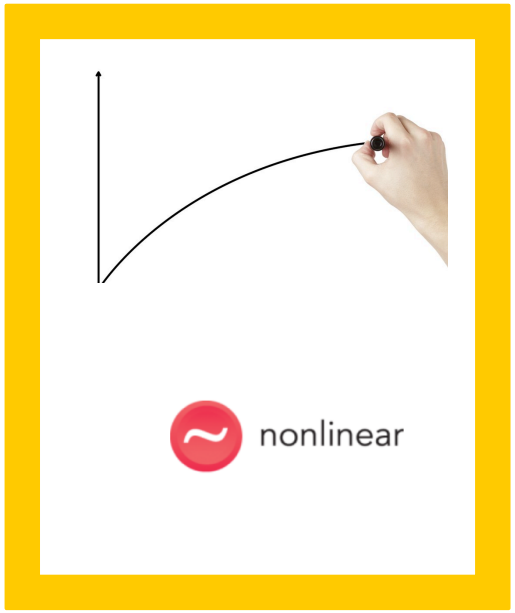
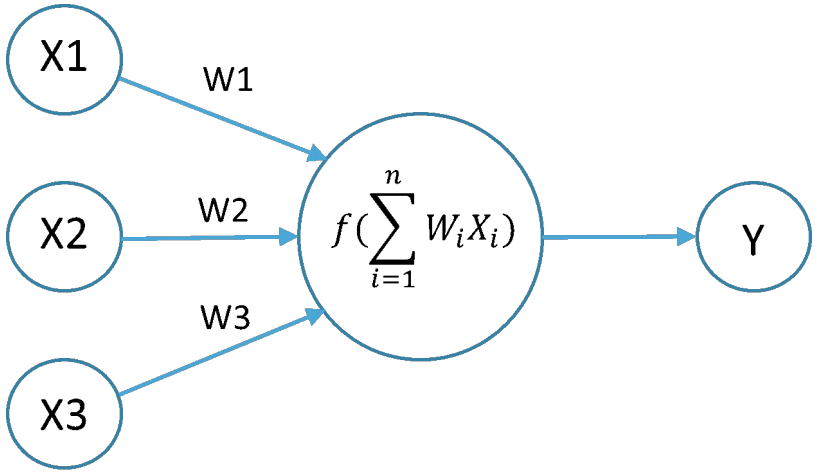
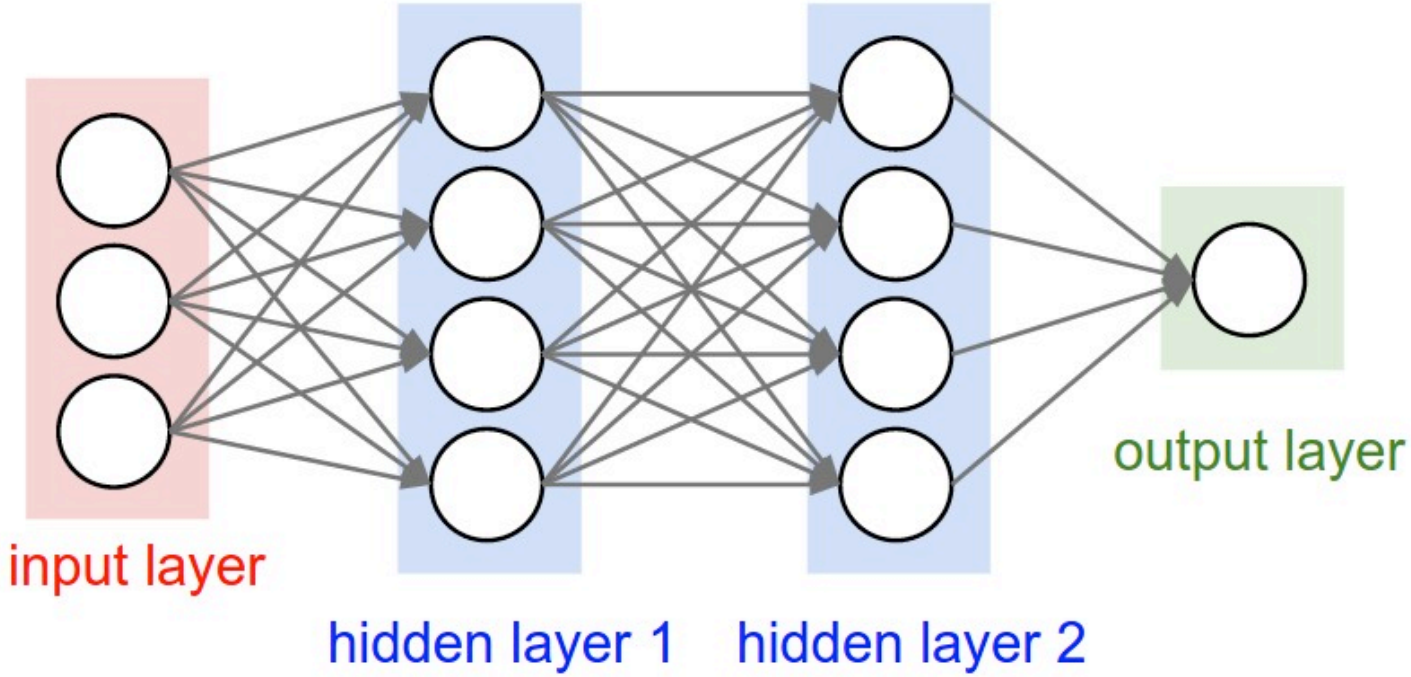


Towards Nonlinear Methods

Beyond the limitations of linear methods.

Review: Neural Networks





Neuron Model

Y = f($\mathbf{W}^T\mathbf{X}$)

Examples of Deep Neural Networks:

Fully Connected Deep Networks, MLP (Fig above)

CNN: Convolutional Neural Networks

RNN: Recurrent Neural Networks

Review: Pipeline/Concept Map

DATA

Structured (numerical, categorical attributes)
Digital Logs (Tweets, SMS)
RawData/Sensors (Image/Speech)
User behaviors
Etc.

FEATURE

Intuitive User defined
Raw data itself

Statistics (Histograms, PCA)

Signal Process (Fourier Xform)

FEATURE XFORMATIONS

Feature Selection

Feature Extraction

Dimensionality Reduction

Data Visualization
Eg. PCA, tSNE

ML PROBLEM

1. Classification
   a. Binary
   b. Multiclass
2. Regression
3. Clustering
4. Prediction (time series)

ALGORITHMS

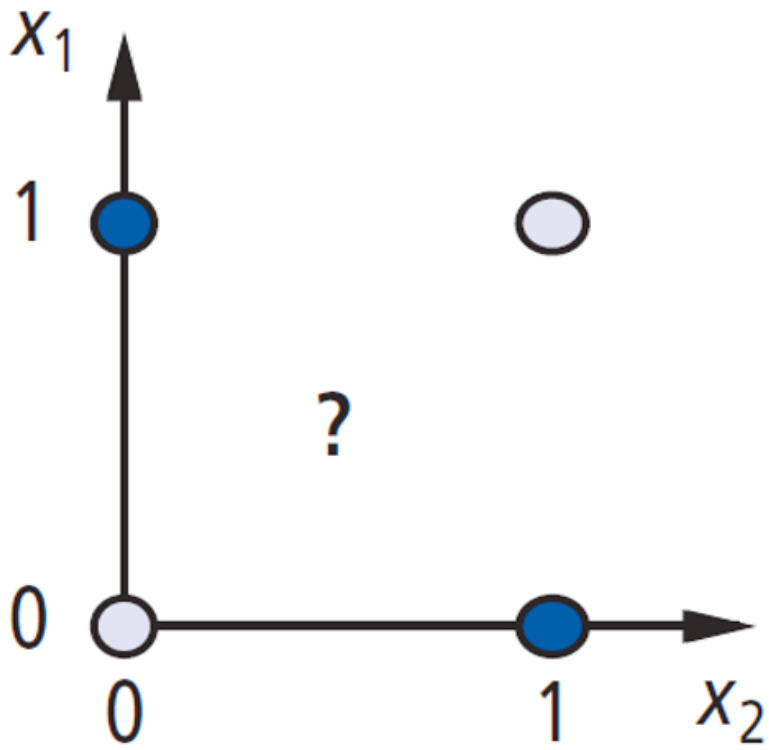
1. KNN
2. Naïve Bayes
3. Perceptron
4. Logistic Regression
5. Linear Classifier
6. K Means

