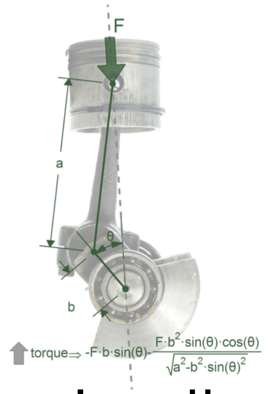
Applied Artificial Intelligence

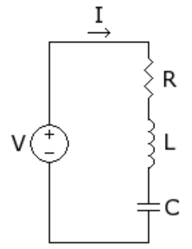
Session 15: Exam Review

Fall 2018
NC State University
Lecturer: Dr. Behnam Kia
Course Website: https://appliedai.wordpress.ncsu.edu/

What is an AI problem?

- If the problem is described by a set of formal mathematical rules (coming from Math, Physics, Chemistry, Biology, etc.), and there are known methods to solve it, develop a conventional computer program and solve it. Usually this is <u>**not**</u> an AI problem – unless it has exponential complexity.



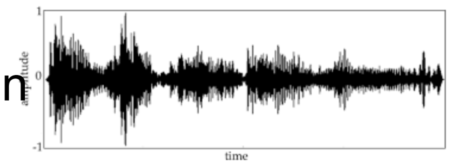


$$RI(t) + L\frac{dI(t)}{dt} + V(0) + \frac{1}{C}\int_0^t I(\tau)\,d\tau = V(t)\,.$$

- If there is no formal mathematical description to the problem, but the problem is easy for humans to solve (intuitively), then this is an AI problem.

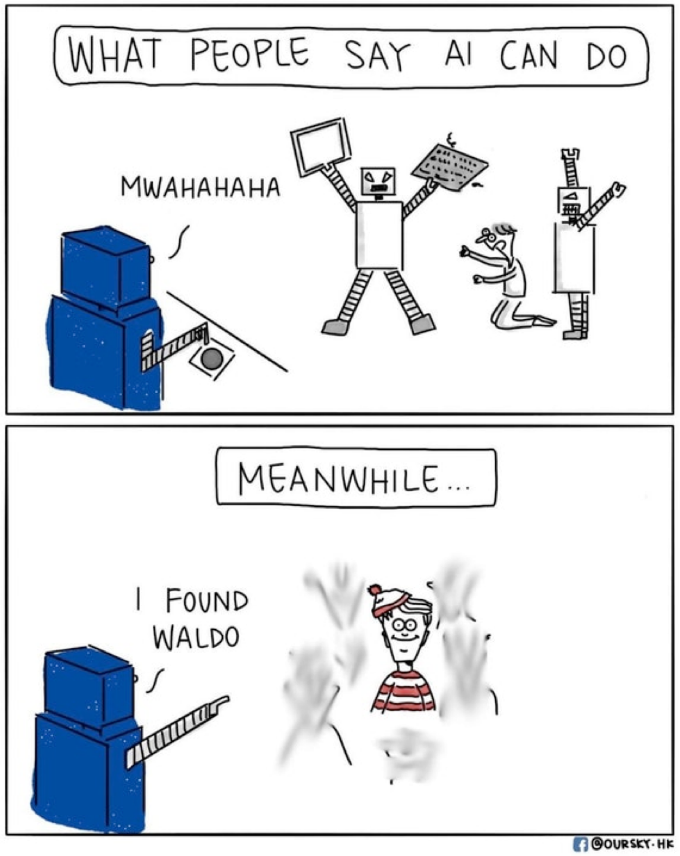
Classification Dog/Cat?

Speech Recognition



What is Artificial Intelligence

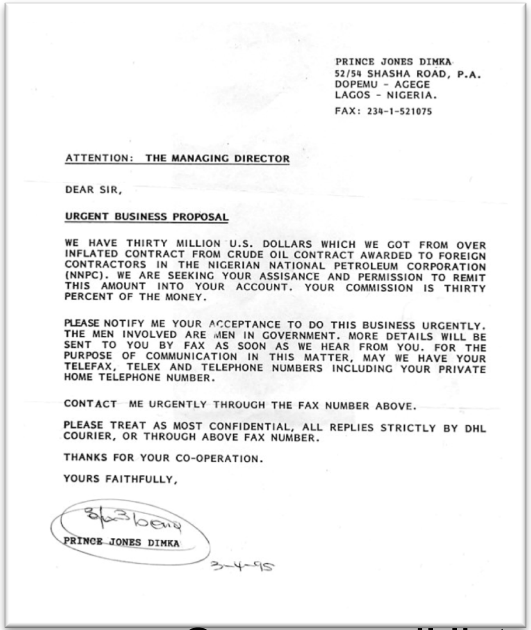
- Artificial intelligence “tries” to mimic human intelligence.
- So far AI methods have been problem-specific, and we have no general-purpose, human-like *General AI* system.
- In this course we focus on these problem-specific AI methods, also known as *Narrow AI*.



New Additions to AI (Machine Learning) Problems

- Extracting knowledge from Big Data
  - And usually there is no formal, solution.

- Solving dynamic and varying problems.
  - The static version of the problem may or may not have a formal solution. But when the problem changes, so should the solution. This is not trivial.



PRINCE JONES DIMKA
52/54 SHASHA ROAD, P.A.
DOPEMU - AGEGE
LAGOS - NIGERIA.

FAX: 234-1-521075

ATTENTION: THE MANAGING DIRECTOR

DEAR SIR,

URGENT BUSINESS PROPOSAL

WE HAVE THIRTY MILLION U.S. DOLLARS WHICH WE GOT FROM OVER INFLATED CONTRACT FROM CRUDE OIL CONTRACT AWARDED TO FOREIGN CONTRACTORS IN THE NIGERIAN NATIONAL PETROLEUM CORPORATION (NNPC). WE ARE SEEKING YOUR ASSISANCE AND PERMISSION TO REMIT THIS AMOUNT INTO YOUR ACCOUNT. YOUR COMMISSION IS THIRTY PERCENT OF THE MONEY.

PLEASE NOTIFY ME YOUR ACCEPTANCE TO DO THIS BUSINESS URGENTLY. THE MEN INVOLVED ARE MEN IN GOVERNMENT. MORE DETAILS WILL BE SENT TO YOU BY FAX AS SOON AS WE HEAR FROM YOU. FOR THE PURPOSE OF COMMUNICATION IN THIS MATTER, MAY WE HAVE YOUR TELEFAX, TELEX AND TELEPHONE NUMBERS INCLUDING YOUR PRIVATE HOME TELEPHONE NUMBER.

CONTACT ME URGENTLY THROUGH THE FAX NUMBER ABOVE.

PLEASE TREAT AS MOST CONFIDENTIAL, ALL REPLIES STRICTLY BY DHL COURIER, OR THROUGH ABOVE FAX NUMBER.

THANKS FOR YOUR CO-OPERATION.

YOURS FAITHFULLY,

[illegible]

PRINCE JONES DIMKA

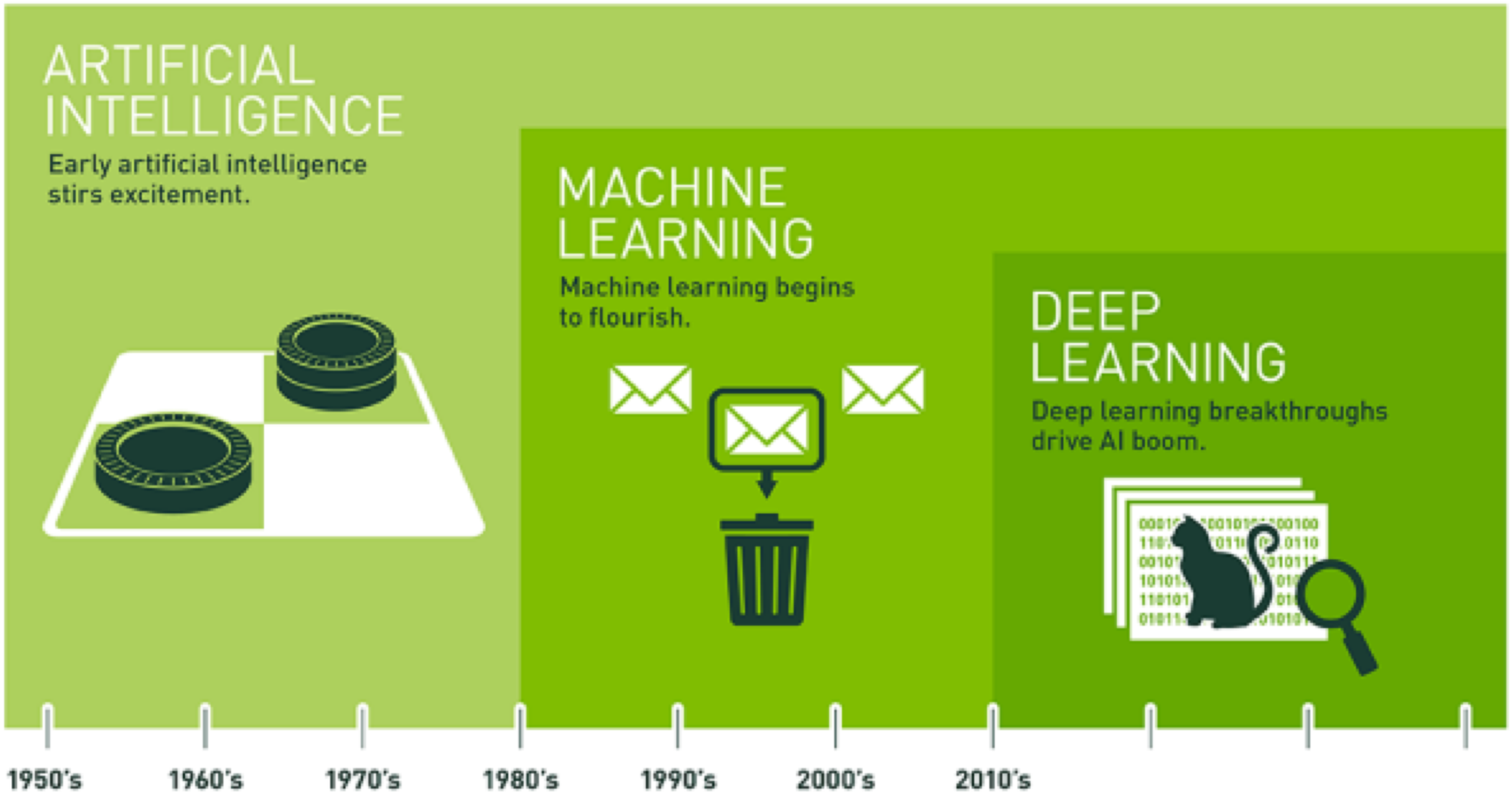
3-4-95

Spam email list

AI Problems

- Problems simple for humans, but not for computers.
- Extracting knowledge from Big Data.
- Solving dynamic, varying problems.

Artificial Intelligence


ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

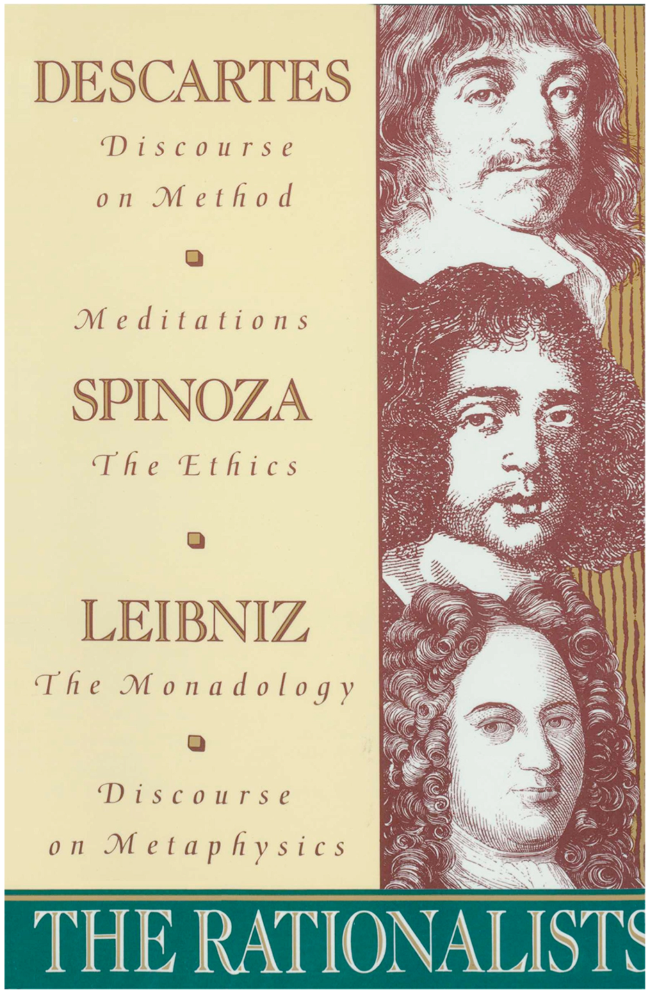
Picture from NVIDIA's deep learning institute

What you can expect from this course?

At the end of the semester you will:

- know what an AI problem is and what is not!
- learn the basic foundations of deep learning and how to apply it to AI problems.
- gain basic hands-on experience with AI development tools and software.
- get enough experience, knowledge, and confidence to pursue on your own and learn more advanced topics.

