

# ECE 5424: Introduction to Machine Learning

Topics:

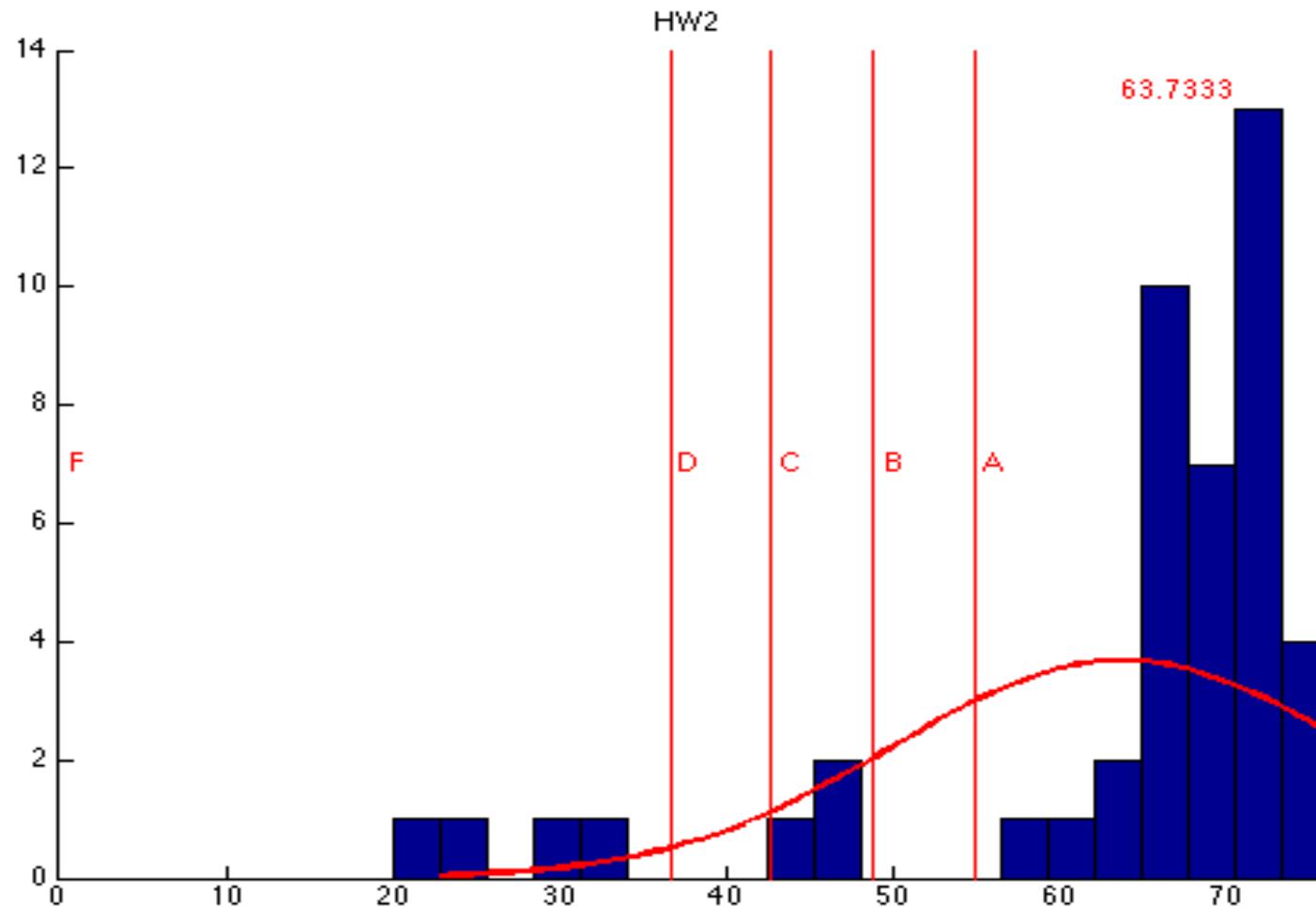
- SVM
  - Multi-class SVMs
- Neural Networks
  - Multi-layer Perceptron