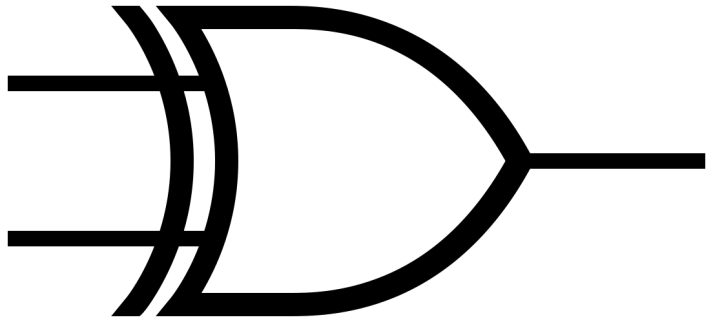
Objective

Loss Function
Optimization
Gradient Descent

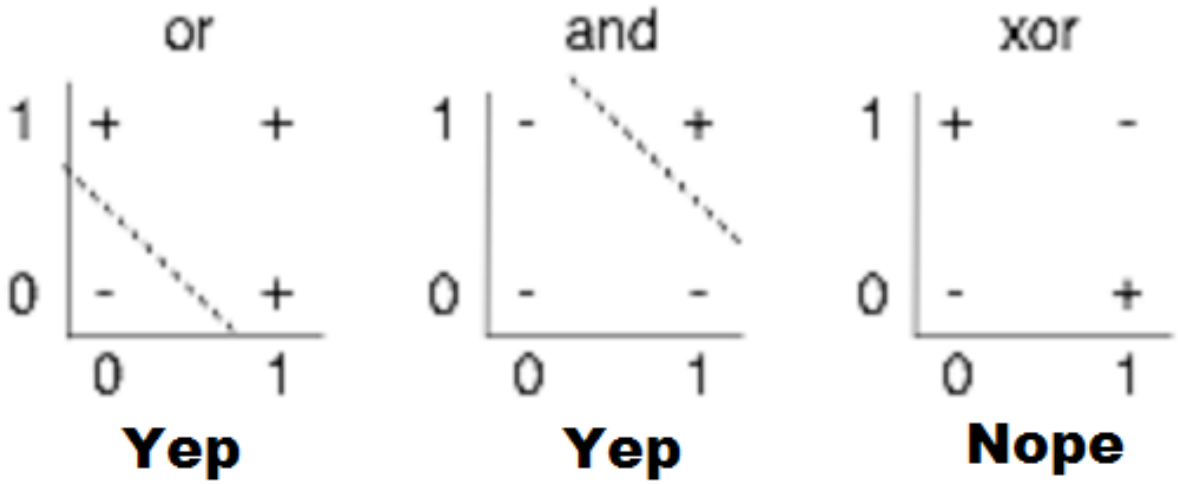
XOR: Limitation of Linear Methods

x_1	x_2	x_1 **XOR** x_2
0	0	0
0	1	1
1	0	1
1	1	0



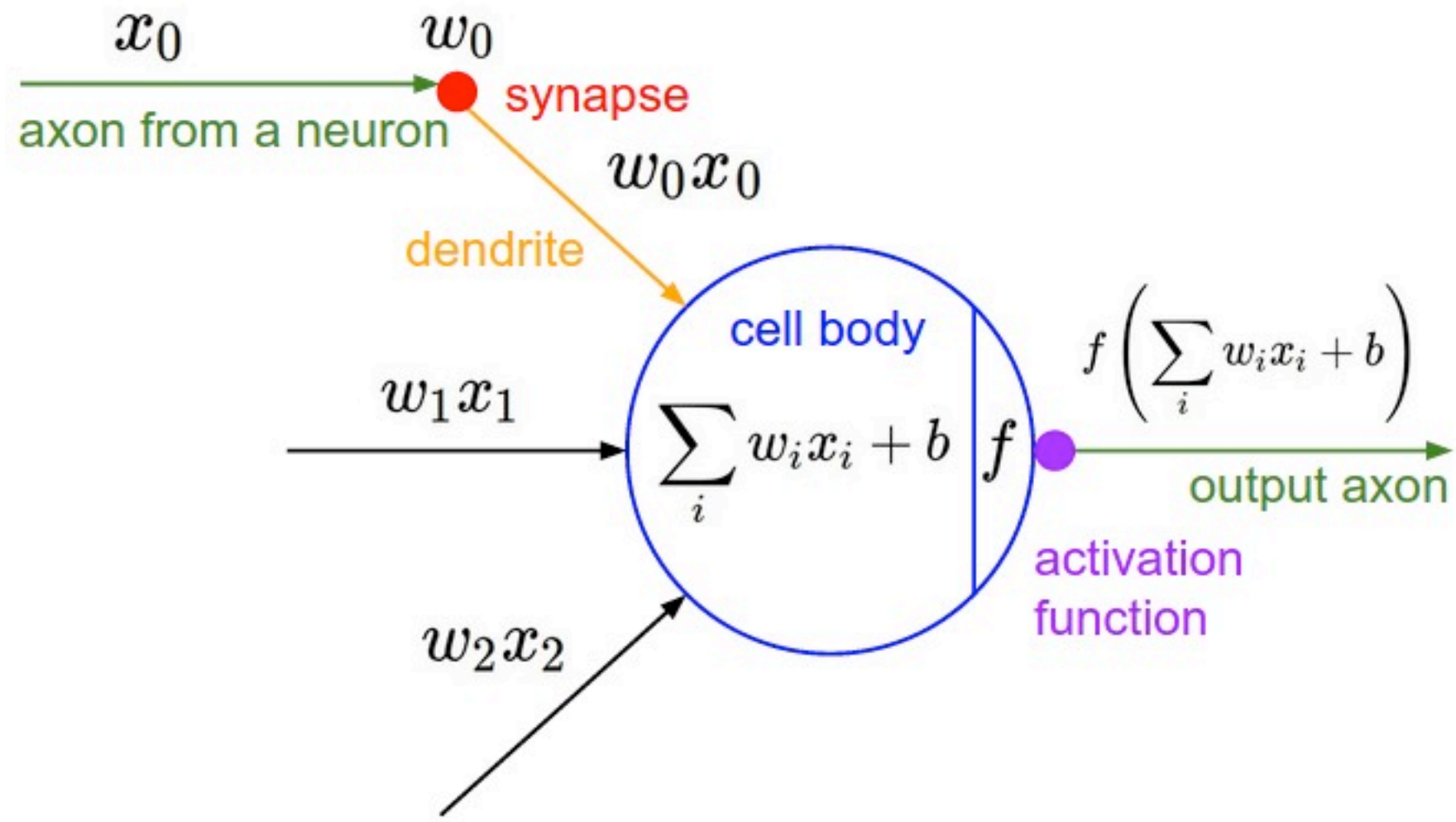


XOR: Linear “Non Separability”