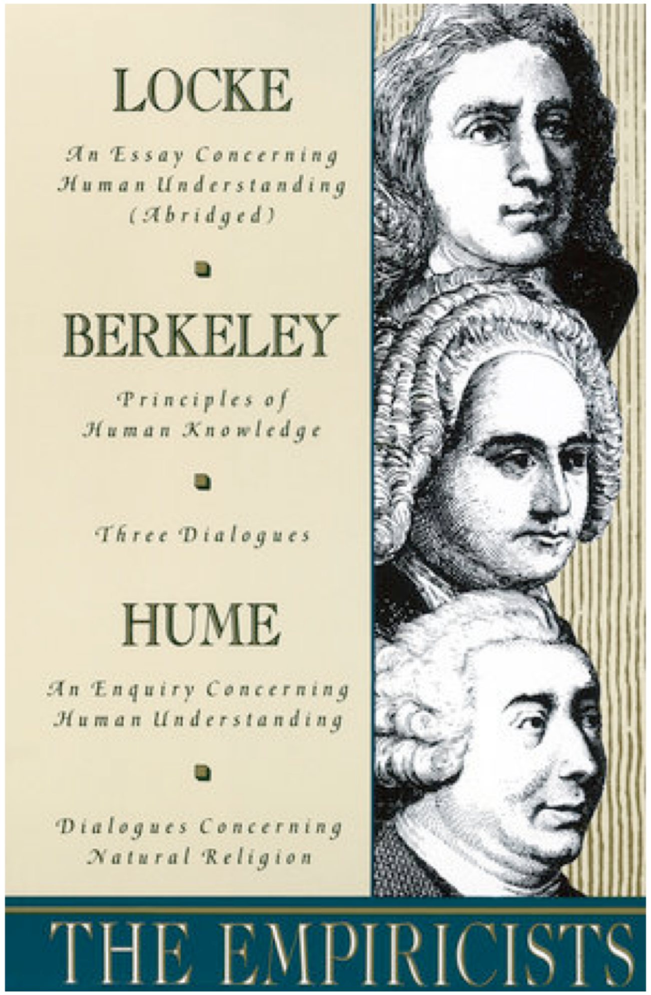
Rationalism



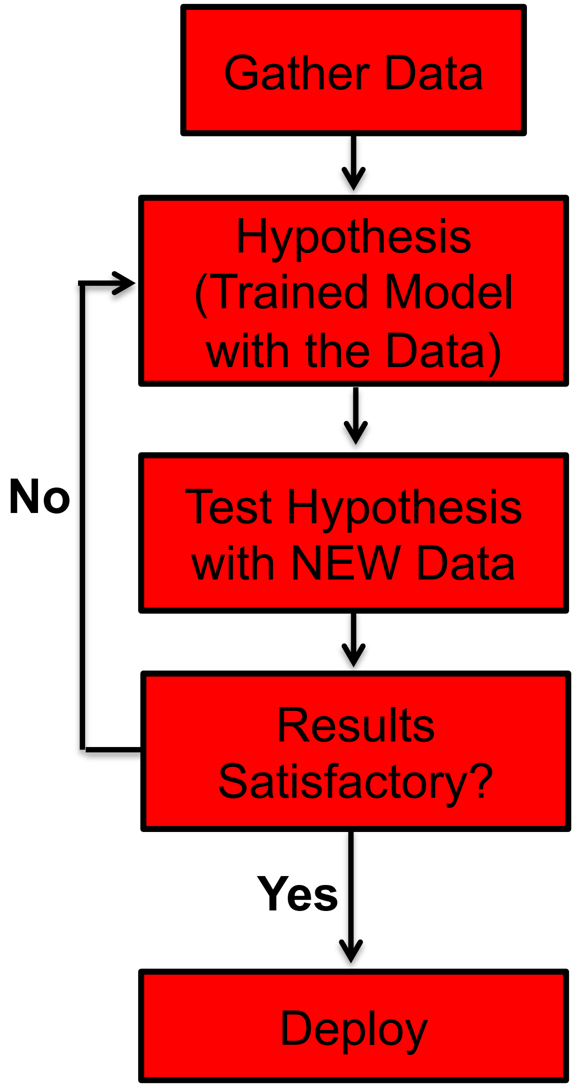
- Mind is a reasoning machine.
- It is equipped with knowledge, and with a reasoning engine it deduces new knowledge or solutions.
- So to create AI we need:
  - Knowledge representation.
  - A reasoning engine.

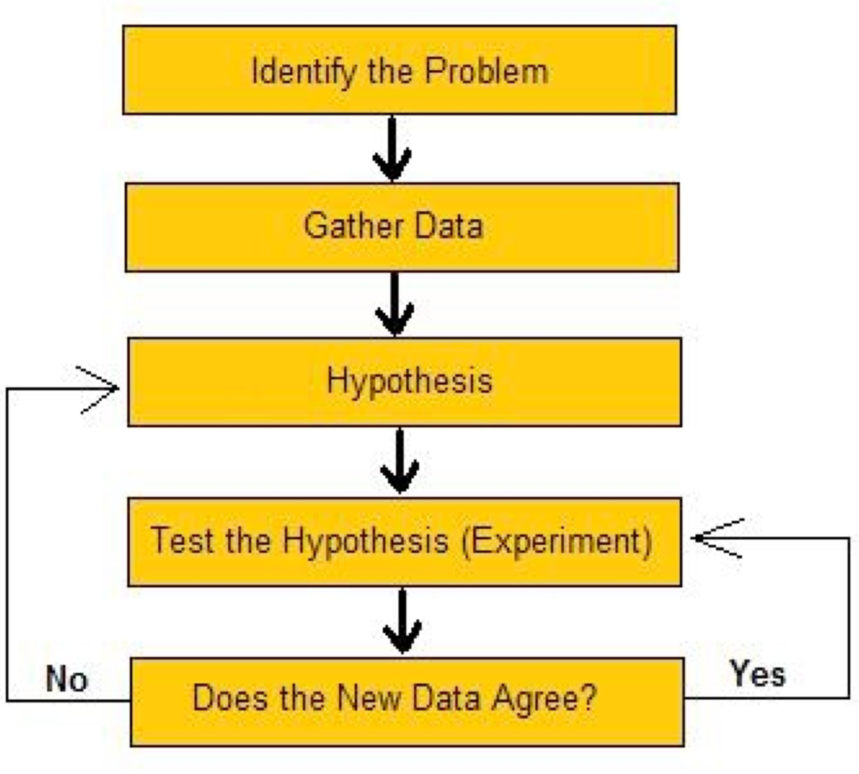
The Empiricists

- Mind is a learning machine!
- Empiricism emphasizes the role of experience, discounts the value of a priori reasoning.
- So to create AI we need:
  - Learning algorithms.
  - A lot of data.


LOCKE
An Essay Concerning Human Understanding (Abridged)
BERKELEY
Principles of Human Knowledge
Three Dialogues
HUME
An Enquiry Concerning Human Understanding
Dialogues Concerning Natural Religion
THE EMPIRICISTS

Machine Learning Flowchart (Which Follows Scientific Method)


Gather Data
Hypothesis (Trained Model with the Data)
Test Hypothesis with NEW Data
Results Satisfactory?
No
Yes
Deploy

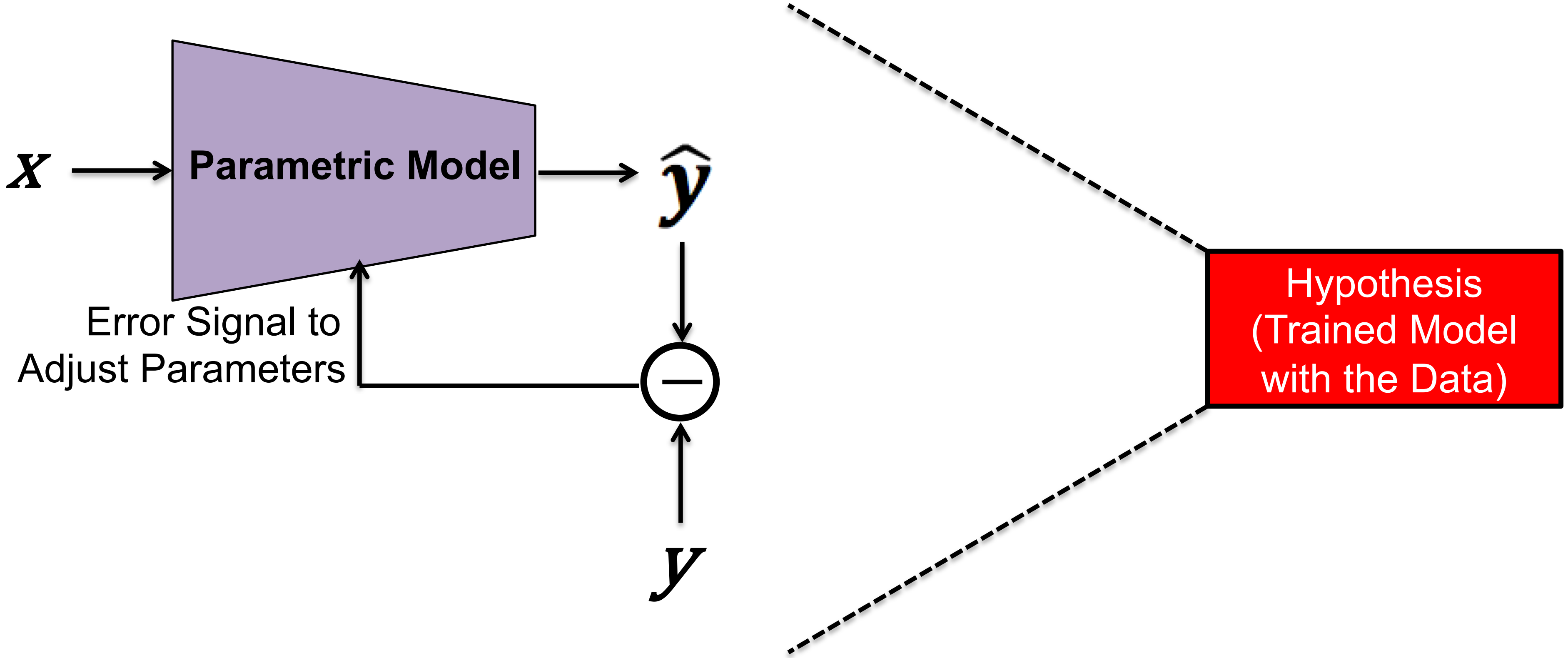


Scientific Method

Machine Learning: Training Model with Data

Training Data: (x_i, y_i), i=1,2,3,...,N

x is called feature, y is label.


x
Parametric Model
$\hat{y}$
$-$
y
Error Signal to Adjust Parameters
Hypothesis (Trained Model with the Data)

How to Adjust Parameters?

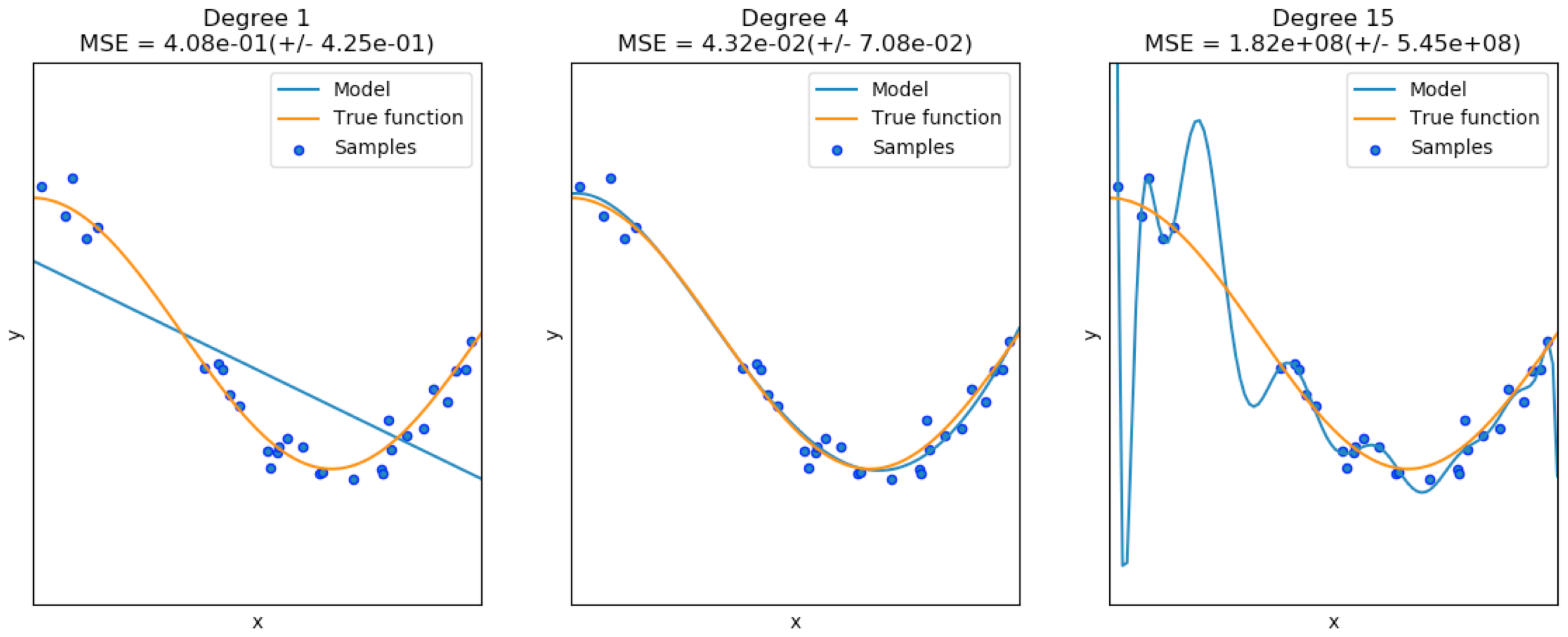
- This is an optimization problem to find parameter values, *P**, that minimizes the error.

