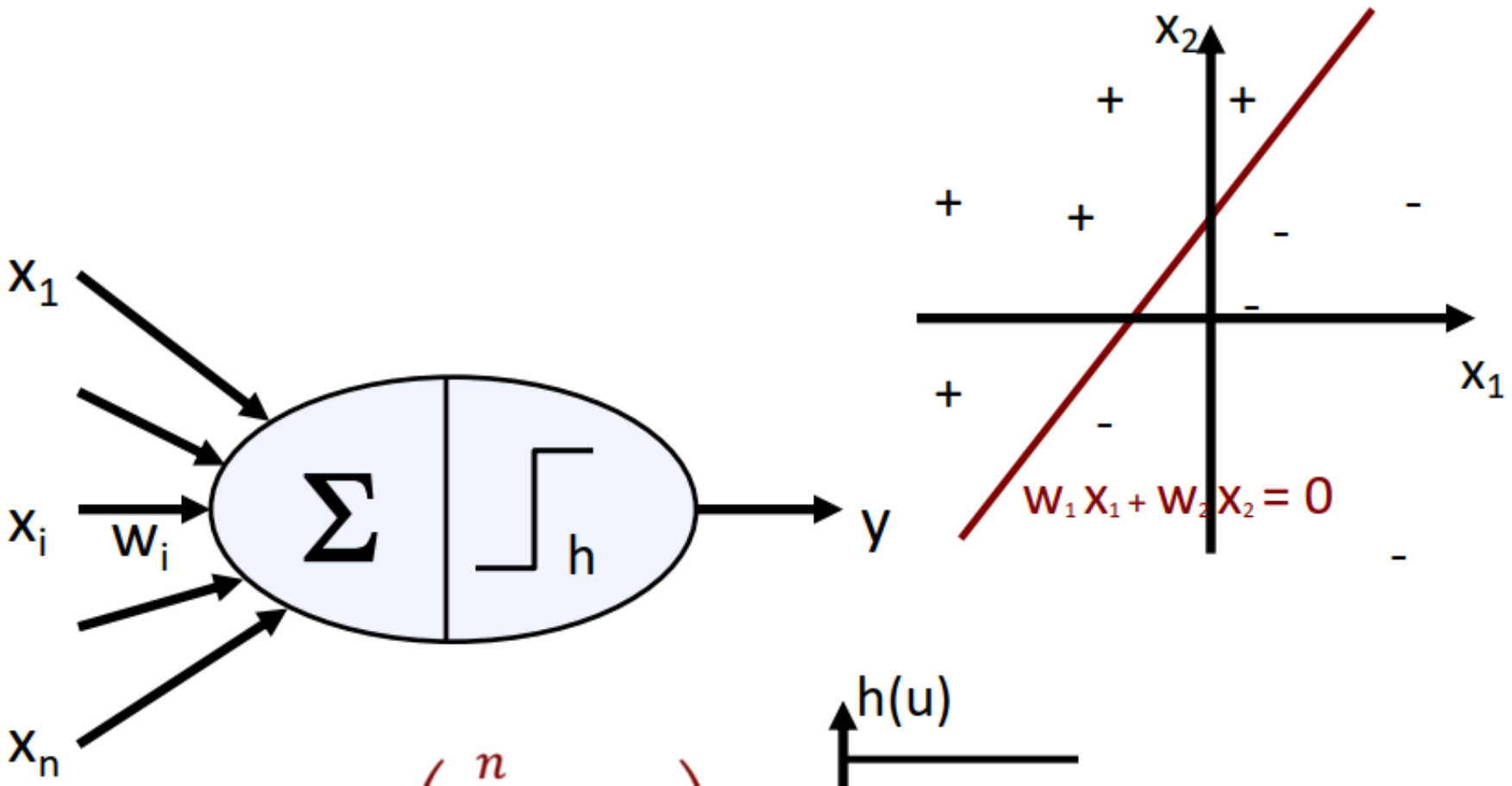


# Special Case: Perceptron

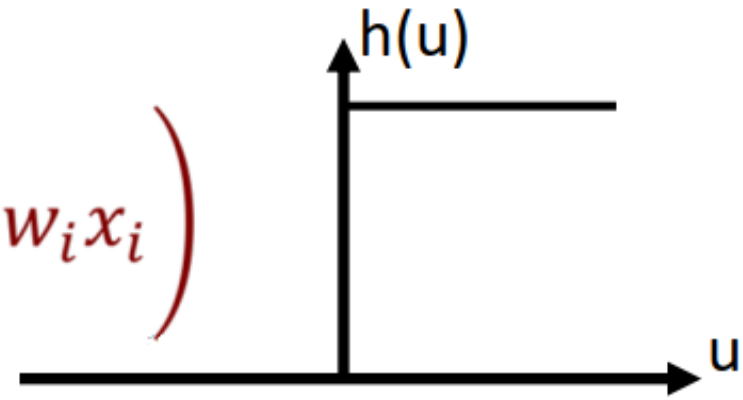



h is a simple step function (sgn)

# Perceptron is a Linear Classifier




$$y = f(\boldsymbol{x}, \boldsymbol{w}) = h\left(\sum_{i=1}^{n} w_i x_i\right)$$

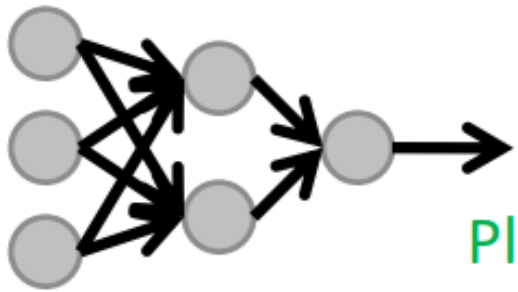


# Good News/Bad News

- Good news
  - *Perceptron learning rule* can learn to distinguish any two classes that are linearly separable
  - ***If*** classes are separable, perceptron learning rule will converge for any learning rate
- Bad news
  - Linear separability is a strong assumption
  - Failure to appreciate this led to excessive optimism and first neural network crash

# Multilayer Networks

- Once people realized how simple perceptrons were, they lost interest in neural networks for a while
- Multilayer networks turn out to be much more expressive (with a smoothed step function)
  - Use sigmoid, e.g., $h=\tanh(w^T x)$ or logistic sigmoid
  - With 2 layers, can represent any continuous function
  - With 3 layers, can represent many discontinuous functions
- Tricky part: How to adjust the weights



Play with it at: http://playground.tensorflow.org

# NN History Through the Second Coming

- Second wave of interest in neural networks lost research momentum in the 1990s – though still continued to enjoy many practical applications
- Neural network tricks were not sufficient to overcome competing methods:
  - Support vector machines
  - Clever feature selection methods wrapped around simple or linear methods
- 2000-2010 was an era of linear + special sauce
- What changed?

# Deep Networks

- Not a learning algorithm, but a family of techniques
  - Improved training techniques (though still essentially gradient descent)
  - Clever crafting of network structure – convolutional nets
  - Some new activation functions

- Exploit massive computational power
  - Parallel computing
  - GPU computing
  - Very large data sets (can reduce overfitting)