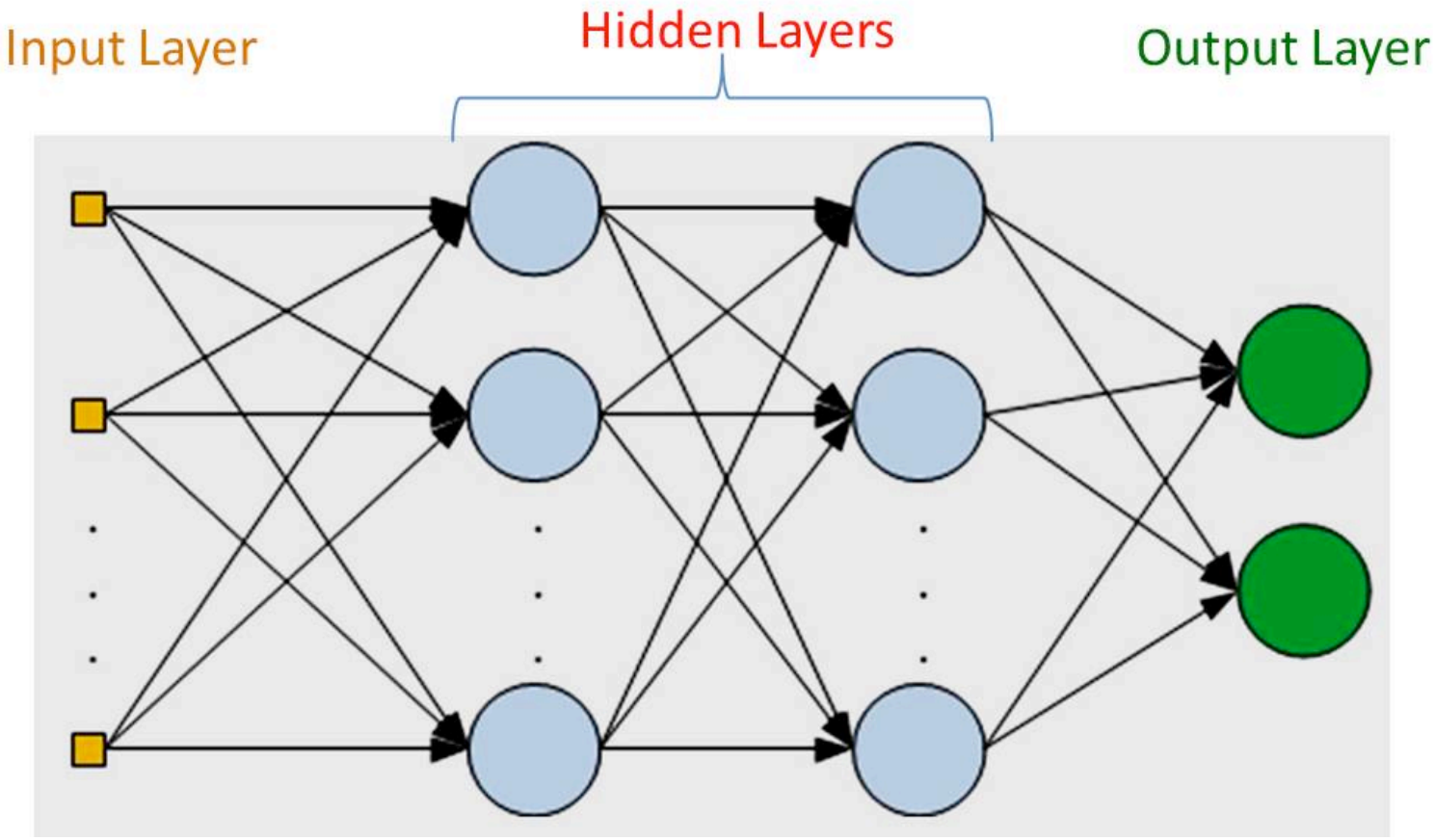
Not all concepts can be represented with linear functions.

Review: Neuron Model



Neural Network: Simple Abstract View

Multi Layer Perceptron (MLP)



Two Computational Blocks/Steps

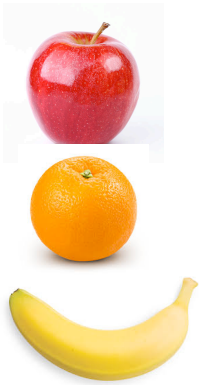
$$\mathbf{y} = \mathbf{W}\mathbf{x}$$

$$\mathbf{z} = \phi(\alpha) = \frac{1}{1 + e^{-\alpha}}$$

Sigmoid Nonlinearity

Multi Class Classification using MLP

- Input: $(\mathbf{X_i}, \mathbf{y_i})$
  - $\mathbf{X}$ = [x1,x2,x3,…,xd]
- Encode label **y** as
  - [1,0,0] for class 1
  - [0,1,0] for class 2
  - [0,0,1] for class 3

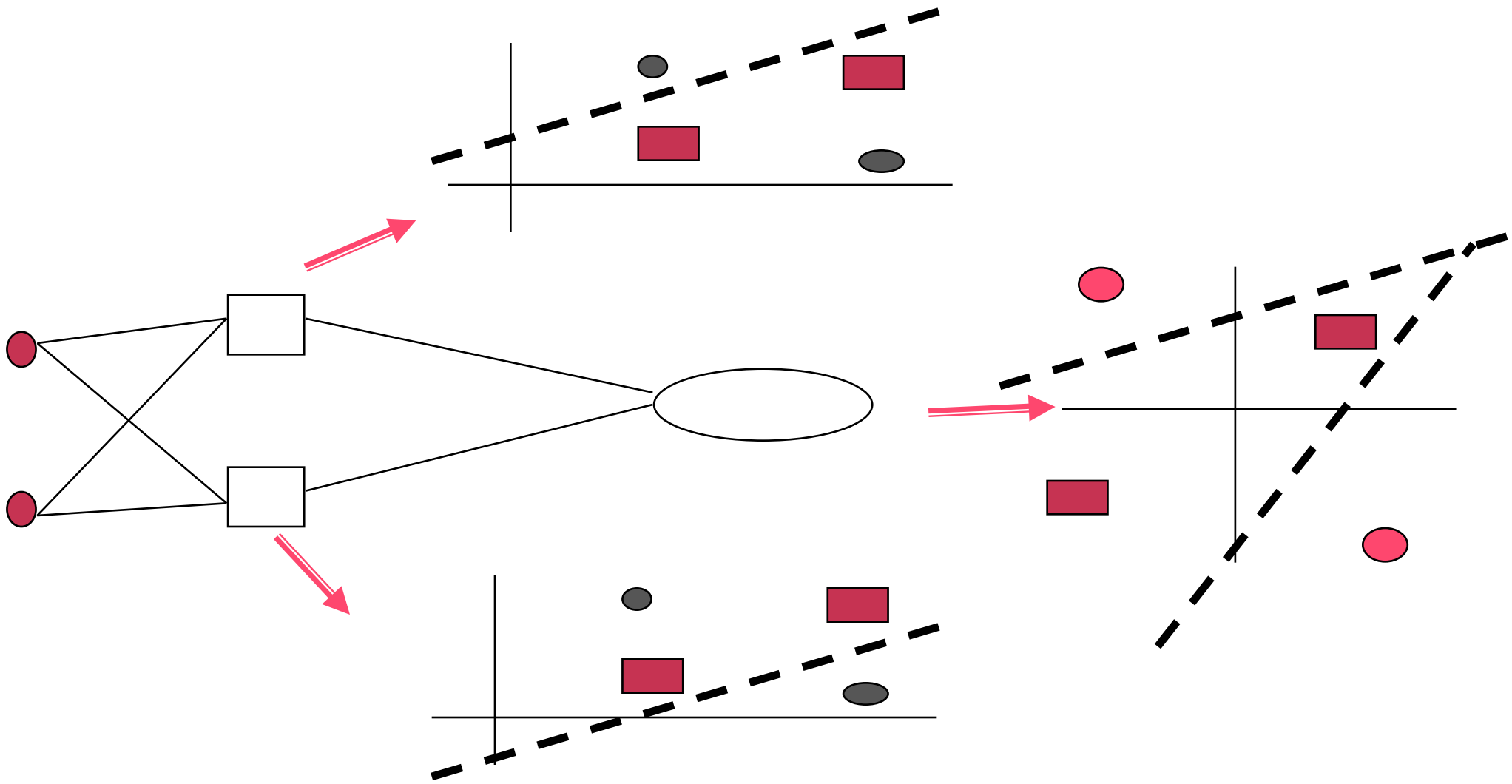


- Loss
  - MSE (Mean square error)

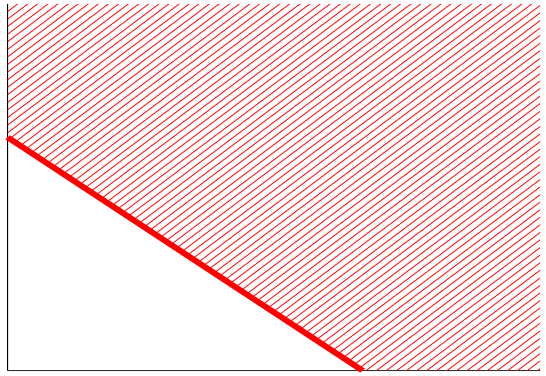
- Let predicted label be **z.**
- Remains the same even for regression.
- Our objective.
  - Minimize the difference between $\mathbf{z_i}$ and $\mathbf{y_i}$ for each i