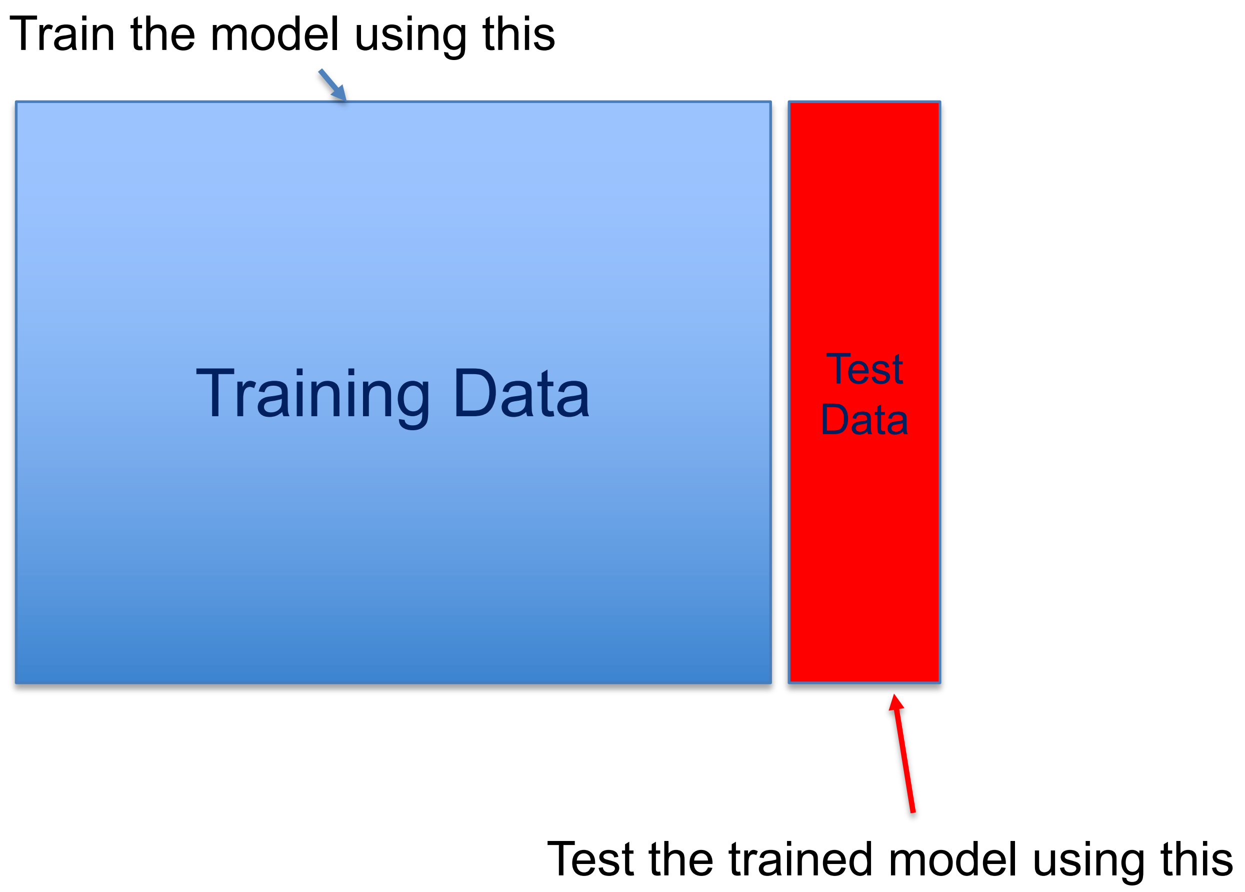
$$P^* = \operatorname{argmin}_P \sum_{Training\,set} Error(x, y, \hat{y})$$

Another common way to say this is shown below, where J is cost function

$$P^* = \operatorname{argmin}_P J(P)$$

Underfitting and Overfitting




Train the model using this
Training Data
Test Data
Test the trained model using this

Parametric Models:

- Linear Regression
- SVM
- Naïve Bayes
- Feedforward multilayer networks
- Convolutional Network
- Recurrent Neural Network
  - LSTM
  - GRU

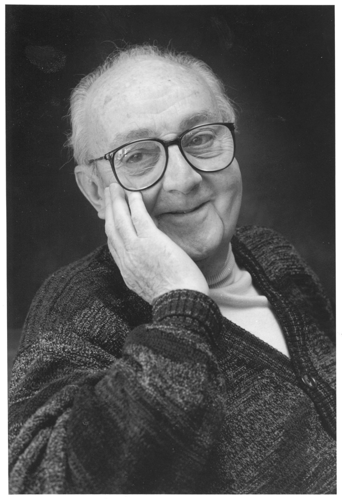
.

.

.

On the Limitation of the Models

- "All models are wrong but some are useful[1]"



[1]*Box, G. E. P. (1979), "Robustness in the strategy of scientific model building", in Launer, R. L.; Wilkinson, G. N., Robustness in Statistics, Academic Press, pp. 201–236.*

On the Limitation of the Models

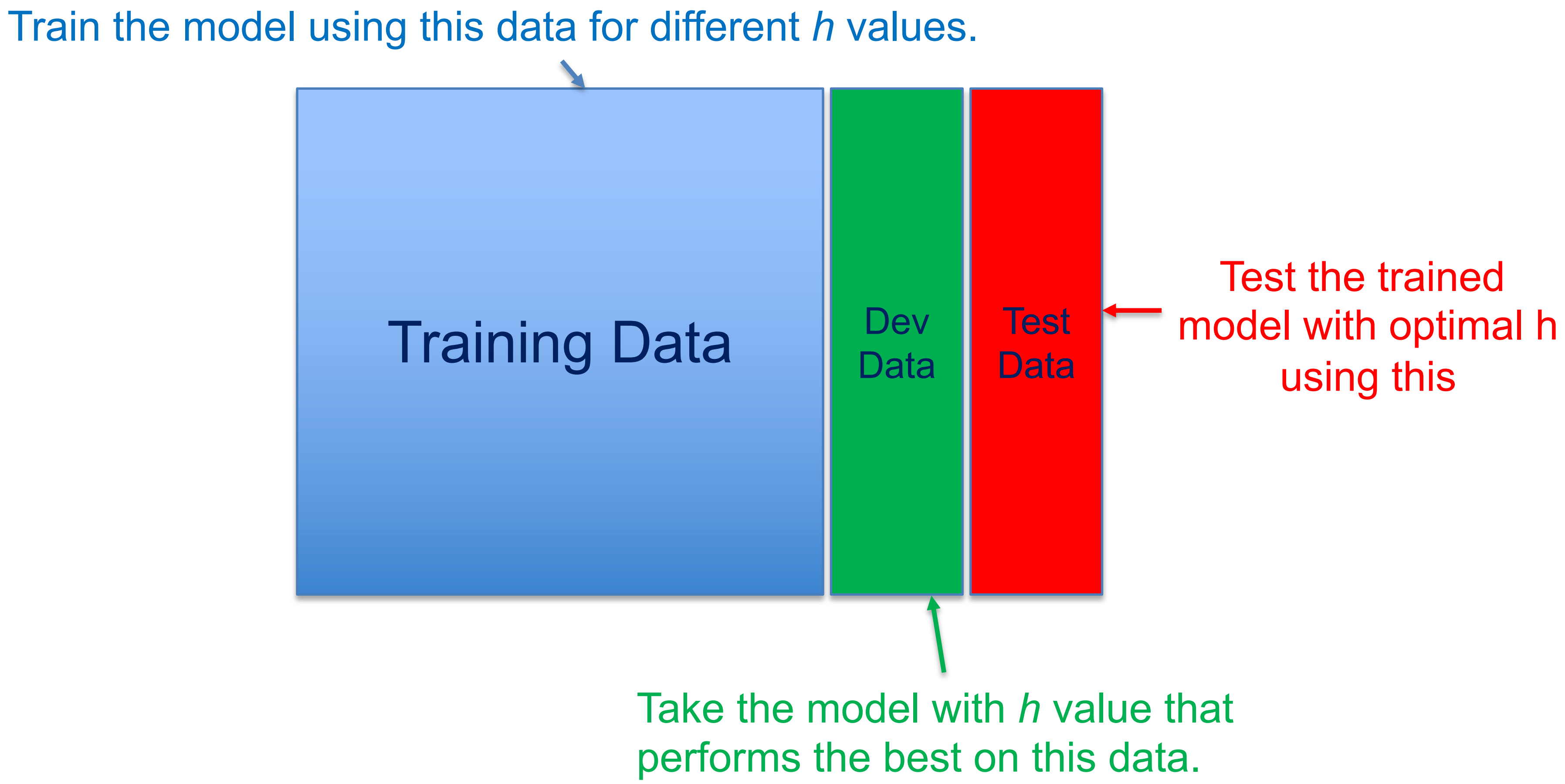
- “Models have limitations, stupidity does not!”

[1]*Rodriguez, A. (2010) Arizona State University, ”Linear System Theory", course notes*.

On the Limitation of the Models

- "Models have limitations, stupidity does not!"

[1]*Rodriguez, A. (2010) Arizona State University, "Linear System Theory", course notes*.


Train the model using this data for different *h* values.
Training Data
Dev Data
Test Data
Test the trained model with optimal h using this
Take the model with *h* value that performs the best on this data.

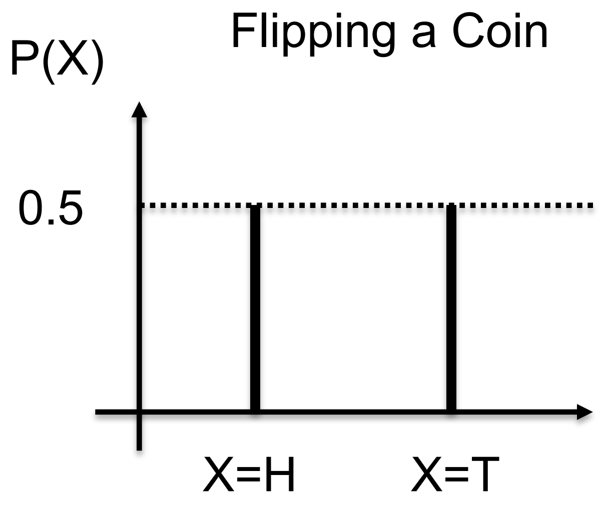
Probability and Statistics

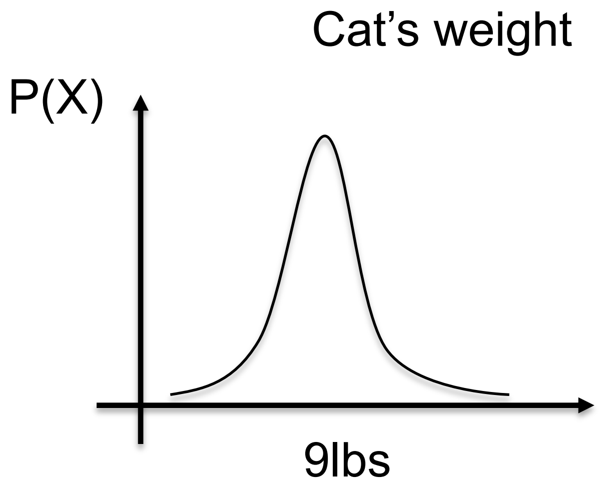
Random Variable

- A random variable is a variable that takes different possible outcomes of a random experiment.
- Random variable can be discrete or continuous.

Probability Distribution Functions

- Probability distribution function is a description of how likely a random variable or a set of variables is to take on each of its possible states.





Conditional Probability

$P(y|x)$

Bayes Rule

Posterior $likelihood$ $Prior$

- $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$

$Evidence\ (normalization\ factor)$

Example

- Imagine you are designing a filter for spam emails. And from training data (you have lots of labeled good emails(nicknamed ham) and lots of spam emails labeled spam). You have derived a probabilistic model P(email text | spam) and another probabilistic model P(email text | ham).

- You receive a new email, *new_email*, that you don't know whether it is spam or a good email. And you have no prior knowledge about the probability of emails being spam or ham. Your probability models say that:

P(new_email| spam)=0.4

P(new_email |ham)=0.6

How your probabilistic system should label and classify this new_email in order to minimize the error rate?

- $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$

P(spam |new_email)=P(new_email |spam)P(spam)/P(new_email)
P(ham |new_email)=P(new_email |ham)P(ham)/P(new_email)

- $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$

P(spam |new_email)=P(new_email |spam)P(spam)/P(new_email)

P(ham |new_email)=P(new_email |ham)P(ham)/P(new_email)

P(spam |new_email)=0.4*P(spam)/P(new_email)

P(ham |new_email)=0.6*P(ham)/P(new_email)

P(spam |new_email)=0.4*0.5/P(new_email) P(spam)=p(spam)=0.5

P(ham |new_email)=0.6*0.5/P(new_email)

- $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$

P(spam |new_email)=P(new_email |spam)P(spam)/P(new_email)

P(ham |new_email)=P(new_email |ham)P(ham)/P(new_email)

P(spam |new_email)=0.4*P(spam)/P(new_email)

P(ham |new_email)=0.6*P(ham)/P(new_email)

P(spam |new_email)=0.4*0.5/P(new_email)

$\boxed{\text{P(ham |new\_email)=0.6*0.5/P(new\_email)}}$

P(spam)=p(spam)=0.5

- $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$

P(spam |new_email)=P(new_email |spam)P(spam)/P(new_email)

P(ham |new_email)=P(new_email |ham)P(ham)/P(new_email)

P(spam |new_email)=0.4*P(spam)/P(new_email)

P(ham |new_email)=0.6*P(ham)/P(new_email)

P(spam |new_email)=0.4*0.5/P(new_email)

$\boxed{\text{P(ham |new\_email)=0.6*0.5/P(new\_email)}}$

P(spam)=p(spam)=0.5

Error Probability

Example

- Imagine you are designing a filter for spam emails. And from training data (you have lots of labeled good emails(nicknamed ham) and lots of spam emails labeled spam). You have derived a probabilistic model P(email text | spam) and another probabilistic model P(email text | ham).

- You receive a new email, *new_email*, that you don't know whether it is spam or a good email. Imagine in that day your email was under attack by spammers and 90% of the email you have received that day were spam, P(spam)=0.9. Your probability models say that:

P(new_email| spam)=0.4

P(new_email |ham)=0.6

How your probabilistic system should label and classify this new_email in order to minimize the error rate?