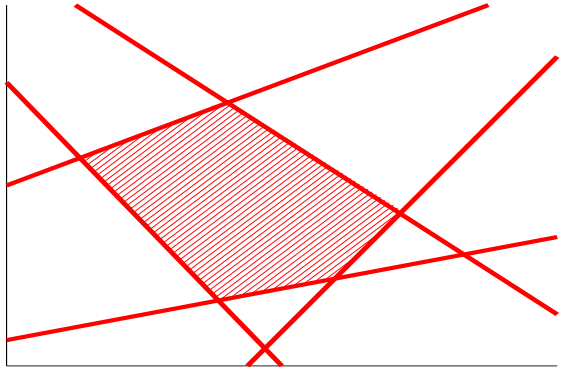
$$L(W) = \sum_i ||\mathbf{z_i} - \mathbf{y_i}|| = \sum_i \sum_j (z_{ij} - y_{ij})^2$$

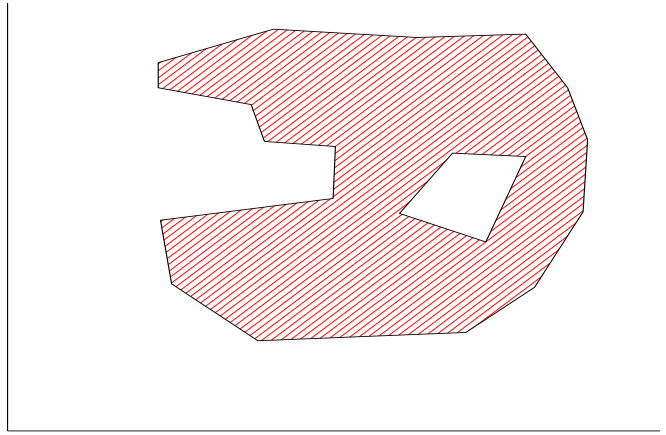
How MLP Works? (A naïve view)

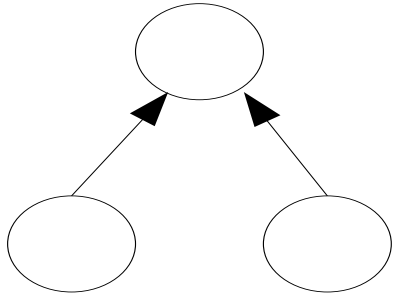


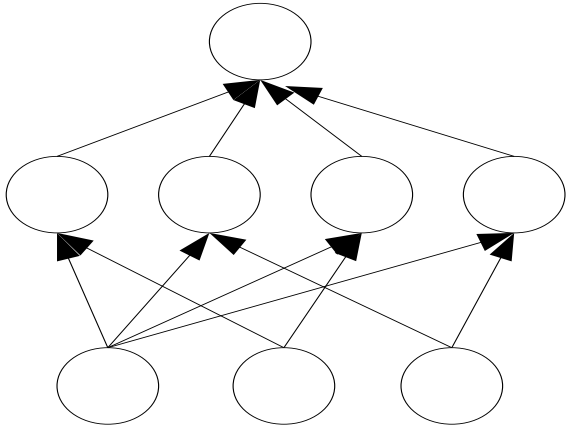
Deeper Networks

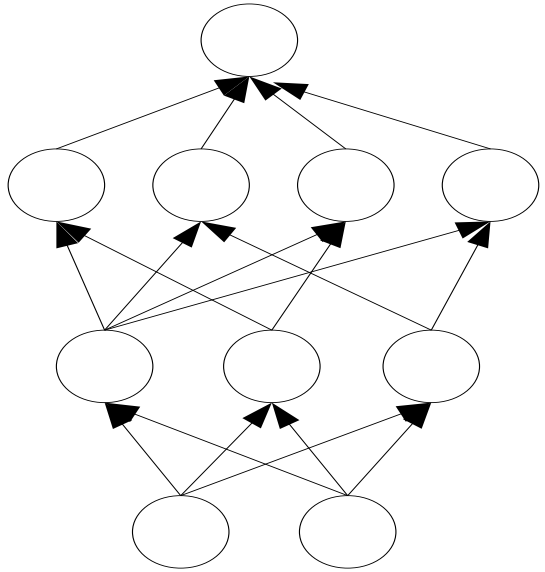




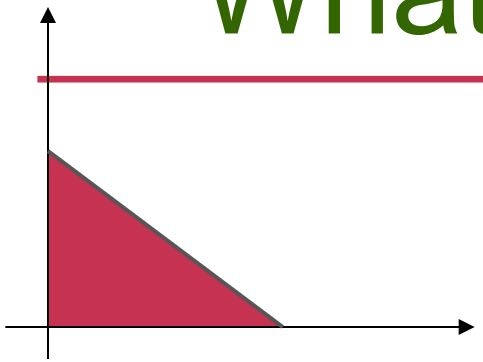


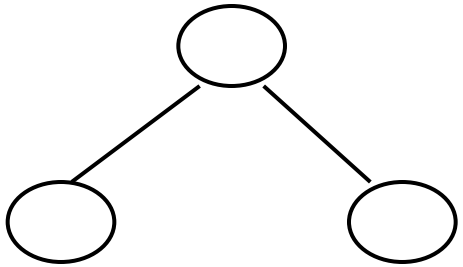




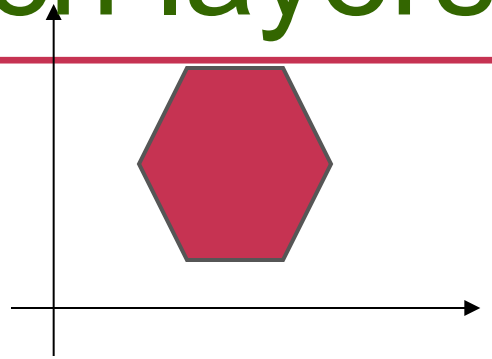


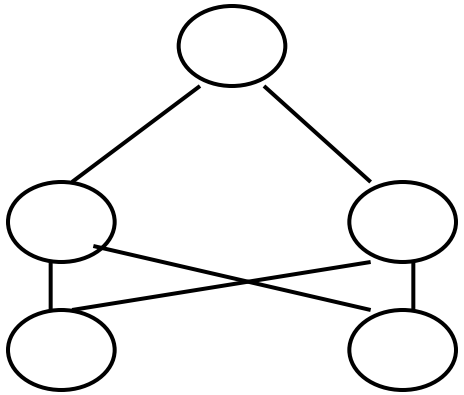
What do each layers do?