- $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$

P(spam |new_email)=P(new_email |spam)P(spam)/P(new_email)
P(ham |new_email)=P(new_email |ham)P(ham)/P(new_email)

- $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$

P(spam |new_email)=P(new_email |spam)P(spam)/P(new_email)

P(ham |new_email)=P(new_email |ham)P(ham)/P(new_email)

P(spam |new_email)=0.4*0.9/P(new_email)

P(ham |new_email)=0.6*0.1/P(new_email)

- $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$

P(spam |new_email)=P(new_email |spam)P(spam)/P(new_email)

P(ham |new_email)=P(new_email |ham)P(ham)/P(new_email)

P(spam |new_email)=0.4*0.9/P(new_email)

P(ham |new_email)=0.6*0.1/P(new_email)

$\boxed{\text{P(spam |new\_email)=0.36/P(new\_email)}}$

P(ham |new_email)=0.06/P(new_email) $\leftarrow$ Error Probability

Naïve Bayes Rule

Bayes Rule

$$p(y \mid x_1, x_2, \dots, x_k) = \frac{p(x_1, x_2, \dots, x_k \mid y)\, p(y)}{p(X)}$$

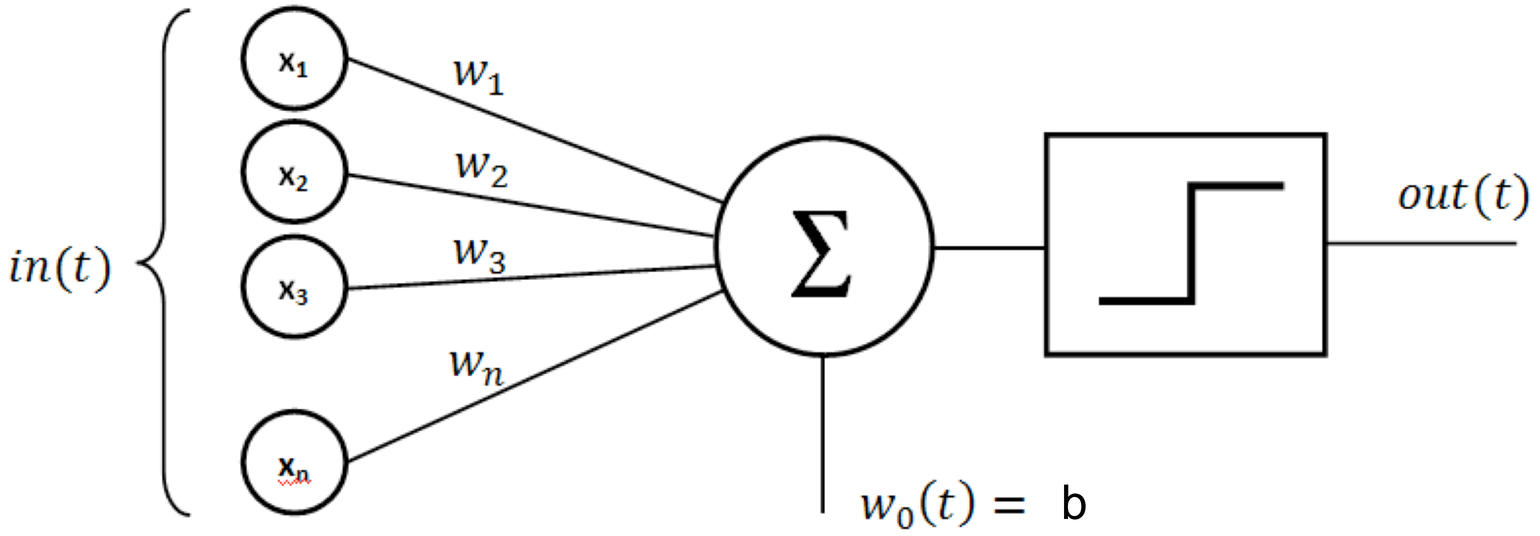
$$\approx \frac{p(x_1 \mid y)\, p(x_2 \mid y) \dots p(x_k \mid y)\, p(y)}{p(X)}$$

Naïve Bayes Rule

We have assumed that features x_1, x_2,…, and x_k are conditionally independent given y

$$p(x_1, x_2, \dots, x_k \mid y) \approx p(x_1 \mid y)\, p(x_2 \mid y) \dots p(x_k \mid y)$$

Linear Algebra



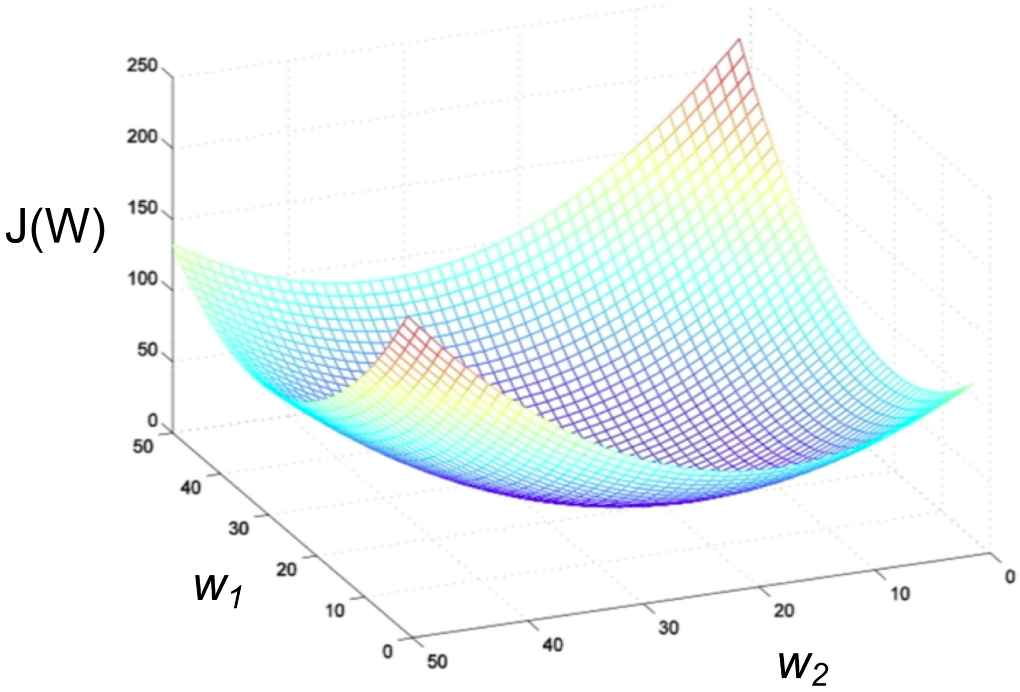
$$output = f(X_p) = \begin{cases} 1 \; if \; W^T.X_p > 0 \\ 0 \; if \; otherwise \end{cases}$$

Vectorized Gradient Descent in 2D

$$W_{next} = W_{current} - \alpha \nabla_W J(W)$$

$$\frac{\partial J(W)}{\partial w_j} = \frac{1}{m} \sum_i x_{pj}^i (x_p^i . W - y^i)$$

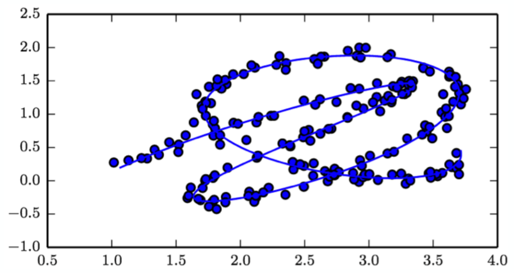
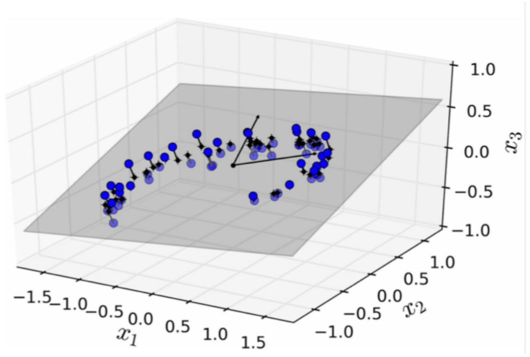
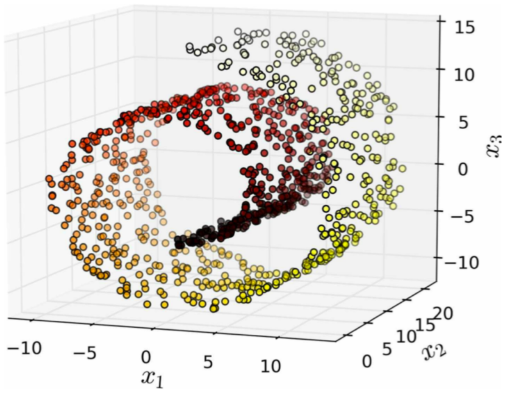
$$\nabla_W J(W) = \frac{1}{m} X_p^T . (X_p \, . W - y)$$



Manifold Assumption (Manifold Hypothesis)

- Manifold Assumption: Real-world high-dimensional data sets lie close to a much lower-dimensional manifold.

- Manifold is a connected subset of a higher dimensional space. (rough definition)

Dimension Reduction

- Linear Algebra provides many techniques, including Principal Component Analysis, for dimension reduction.

- Many of these techniques are implemented in Python Modules (e.g. in scikit-learn).

Principal Component Analysis (PCA)

- PCA is the most common dimension reduction algorithm.
- There are many other dimension reduction algorithms as well.

We have Regression Problems, and we have Classification Problems.

Linear Models:

- Linear Regression
- Logistic Regression

Linear Regression Model: House Prices

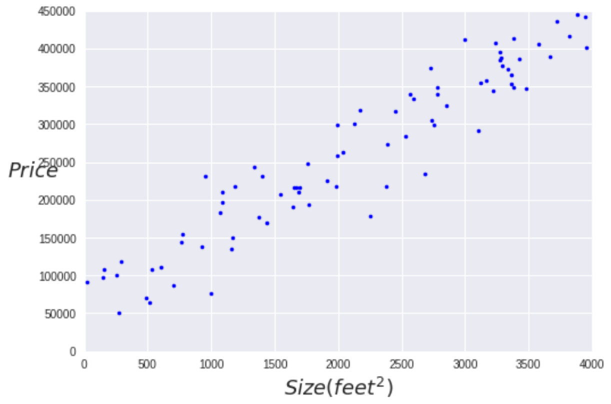
Input: x (the size of house)
Target: y (the price of the house), y$\in R$
Linear Model: $\hat{y} = w_0 x_0 + w_1 x_1$

$$\hat{y} = X.W$$

where:

$$x_p = [1, x_1]$$

$$W = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$



House Prices Mean Squared Error as Cost Function

(x ft², $ y K)

```
(3883 ft²,$432K)
(1668 ft²,$218K)
(3577 ft²,$366K)
(765 ft²,$123K)
(3822 ft²,$493K)
```

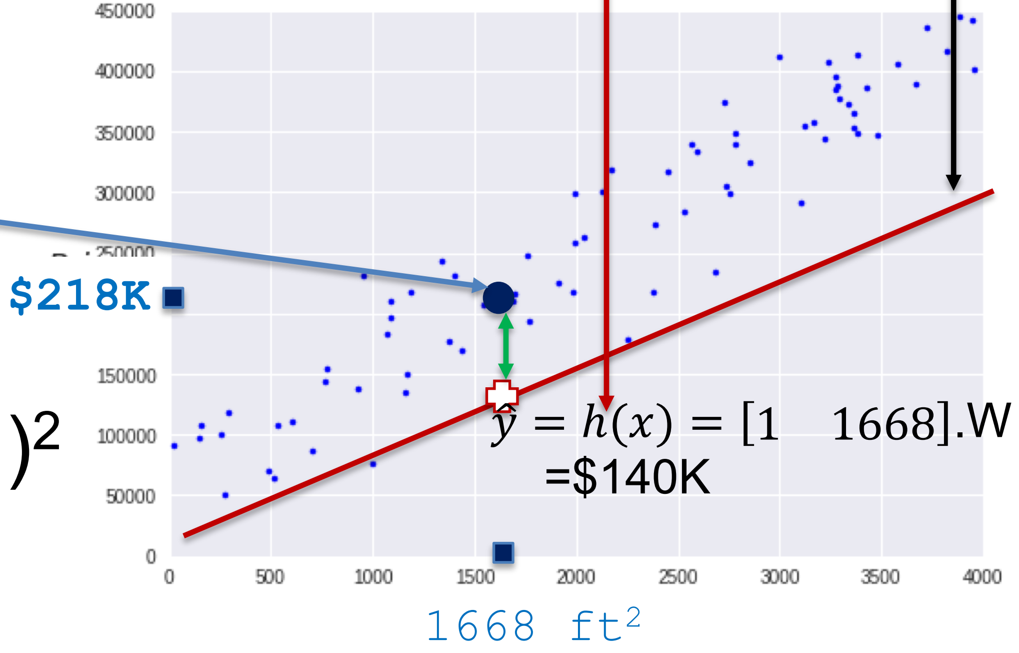
(1668 ft², $218K)

Cost Function for entire training set:

$$J(W)=\frac{1}{m}\sum_i \left(h(\hat{x}^i) - y^i\right)^2$$

Size of training data

$$W = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$?