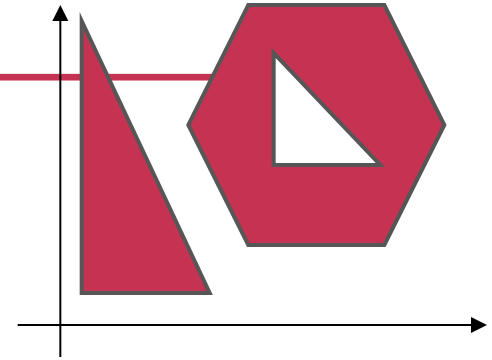


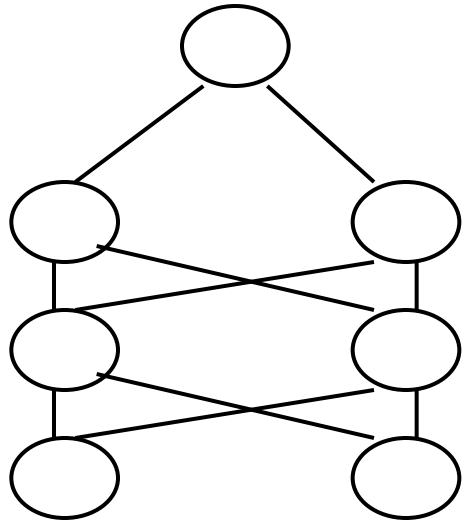
1st layer draws linear boundaries





2nd layer combines the boundaries





3rd layer can generate arbitrarily complex boundaries

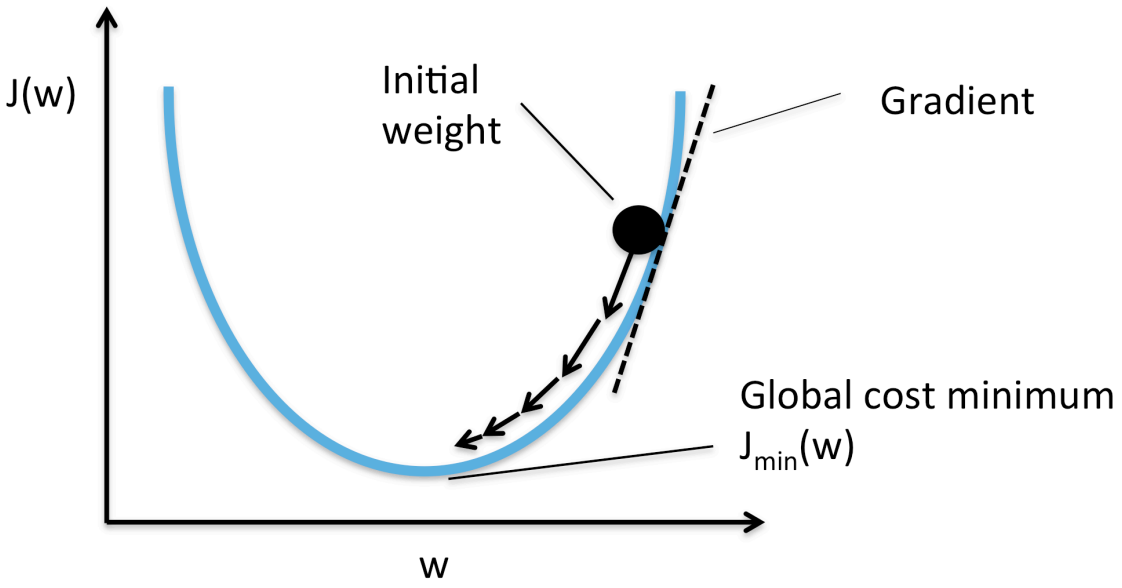
Summary

- Many “perceptron” networks can be stacked to generate Multi Later Perceptron (MLP).
- Any arbitrary function can be approximated.
  - Given that we can train!! (this could be tricky)
- Classically the nonlinearity is a simple sigmoid or similar fns.
- Often people use MLPs with one or two hidden layers.
  - Not very deep.

Training MLP

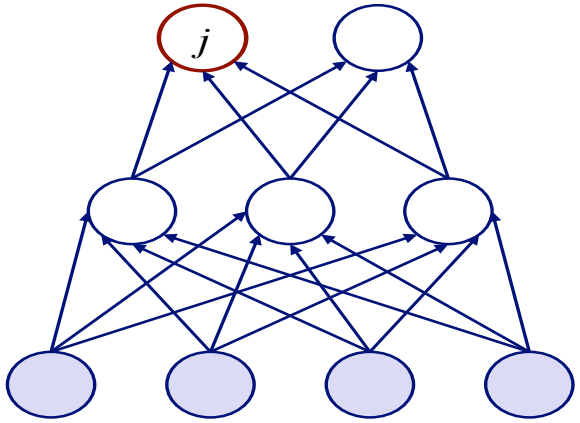
Introduction to “Back propagation”

Review: Gradient Descent



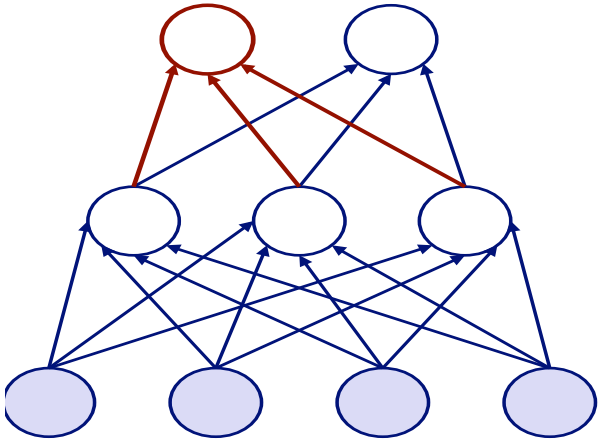
- Popular algorithm for optimization.
- Start with a guess
  - Move on the negative slope
  - Until we reach minima

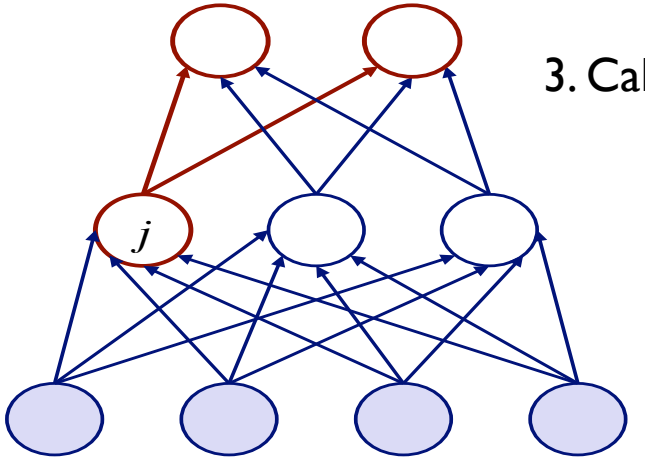
Error Back propagation



1. Calculate error

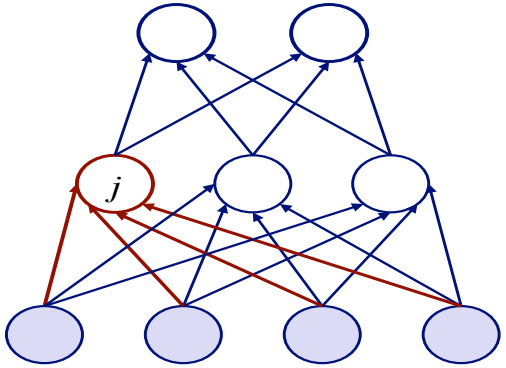
2. Determine updates for weights going to outputs.



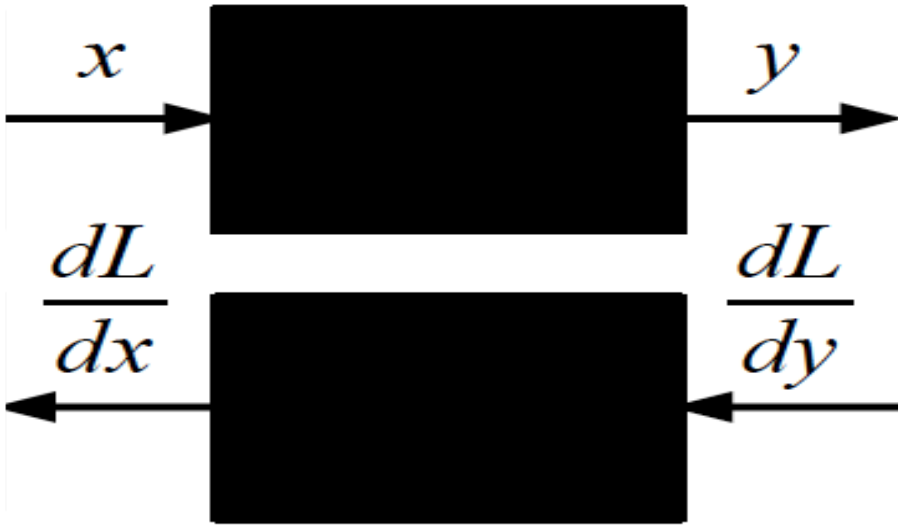


3. Calculate error for hidden units

4. Determine updates for weights to hidden units using hidden-unit errors



Chain Rule

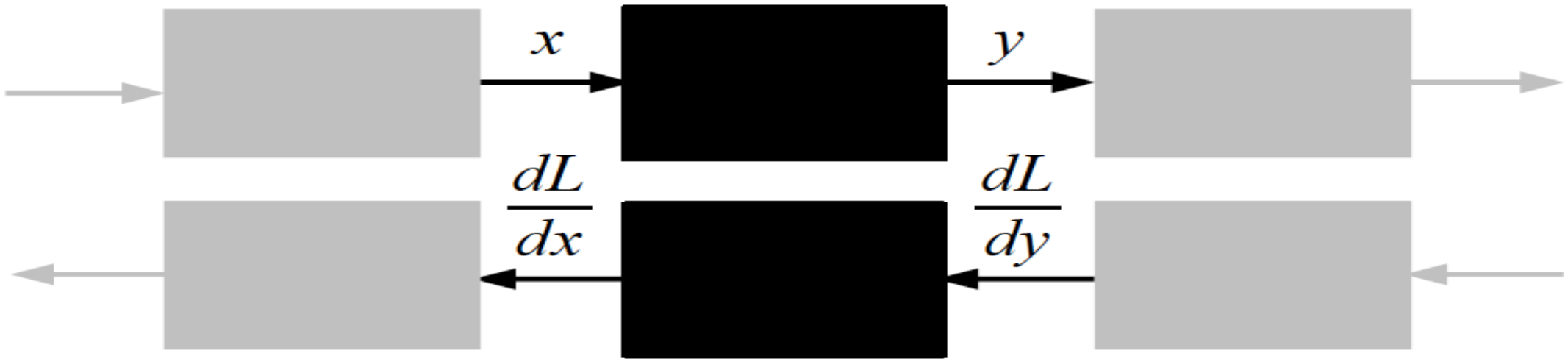


Given $y(x)$ and dL/dy,

What is dL/dx ?