The Problems of the Analytical Method

$$W_{optimal} = \underset{W}{\text{argmin}}\ J(W)$$

$$J(W) = \frac{1}{2n}\sum_i (W^T . x_p^i - y^i)^2$$

$$\frac{\partial J(W)}{\partial W} = 0 \Rightarrow W_{optimal} = (X_p^T . X_p)^{-1} . X_p^T . y$$

(n,1+1)

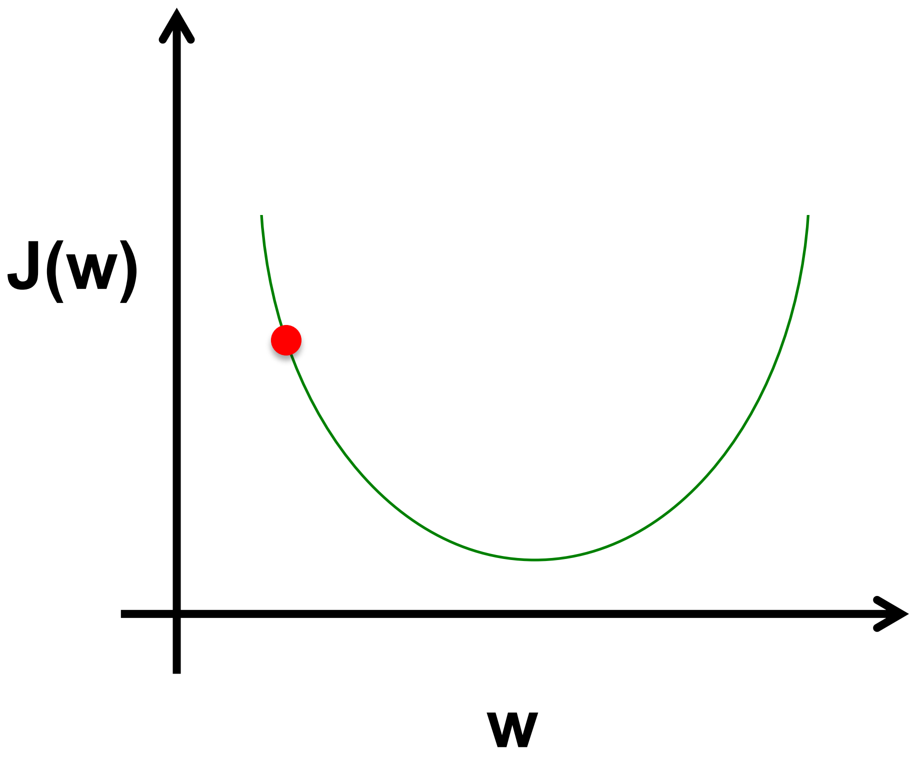
$$X_p = \begin{array}{c} \begin{bmatrix} 1 & 3883 \\ 1 & 1668 \\ 1 & 3577 \\ \vdots & \vdots \\ 1 & 1668 \end{bmatrix} \end{array} \begin{array}{c} x_p^1 \\ \\ \\ \\ x_p^n \end{array}$$

(n,1)

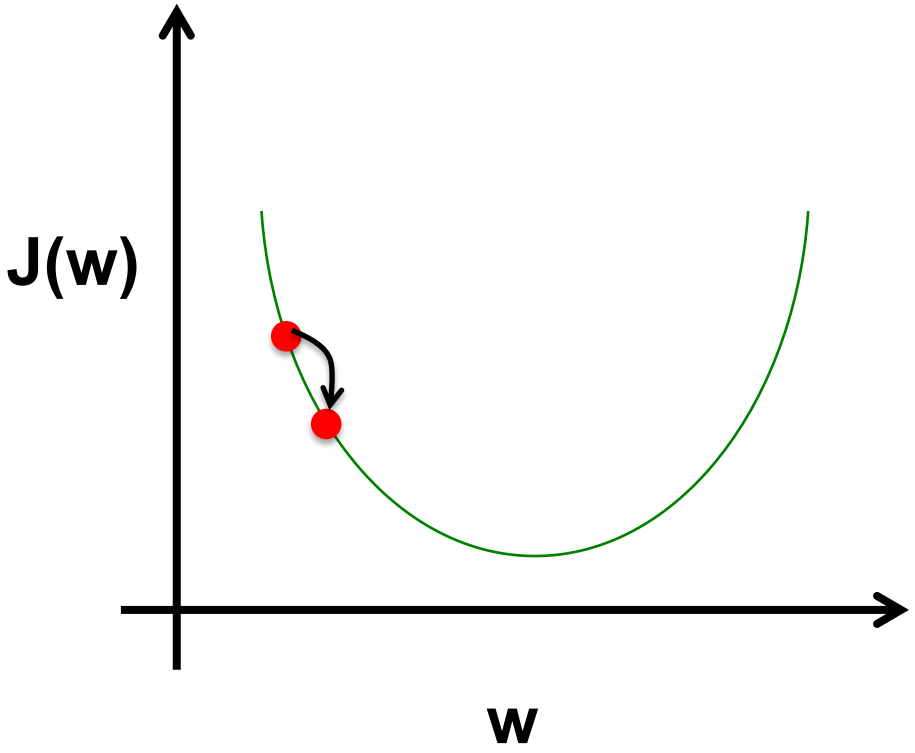
$$y = \begin{bmatrix} 432K \\ 218K \\ 366K \\ \vdots \\ 218K \end{bmatrix} \begin{array}{c} y^1 \\ \\ \\ \\ y^n \end{array}$$

See proof at: The Elements of Statistical Learning, T. Hastie, R. Tibshirani, J. Friedman, Page 12, and pages 44-45