$$\frac{dL}{dx} = \frac{dL}{dy} \cdot \frac{dy}{dx}$$

Chain Rule



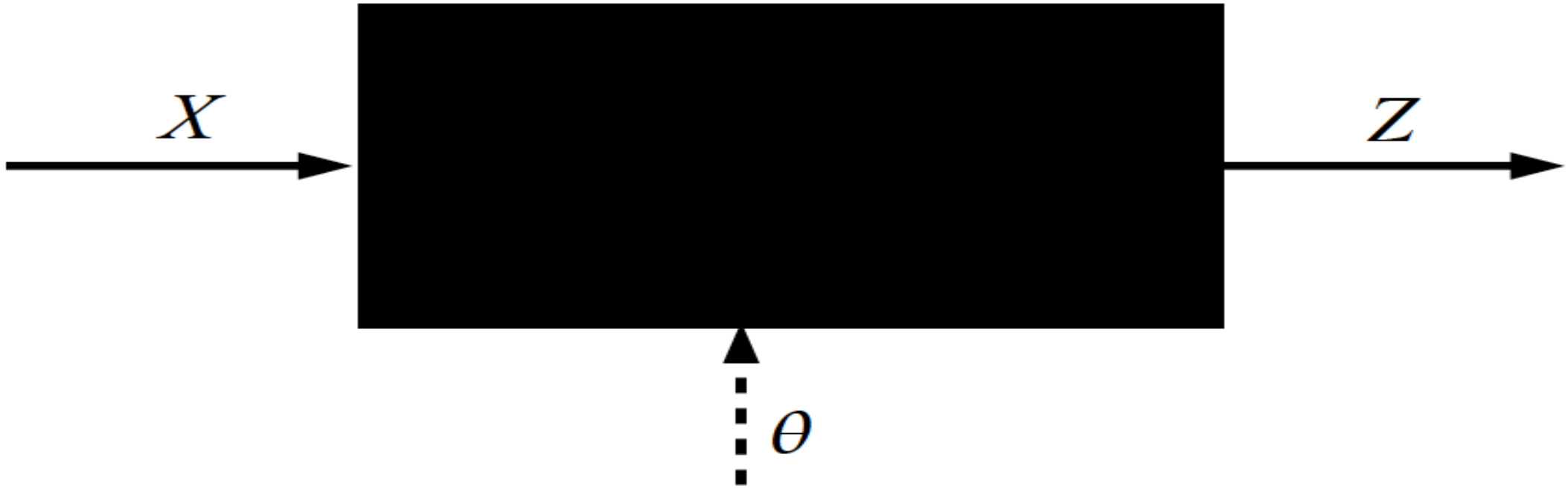
Given $y(x)$ and dL/dy,

What is dL/dx ?

$$\frac{dL}{dx} = \frac{dL}{dy} \cdot \frac{dy}{dx}$$

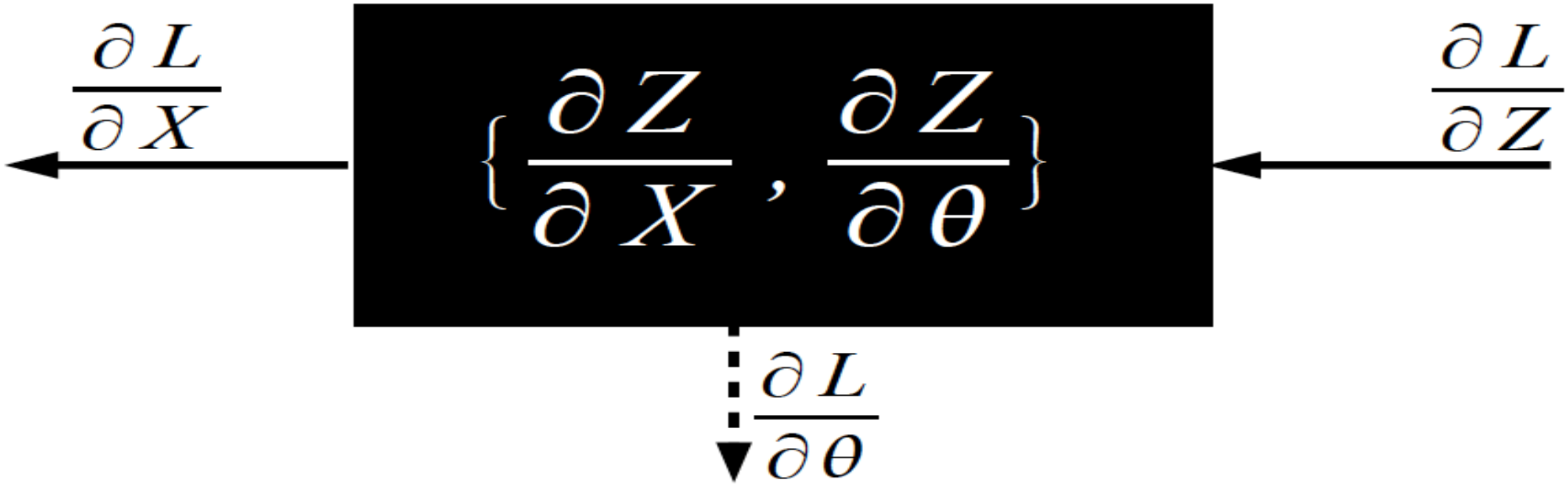
For each block/parameters, we only need to find $\frac{\partial y}{\partial x}$ or $\frac{\partial y}{\partial \theta}$.

Key Computation: Forward-Prop



Theta is the parameters, say the weights within the box.

Key Computation: Back-Prop



$$\theta \leftarrow \theta - \eta \frac{dL}{d\theta}$$

Neural Network Training

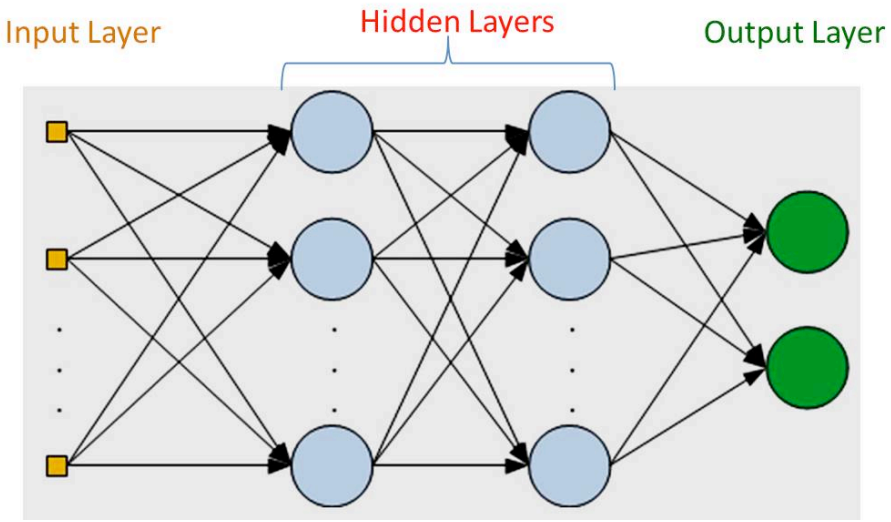
- Step 1: Compute Loss on mini-batch [F-Pass]

Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]
- Step 2: Compute gradients wrt parameters [B-Pass]

$$\theta \leftarrow \theta - \eta \frac{dL}{d\theta}$$

BP for MLP



In either case, we can compute $\frac{\partial y}{\partial x}$ easily.

Two Computational Blocks/Steps

$$\mathbf{y} = \mathbf{W}\mathbf{x}$$

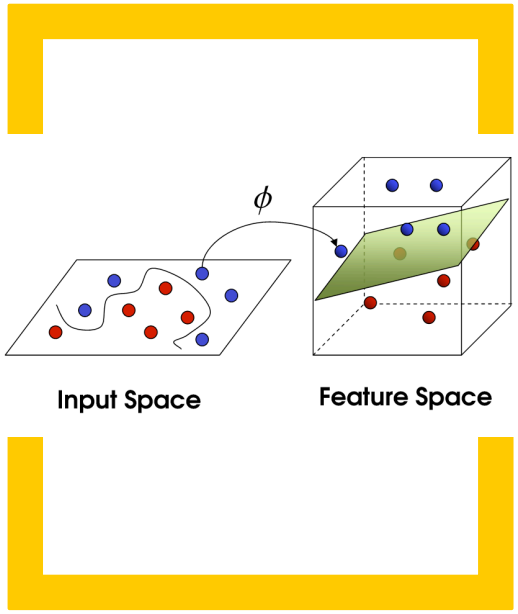
$$\mathbf{y} = \phi(x) = \frac{1}{1 + e^{-x}}$$

Summary: BP

- Step 0: Initialize the Network (MLP), weights
- Step 1:
  - Do forward pass for a batch of randomly selected samples.
  - Predict outputs with the existing weights.
- Step 2:
  - Compute Loss for the set of samples.
- Step 3:
  - Update all the weights using gradient descent. $\theta \leftarrow \theta - \eta \frac{dL}{d\theta}$

Practical Tricks in BP (more later)

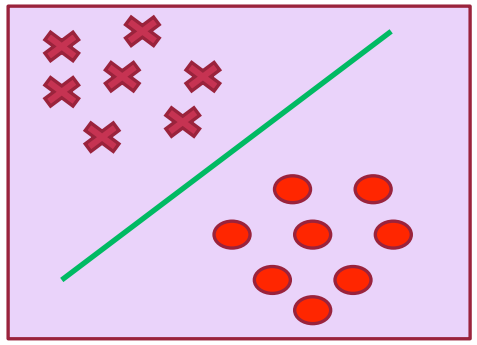
- Control learning rate
  - Usually small learning rate.
  - Vary learning rate over time.
- Remember the past and adjust the speed
  - Eg. Momentum (borrow ideas from physics!!)
- Good initialization
  - Not random. Not zero. Not equal.



Nonlinear Feature Maps and Linear Algorithms

Learning techniques

- Linear classifier

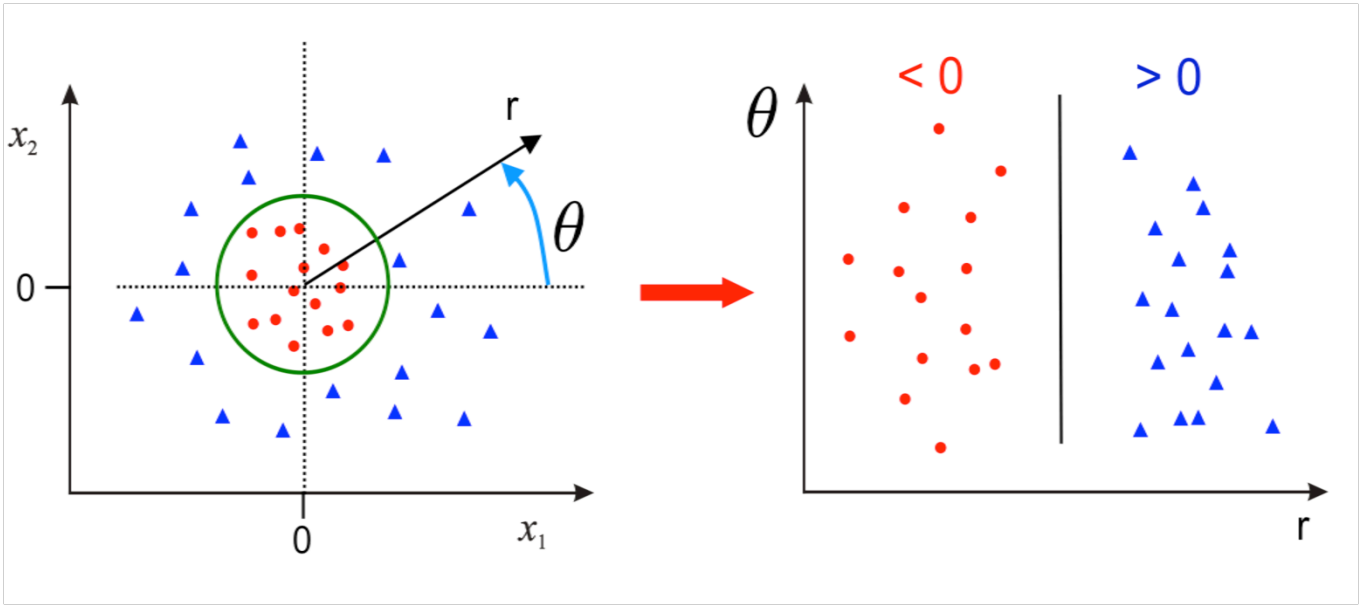


$$g(x_n) = sign(w^T x_n)$$

where *w* is an *d*-dim vector (learned)

- Techniques:
  - Perceptron
  - Logistic regression
  - Support vector machine (SVM)
  - Etc.

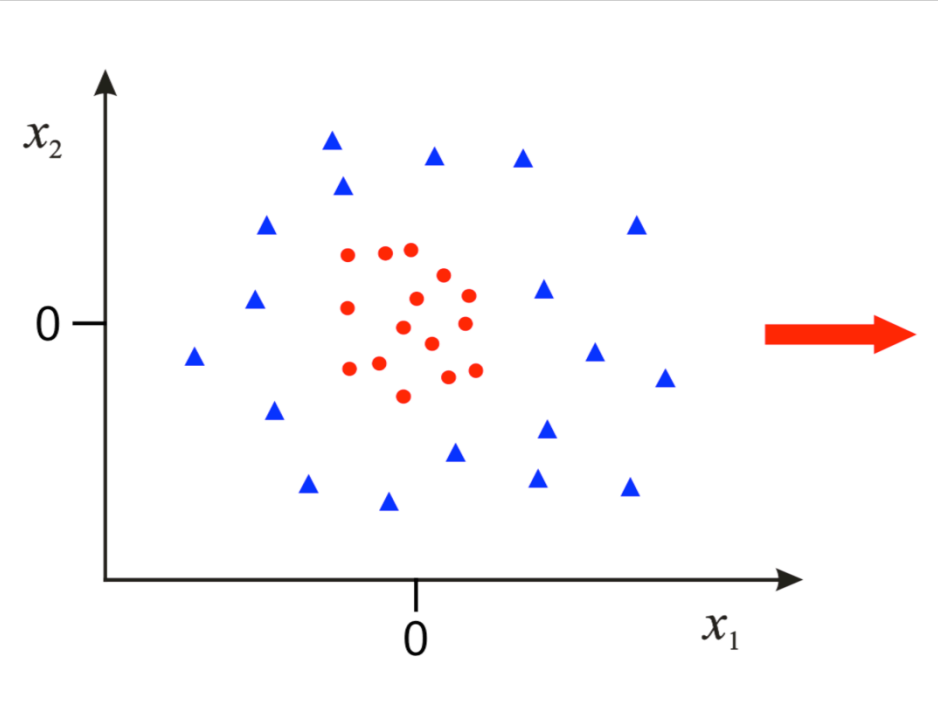
Nonlinearity with Feature Maps



$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r \\ \theta \end{pmatrix}$$

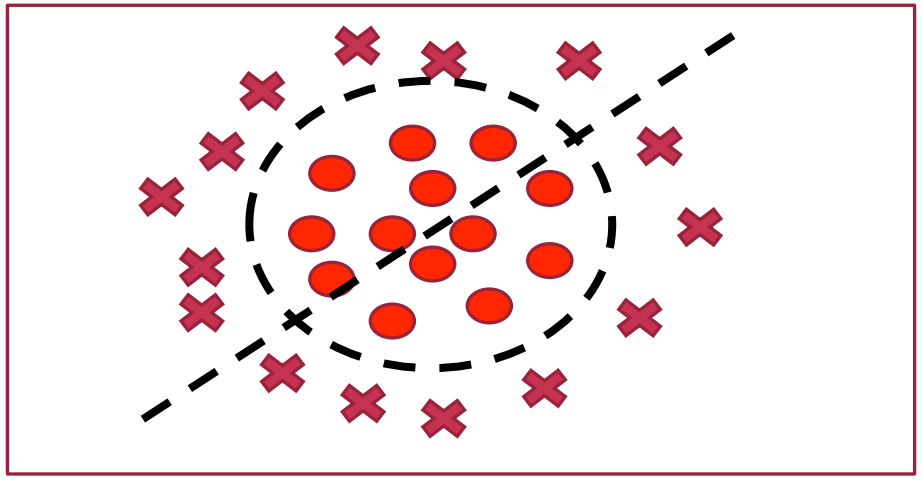
With a "smart" feature map, a linearly non-separable problem can be converted to a separable problem.!!