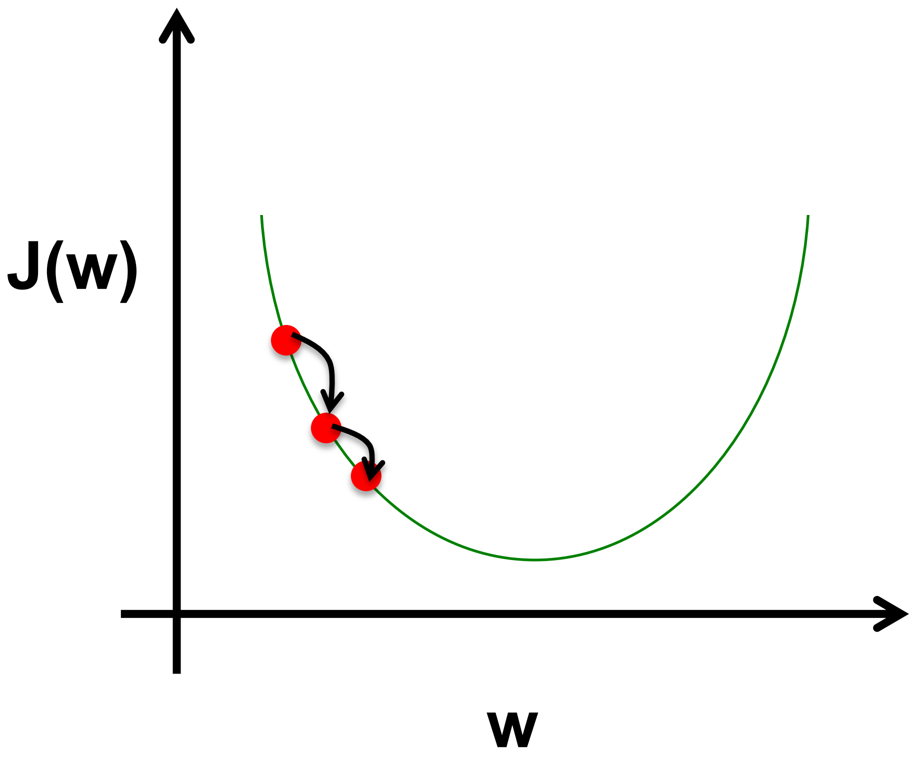
Gradient Descent



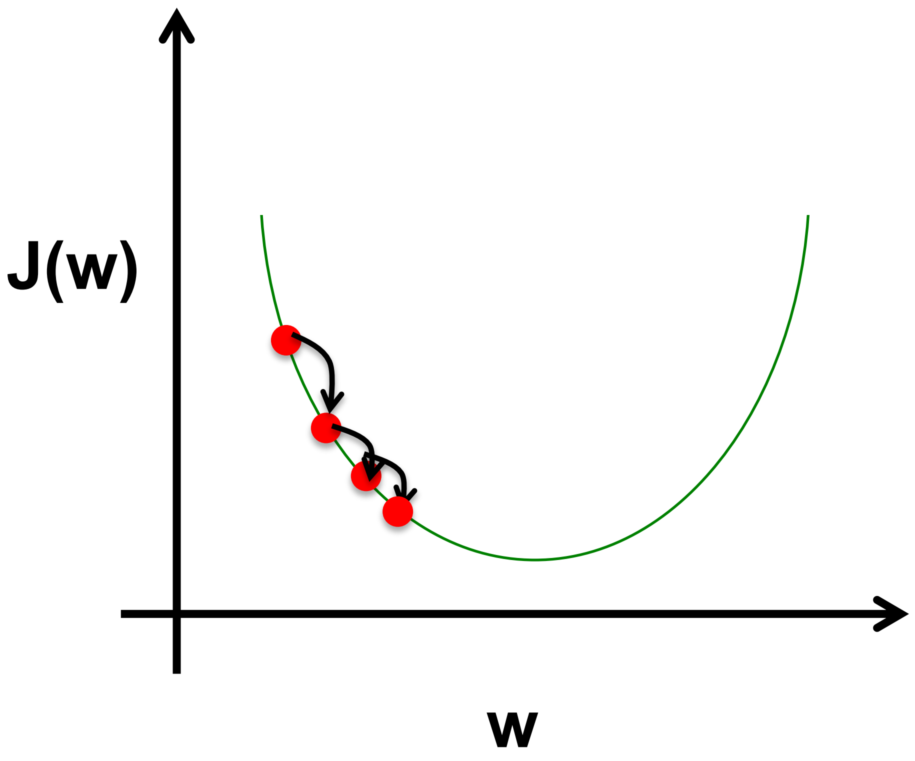
Gradient Descent



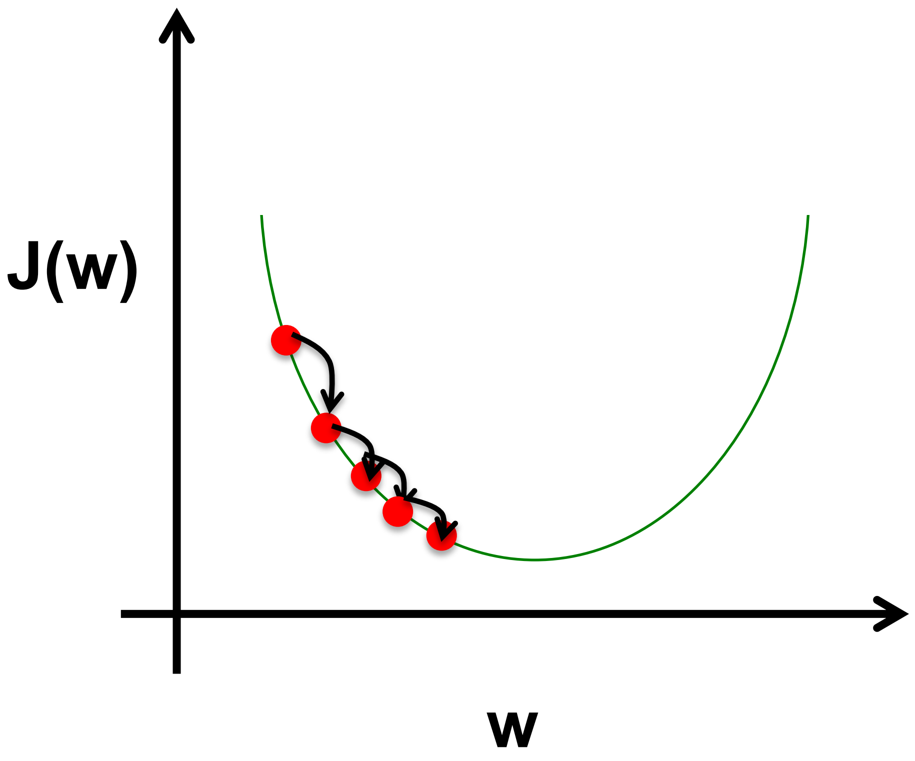
Gradient Descent



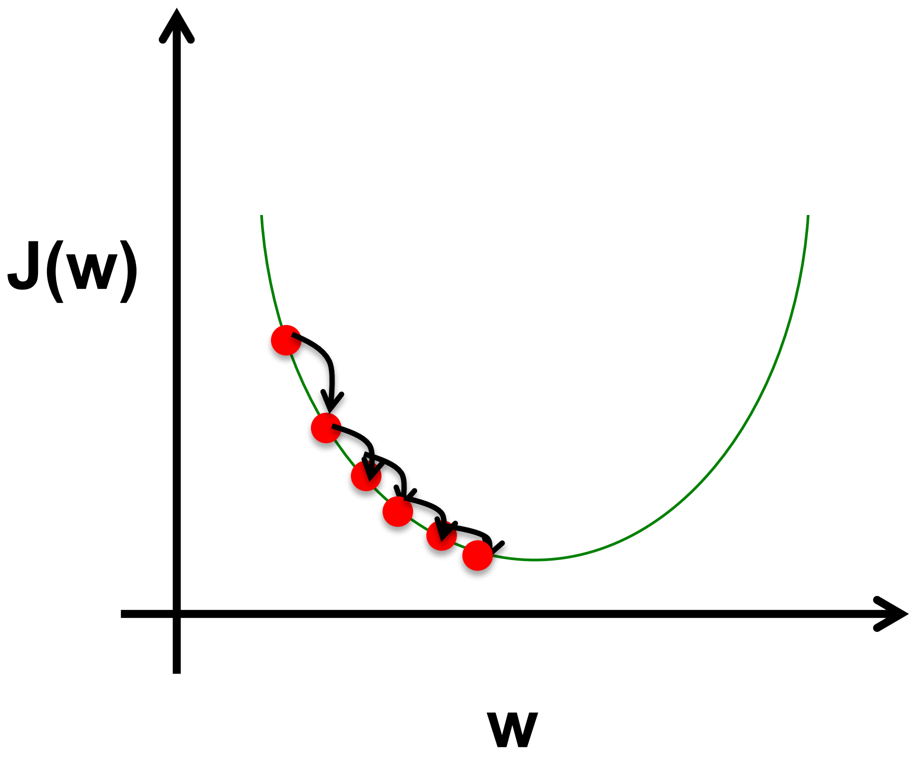
Gradient Descent



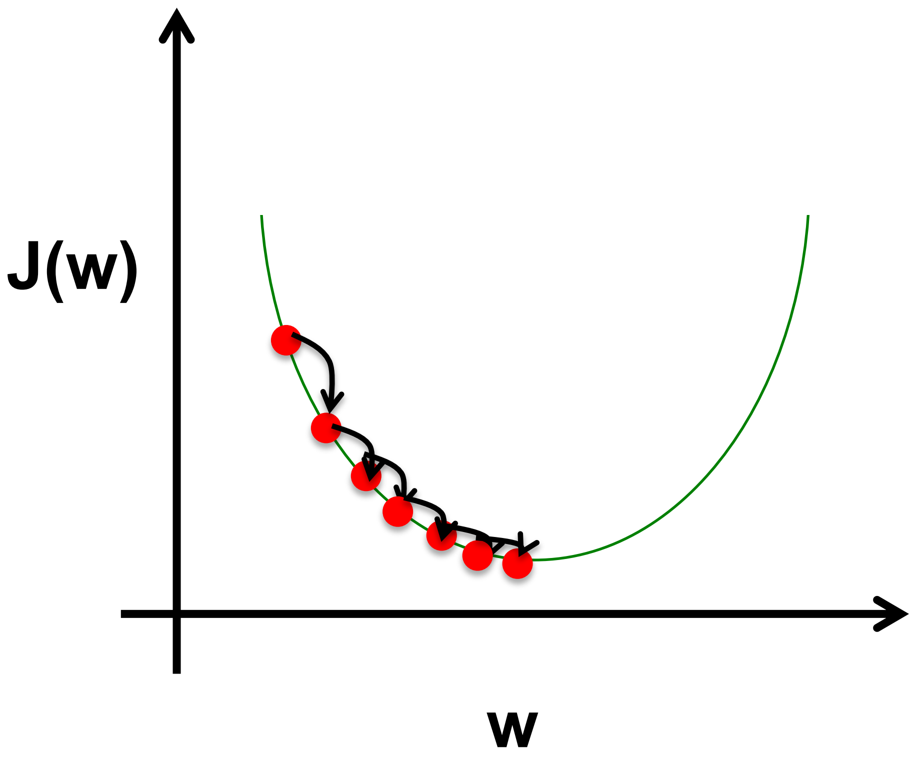
Gradient Descent



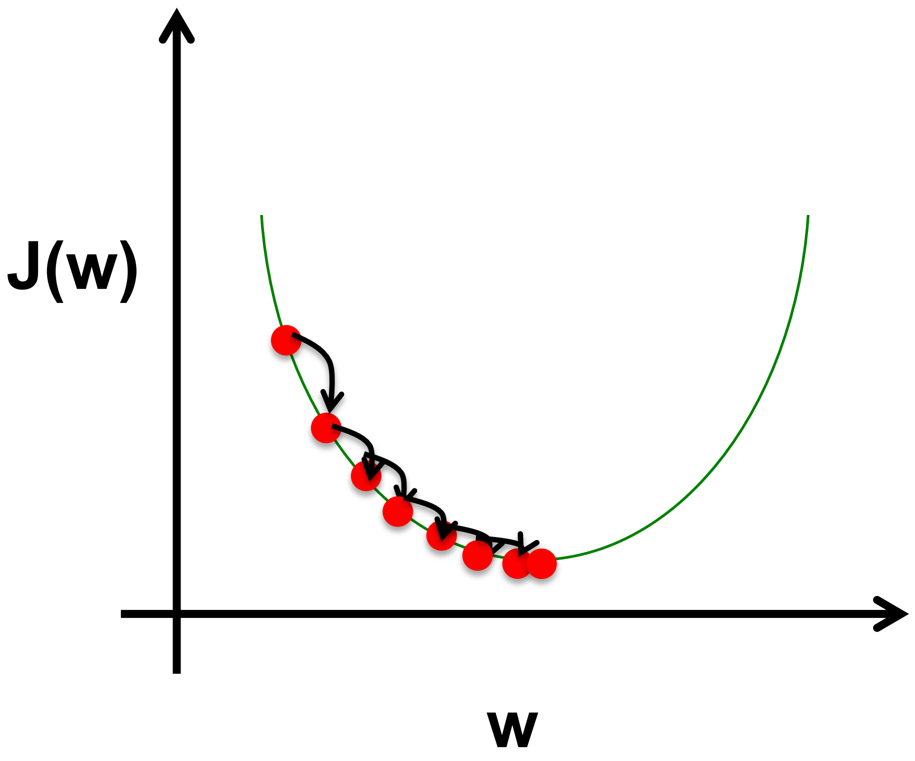
Gradient Descent



Gradient Descent



Gradient Descent

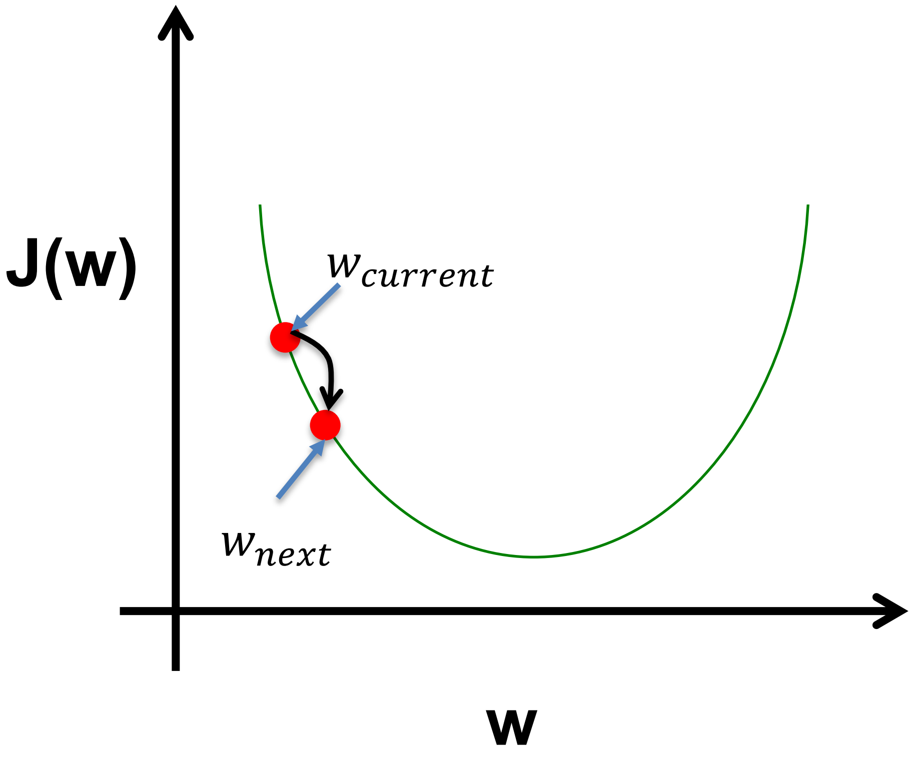


How to Formulate Gradient Descent?

For simplicity we start from 1-D *w*, then will extend the concepts to 2-D *w*’s.

$$w_{next} = w_{current} - \alpha \frac{\partial J(w)}{\partial w}$$

α is learning rate.

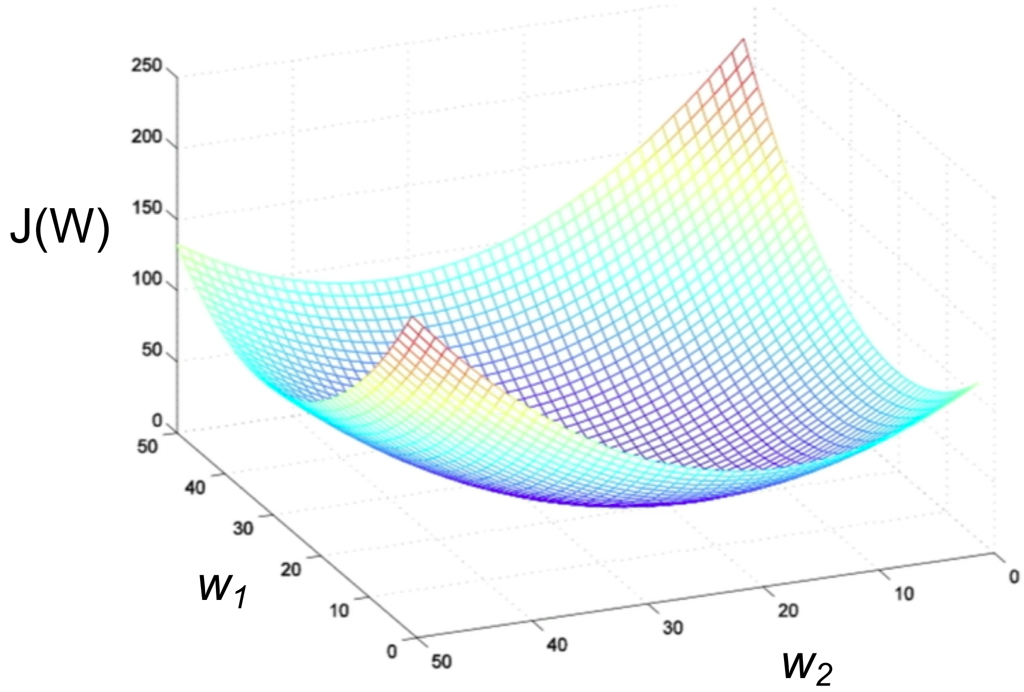


Vectorized Gradient Descent in 2D

$$W_{next} = W_{current} - \alpha \nabla_W J(W)$$

$$\frac{\partial J(W)}{\partial w_j} = \frac{1}{m} \sum_i x_{pj}^i (x_p^i . W - y^i)$$

$$\nabla_W J(W) = \frac{1}{m} X_p^T . (X_p \, . W - y)$$



"Batch" Gradient Descent

$$W_{next} = W_{current} - \alpha \nabla_W J(W)$$

$$\nabla_W J(W) = \begin{bmatrix} \frac{\partial}{\partial w_0} J(W) \\ \frac{\partial}{\partial w_1} J(W) \\ \vdots \\ \frac{\partial}{\partial w_k} J(W) \end{bmatrix}$$

$$\frac{\partial J(W)}{\partial w_j} = \frac{1}{m}\sum_i x_{pj}^i (x_p^i . W - y^i)$$

Loops over the entire training set.

"Stochastic" Gradient Descent

$$W_{next} = W_{current} - \alpha \nabla_W J(W)$$

$$\nabla_W J(W) = \begin{bmatrix} \frac{\partial}{\partial w_0} J(W) \\ \frac{\partial}{\partial w_1} J(W) \\ \vdots \\ \frac{\partial}{\partial w_k} J(W) \end{bmatrix}$$

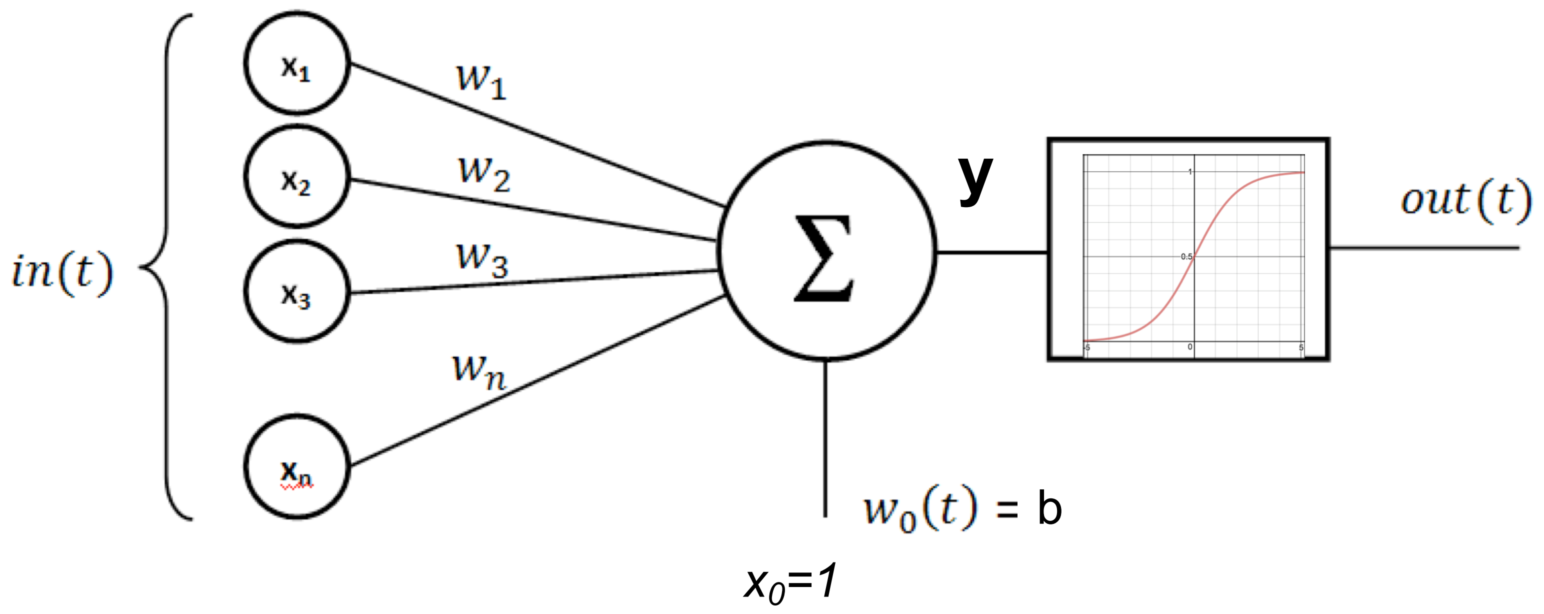
"Batch" Gradient Descent

$$\frac{\partial J(W)}{\partial w_j} = \frac{1}{m} \sum_i x_{pj}^i (x_p^i . W - y^i)$$

$$\frac{\partial J(W)}{\partial w_j} = x_{pj}^i (x_p^i . W - y^i)$$

i is randomly selected.