Q: Transform to (x_1^2, x_2^2)



More general

- Non-linear case



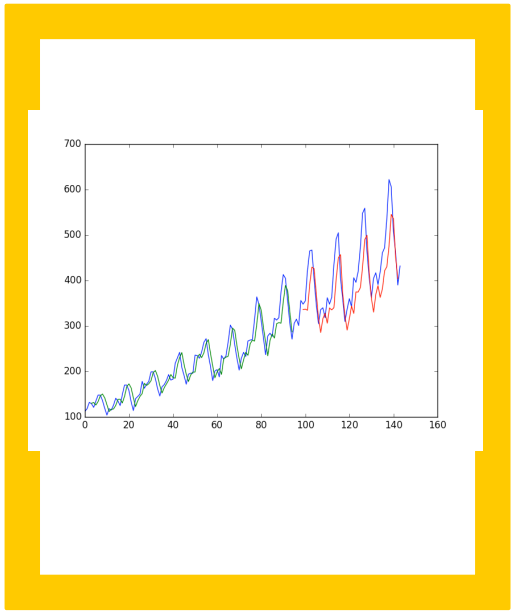
$$x_n = [x_{n1}, x_{n2}]$$

$$x_n = [x_{n1}, x_{n2}, x_{n1} * x_{n2}, {x_{n1}}^2, {x_{n2}}^2]$$

$$g(x_n) = sign(w^T x_n)$$

Summary

- If features can be transformed appropriately, simple linear algorithms (classification, regression) is enough.
- How do we find the feature transformation?
  - Make a reasonable guess?
  - Ans: Use some popular complex function.
- Why linear?
  - Nice algorithms.
  - Speed of evaluation!!



Predicting in Time

Time Series

- Time series is a sequence of observations often ordered in time.
- Problem: Given a sequence, predict future samples.
- Applications:
  - Meteorology, Finance, Marketing etc.
- Notation: x[0], x[1], x[2], …, x[N].
- X[t]. Where t is the time or index in the sequence.

A Simple Model

- X[t] = a1 X[t-1] + a2 X[t-2] + a3 X[t-3] + n
  - Where n is noise.
- Problem:
  - Given the sequence X[0], X[1], ... X[N]
  - Find coefficients a1, a2, a3
- Find the coefficients a1,a2,a3 such that prediction error is minimal.

Model the problem

- Our familiar (x,y) may be seen as:
  - (<x[0],x[1],x[2]>, x[3])
  - (<x[1],x[2],x[3]>, x[4])
- Problem is now cast in a 3D space
  - Regression.

$$\text{Minimize}_{a1,a2,a3} \sum_{t=3}^{N} (x[t] - a1 \cdot x[t-1] - a2 \cdot x[t-2] - a3 \cdot x[t-3])^2$$

More Powerful Model

- X[t] = f(W, X[t-1], X[t-2], X[t-3]) + n
- Problem:
  - Given the sequence X[0], X[1], ... X[N]
  - Find coefficients W
- Data may be modeled as in the above linear case.
- f() may be seen as a MLP

$$\text{Minimize}_W \sum_{t=3}^{N} (x[t] - f(W, x[t-1], x[t-2], x[t-3]))^2$$

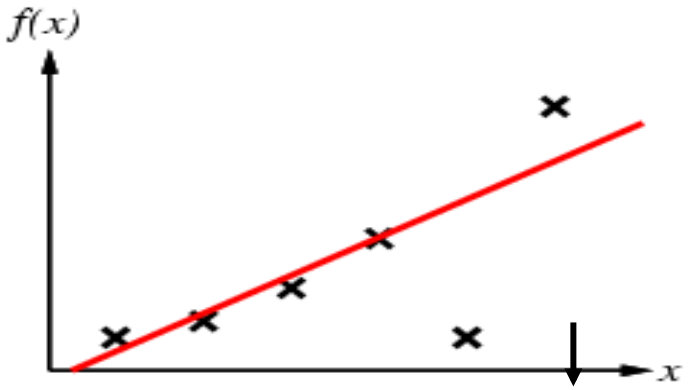
Summary

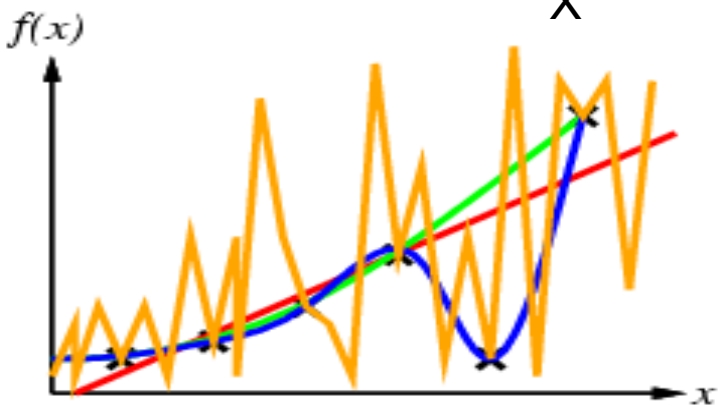
- Predicting future samples is a new problem
- However, solution is similar to what we know.
  - Cast as regression.
- Model can be linear
  - Linear regression
- Or nonlinear
  - MLP
- On how many past samples, the future sample will depend?
  - Order/model to be guessed?

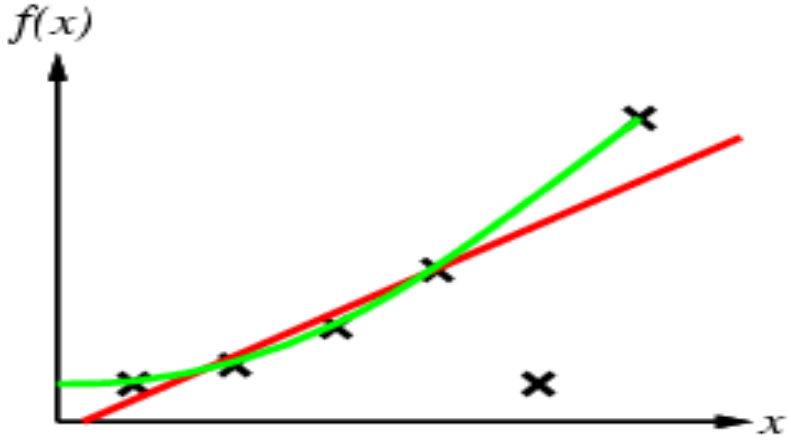
How nonlinear/complex the solution?

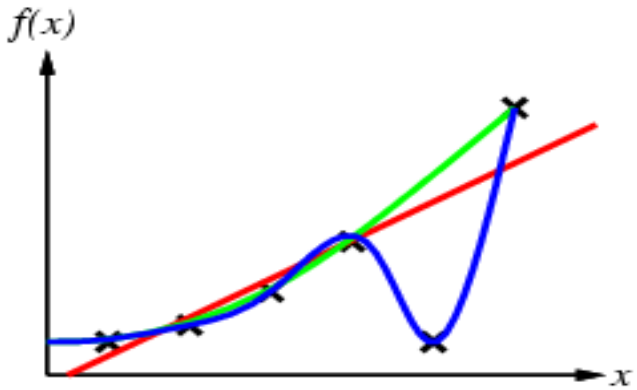
Basic issues in ML

How Nonlinear be the model?









Given six "training" (x_i, y_i) pairs, find the y corresponding to the new "test" x*

Which curve is the best?

Occam's Razor

Select the simplest hypothesis (solution) that suits the data.

Eg. Minimize Sum of "fit error" and "degree of the polynomial"

Thanks. Questions?