- Mini-Batch Gradient Descent: Something between Batch Gradient Descent and Stochastic Gradient Descent. Still very stochastic for small batch sizes.



$$y = b + w_1\ x_1\ + w_2\ x_2\ + \ldots + w_n\ x_n$$
$$= w_0\ x_0 + w_1\ x_1\ + w_2\ x_2\ + \ldots + w_n\ x_n$$
$$= X_p . W$$

$$output = s(y)$$

$$S(y) = \frac{1}{1 + e^{-y}}$$

Cost Function

$$J(W) = -\frac{1}{m}\sum_{i=1}^{m}[y^i \log\left(x_p^i . W\right) + \left(1 - y^i\right)\log\left(1 - x_p^i . W\right)$$

$$\frac{\partial J(W)}{\partial w_j} = \frac{1}{m}\sum_i x_{pj}^i(\widehat{y^i} - y^i)$$

XOR Problem:

XOR is not linearly separable

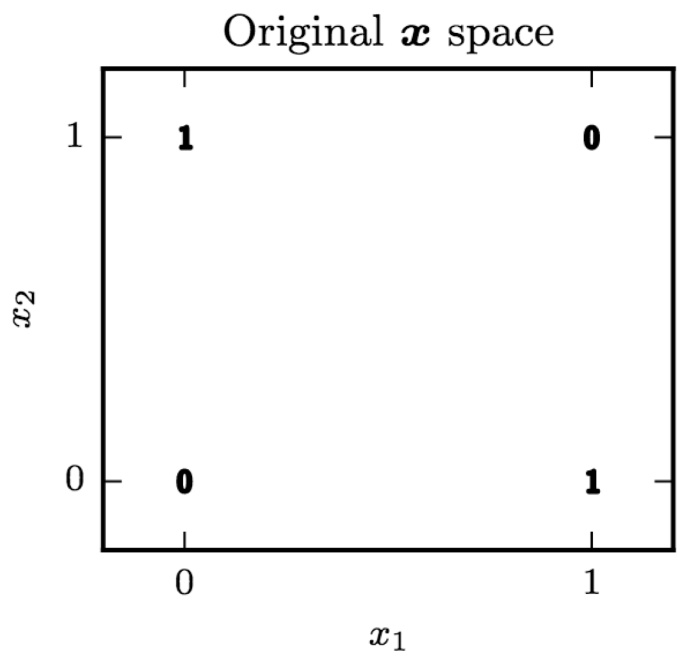


Figure 6.1, left

Slides are from Goodfellow, et.al for “Deep Learning” Book

XOR Problem: What can we do?

- Apply linear models not to input *x*, but to a transformed input $\varphi(x)$, where φ is a nonlinear transformation.

XOR Problem: What can we do?

- Multilayer Neural Network (AKA Deep Learning):
  - The strategy is to learn φ as well

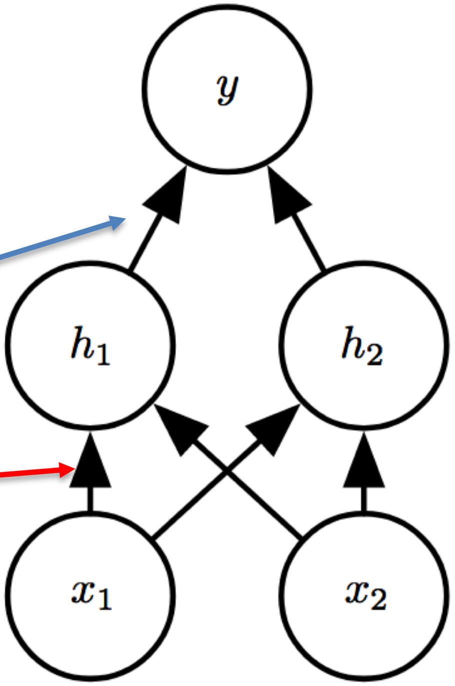
$\hat{y}= \varphi\left(x, w_{\varphi}\right). w_C$

Up until now we had: $\hat{y}=x. w$

XOR Problem: What can we do?

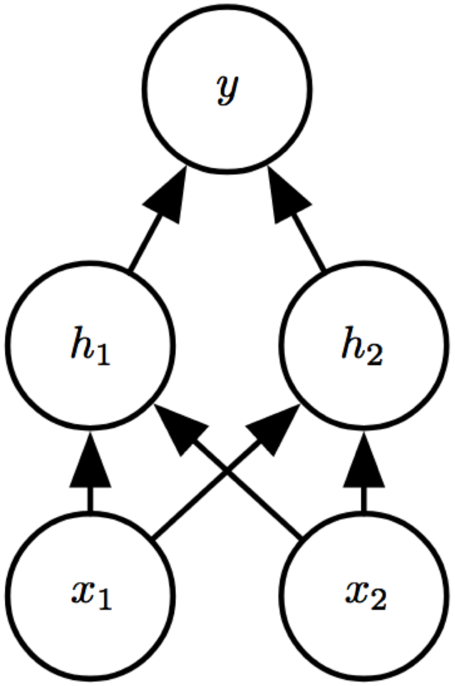
- Multilayer Neural Network (AKA Deep Learning):
  - The strategy is to learn φ as well

$$\hat{y}= \varphi(x, w_{\varphi}).w_C$$



Up until now we had: $\hat{y}=x.w$

Multilayer Neural Network (AKA Deep Learning)



- Logistic unit $\Longrightarrow$ Neuron Model
- Parameters $\Longrightarrow$ weights
- Output function $\Longrightarrow$ Activation function

Multilayer Feedforward Neural Network (Deep Feedforward Networks) Good News

A Feedforward Neural Network with one hidden layer can represent and approximate any function to an arbitrary degree of accuracy.

This is called Universal Approximation Theorem.

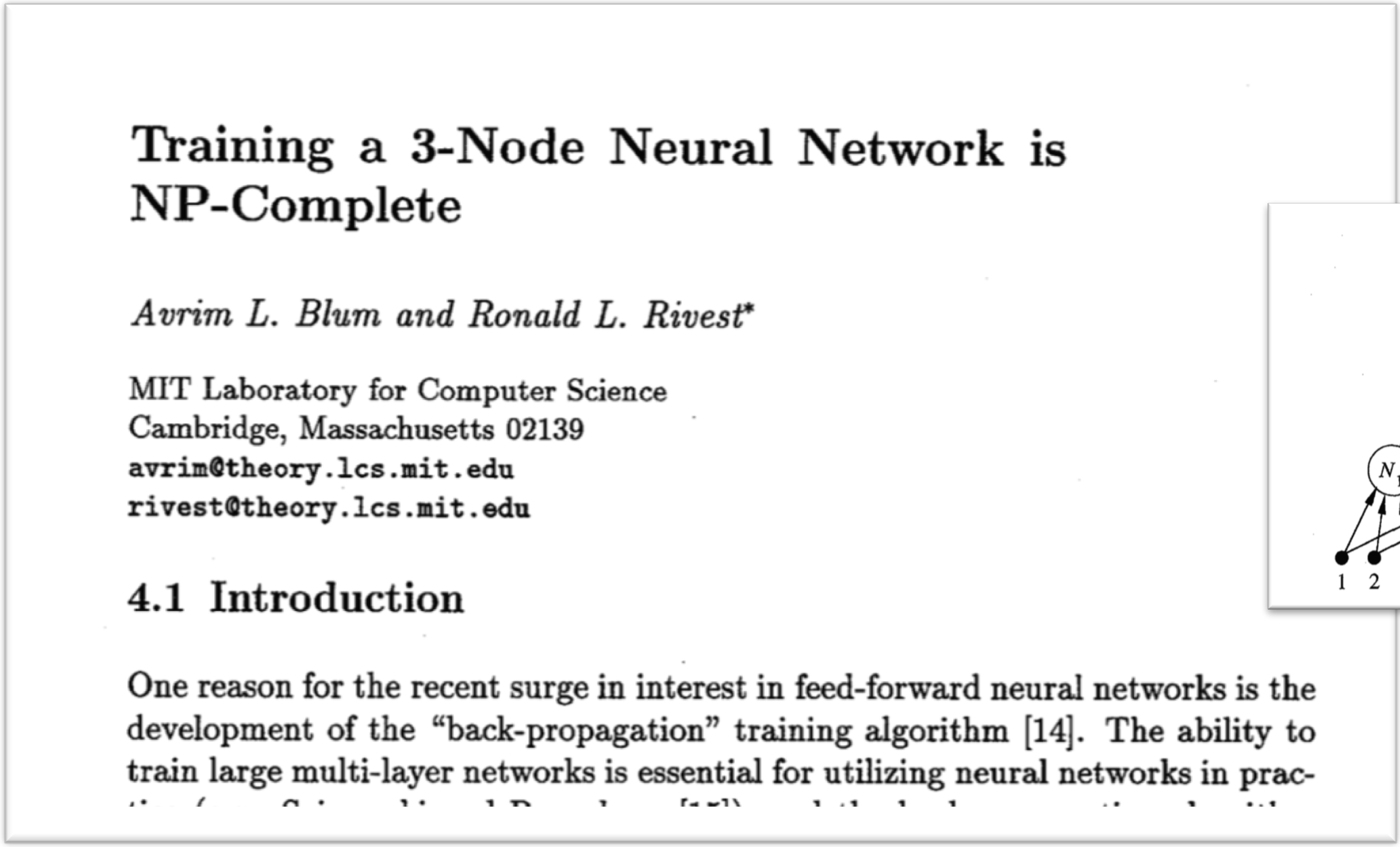
Multilayer Feedforward Neural Network (Deep Feedforward Networks) Good News

- Deeper networks are much more powerful than shallow networks.
- Shallow network may need exponentially more width (neurons in a layer) to implement the same function.

Eldan, Ronen, and Ohad Shamir. "**The power of depth for feedforward neural networks**." *Conference on Learning Theory*. 2016.

J. Hastad. **Almost optimal lower bounds for small depth circuits**. ACM symposium on Theory of computing,. ACM, 1986

Multilayer Feedforward Neural Network (Deep Feedforward Networks) Bad News



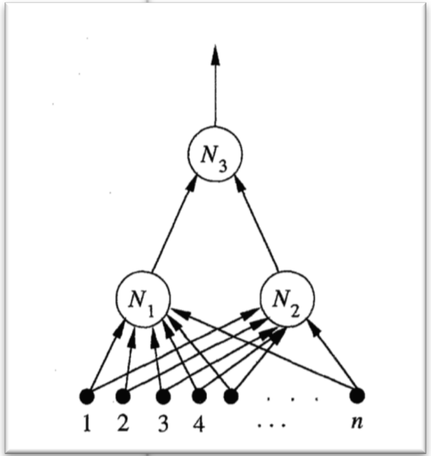
Training a 3-Node Neural Network is NP-Complete

*Avrim L. Blum and Ronald L. Rivest**

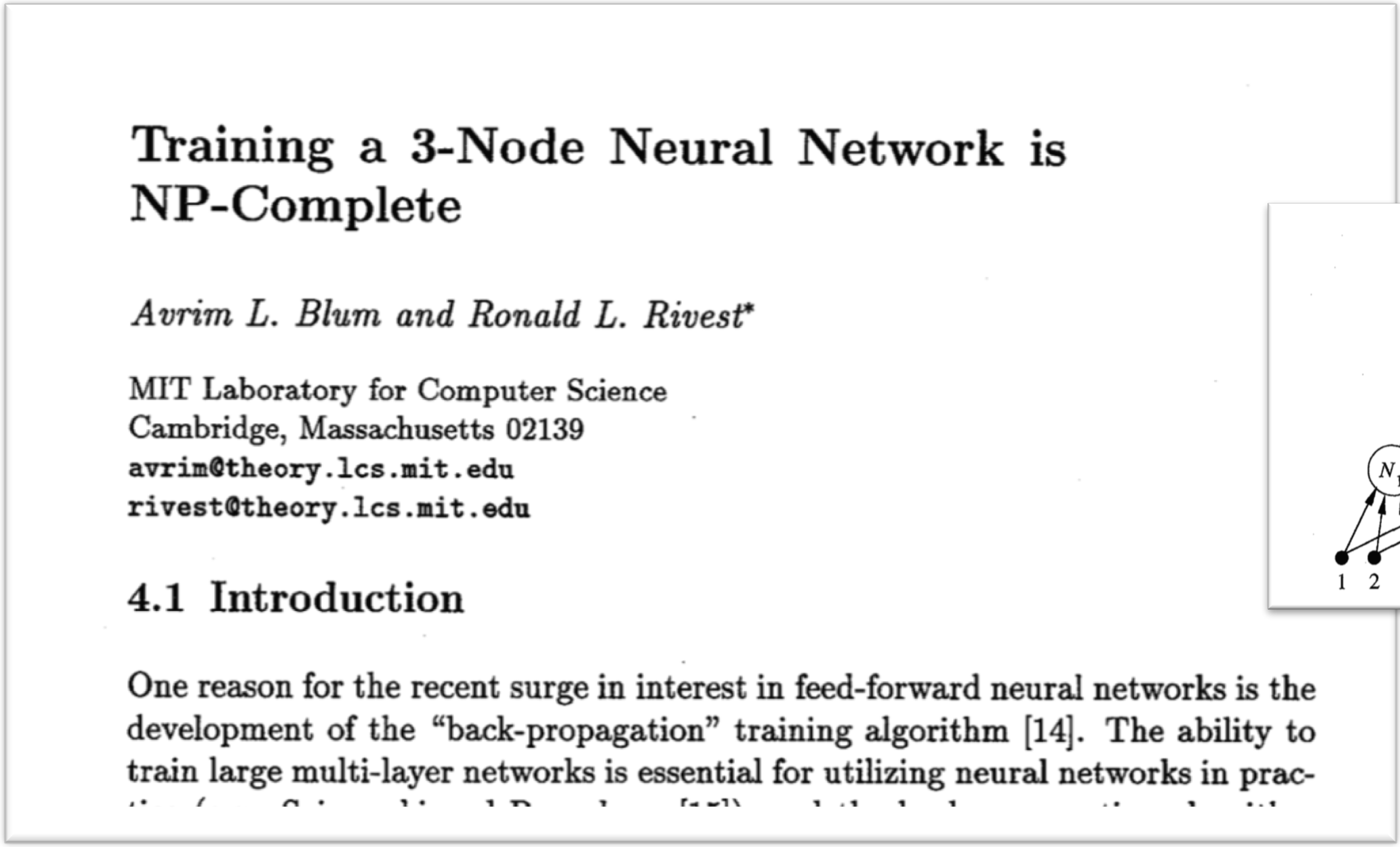
MIT Laboratory for Computer Science
Cambridge, Massachusetts 02139
avrim@theory.lcs.mit.edu
rivest@theory.lcs.mit.edu

4.1 Introduction

One reason for the recent surge in interest in feed-forward neural networks is the development of the "back-propagation" training algorithm [14]. The ability to train large multi-layer networks is essential for utilizing neural networks in prac-



Multilayer Feedforward Neural Network (Deep Feedforward Networks) Bad News



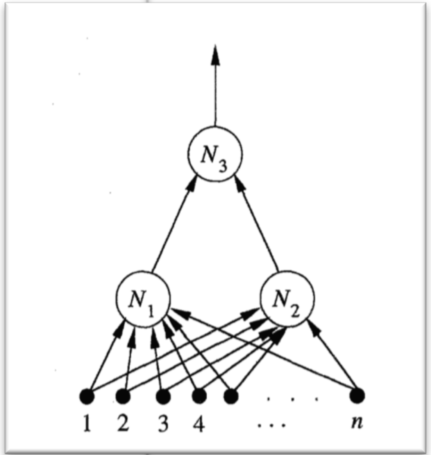
Training a 3-Node Neural Network is NP-Complete

*Avrim L. Blum and Ronald L. Rivest**

MIT Laboratory for Computer Science
Cambridge, Massachusetts 02139
avrim@theory.lcs.mit.edu
rivest@theory.lcs.mit.edu

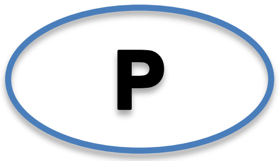
4.1 Introduction

One reason for the recent surge in interest in feed-forward neural networks is the development of the "back-propagation" training algorithm [14]. The ability to train large multi-layer networks is essential for utilizing neural networks in prac-



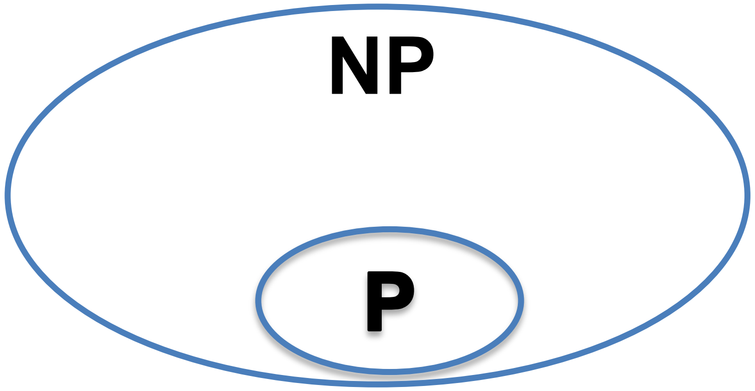
Complexity Classes

P: Can be solved in polynomial time.



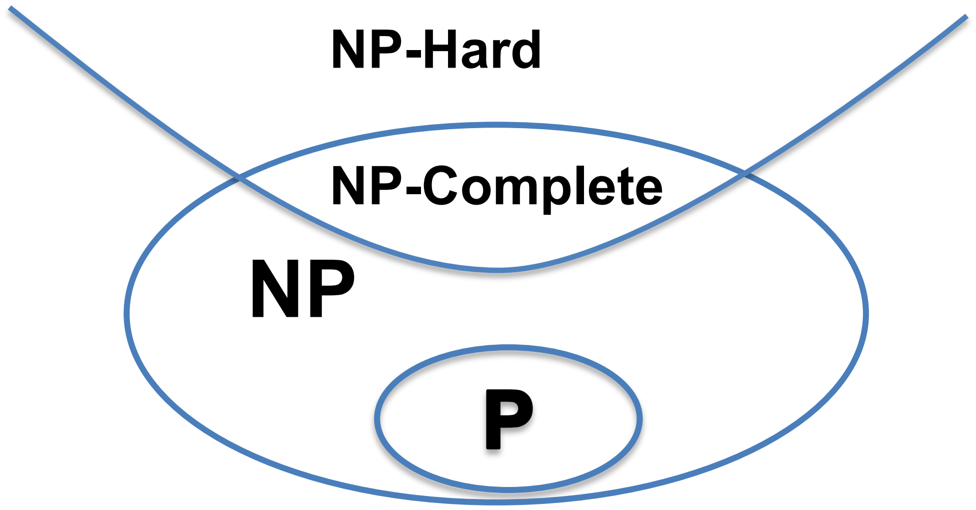
Complexity Classes

P: Can be solved in polynomial time.
NP: Solutions verifiable in polynomial time. Can be solved in-deterministic polynomial time. (informal definition) No polynomial solution yet.



Complexity Classes

P: Can be solved in polynomial time.
NP: Solutions verifiable in polynomial time. Can be solved in-deterministic polynomial time. (informal definition) No polynomial solution yet.
NP-Complete: The hardest NP problems.
NP-Hard: At least as hard as the hardest NP problem.



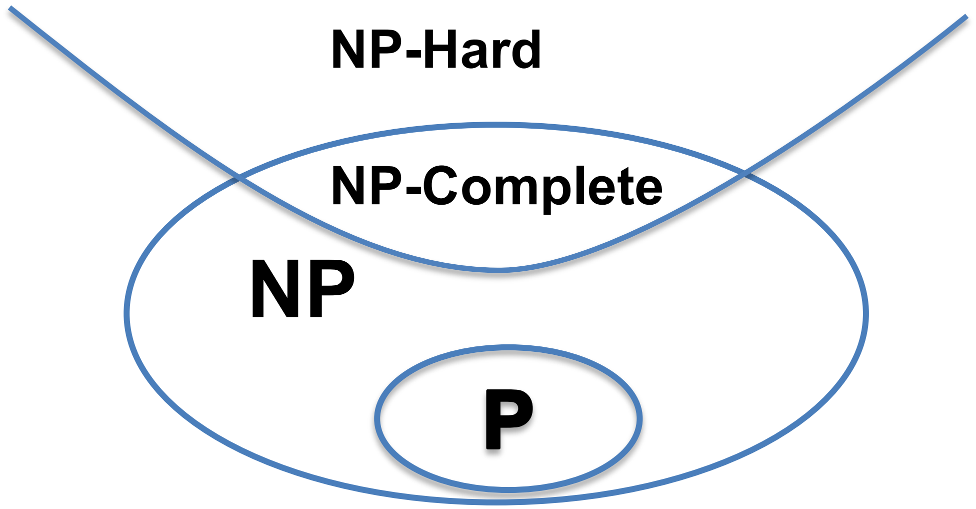
Complexity Classes: One Million Dollar Question!

P: Can be solved in polynomial time.

NP: Solutions verifiable in polynomial time. Can be solved in-deterministic polynomial time. (informal definition) No polynomial solution <u>yet</u>.

NP-Complete: The hardest NP problems.

NP-Hard: As hard as the hardest NP problem.



$P \overset{?}{=} NP$

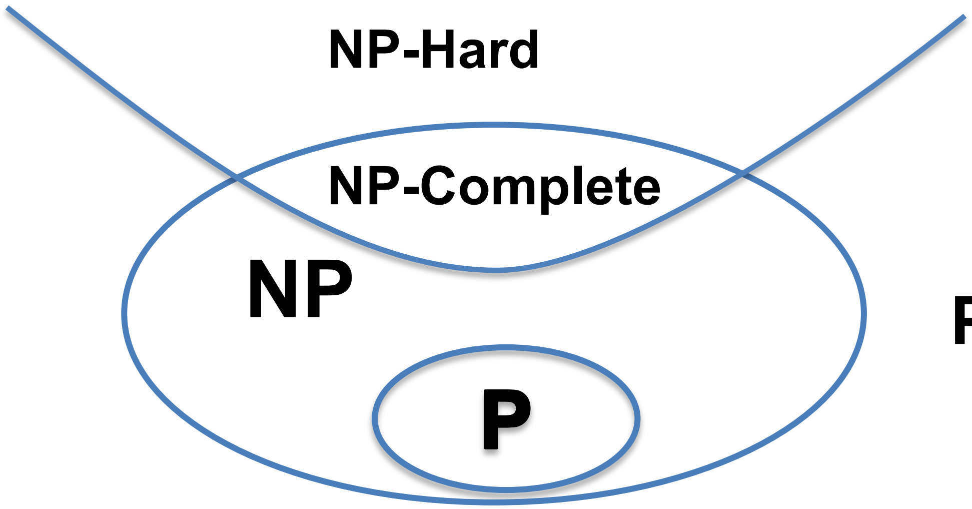
Complexity Classes: One Million Dollar Question!

P: Can be solved in polynomial time.

NP: Solutions verifiable in polynomial time. Can be solved in-deterministic polynomial time. (informal definition) No polynomial solution yet.

NP-Complete: The hardest NP problems.

NP-Hard: As hard as the hardest NP problem.



P≠NP many believe

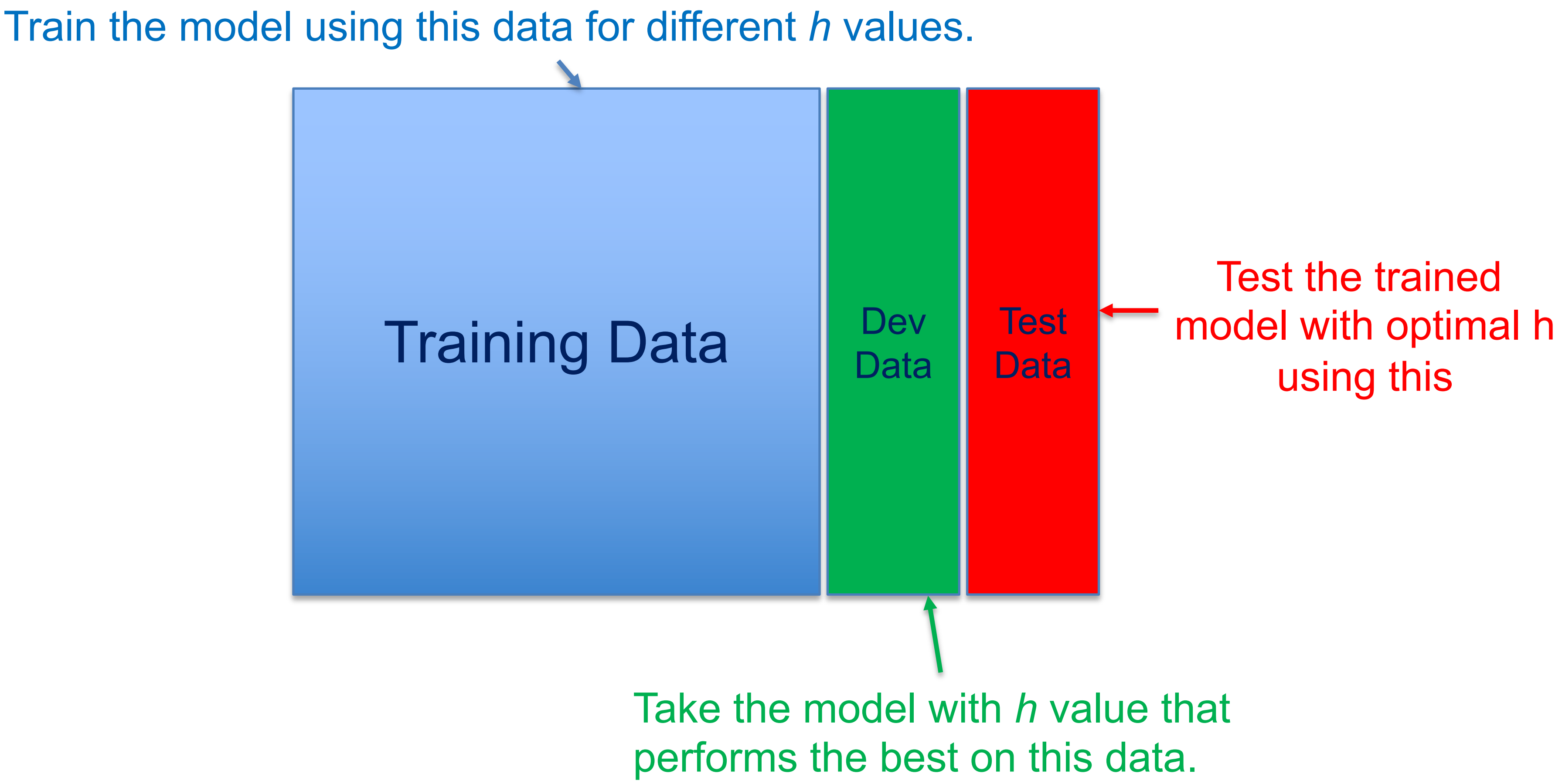
Stochastic Gradient Descent

- `my_model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])`
- `my_model.fit(X_train, Y_train, epochs=50, batch_size=10, verbose=2)`

Multilayer Feedforward Neural Network (Deep Feedforward Networks) Bad News

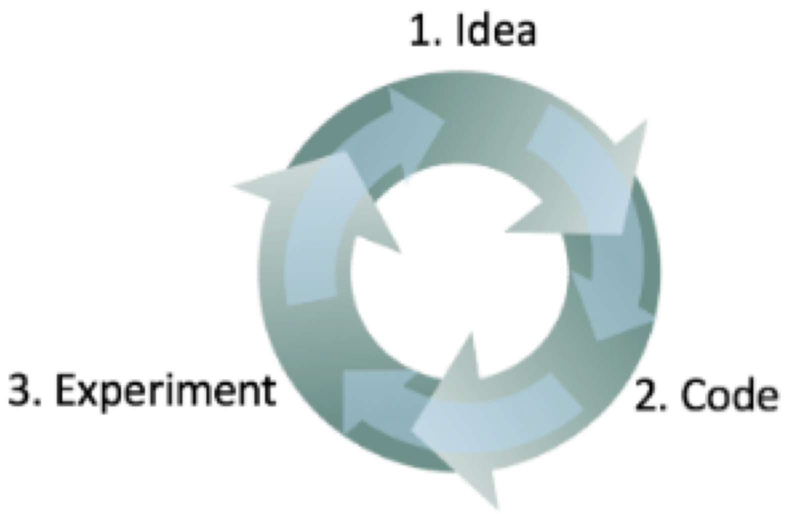
Universal Approximation Theorem says that a large enough network can approximate any function.

But it doesn’t say how to train the network or learn those parameters to approximate the desired function.
And it says nothing about the architecture and size of the network.


Train the model using this data for different *h* values.
Training Data
Dev Data
Test Data
Test the trained model with optimal h using this
Take the model with *h* value that performs the best on this data.

- **Training set:** Which you run your learning algorithm on.
- **Development set (AKA Validation set):** Which you use to tune parameters, select features, and make other decisions regarding the learning algorithm.
- **Test set:** which you use to evaluate the performance of the algorithm, but not to make any decisions regarding what learning algorithm or parameters to use.

Design Process and Cycle for Machine Learning


1. Idea
2. Code
3. Experiment