Readings: Barber 17.5, Murphy 16.5

Stefan Lee  
Virginia Tech

# HW2 Graded

- Mean  $63/61 = 103\%$  Max: 76 Min: 20



# Administrivia

- HW3
  - Due: Nov 7<sup>th</sup> 11:55PM
  - You will implement primal & dual SVMs
  - Kaggle competition: Higgs Boson Signal vs Background classification

```
function svmModel = trainSVMprimal(traindata,trainlabels,C)
ntrain = size(traindata,1);
ndim = size(traindata,2);

% construct H
H =
;

% construct f
f =
;

% construct A
A =
;
A =
;
A =
;

% construct b
b =
;

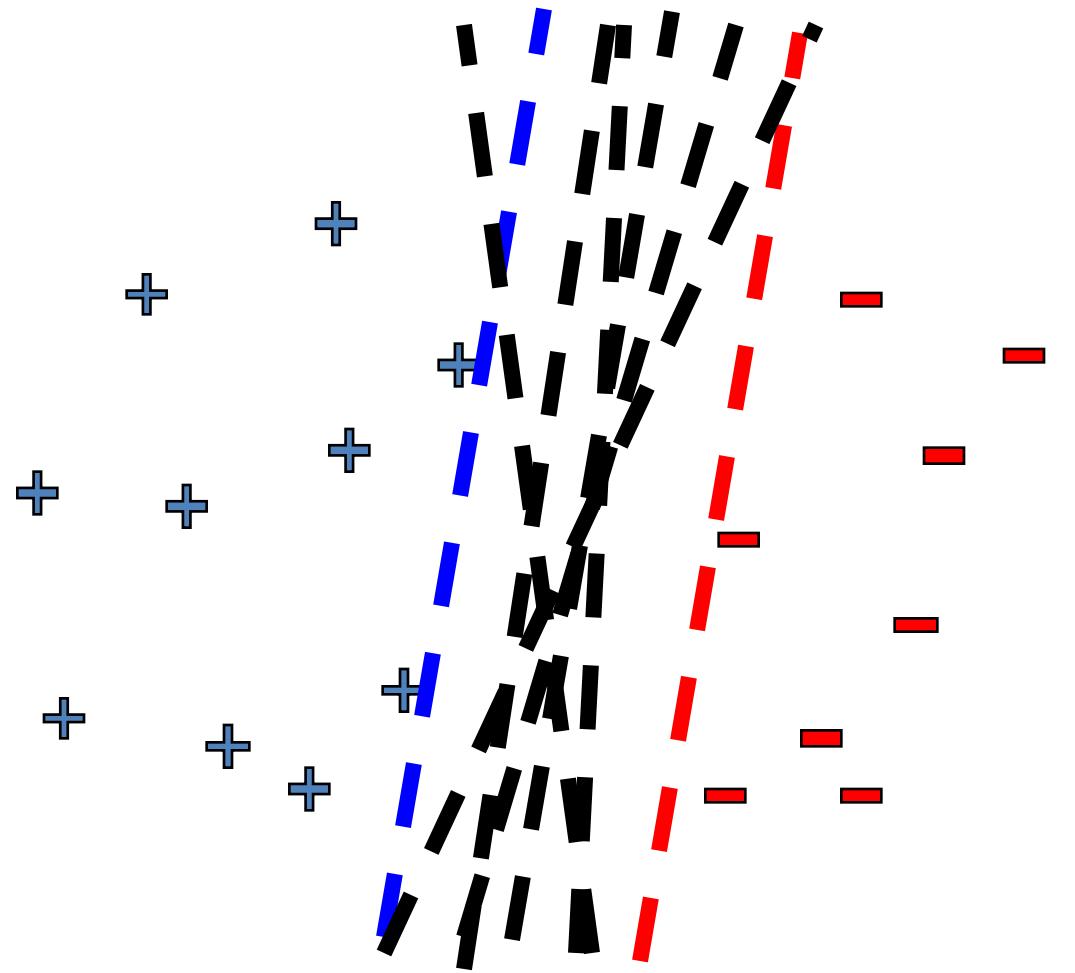
% constuct lb
lb =
;

% solve QP
tic
%z = quadprog(H,f,A,b,[],[],lb);
[z err lm] = qpas(H,f,A,b,[],[],lb);
toc

svmModel.w =
svmModel.w0 =
```

# Recap of Last Time

# Linear classifiers – Which line is better?



$$\mathbf{w} \cdot \mathbf{x} = \sum_j w^{(j)} x^{(j)}$$

# Dual SVM derivation (1) – the linearly separable case

$$\begin{aligned} & \text{minimize}_{\mathbf{w}, b} \quad \frac{1}{2} \mathbf{w} \cdot \mathbf{w} \\ & (\mathbf{w} \cdot \mathbf{x}_j + b) y_j \geq 1, \quad \forall j \end{aligned}$$

# Dual SVM derivation (1) – the linearly separable case

$$L(\mathbf{w}, \alpha) = \frac{1}{2} \mathbf{w} \cdot \mathbf{w} - \sum_j \alpha_j [(\mathbf{w} \cdot \mathbf{x}_j + b) y_j - 1]$$
$$\alpha_j \geq 0, \quad \forall j$$

$$\mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

# Dual SVM formulation – the linearly separable case

$$\text{maximize}_{\alpha} \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

$$\begin{aligned}\sum_i \alpha_i y_i &= 0 \\ \alpha_i &\geq 0\end{aligned}$$

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$b = y_k - \mathbf{w} \cdot \mathbf{x}_k$$

for any  $k$  where  $\alpha_k > 0$

# Dual SVM formulation – the non-separable case

$$\begin{aligned} \text{minimize}_{\mathbf{w}, b} \quad & \frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_j \xi_j \\ (\mathbf{w} \cdot \mathbf{x}_j + b) y_j \geq & 1 - \xi_j, \quad \forall j \\ \xi_j \geq & 0, \quad \forall j \end{aligned}$$

# Dual SVM formulation – the non-separable case

$$\text{maximize}_{\alpha} \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

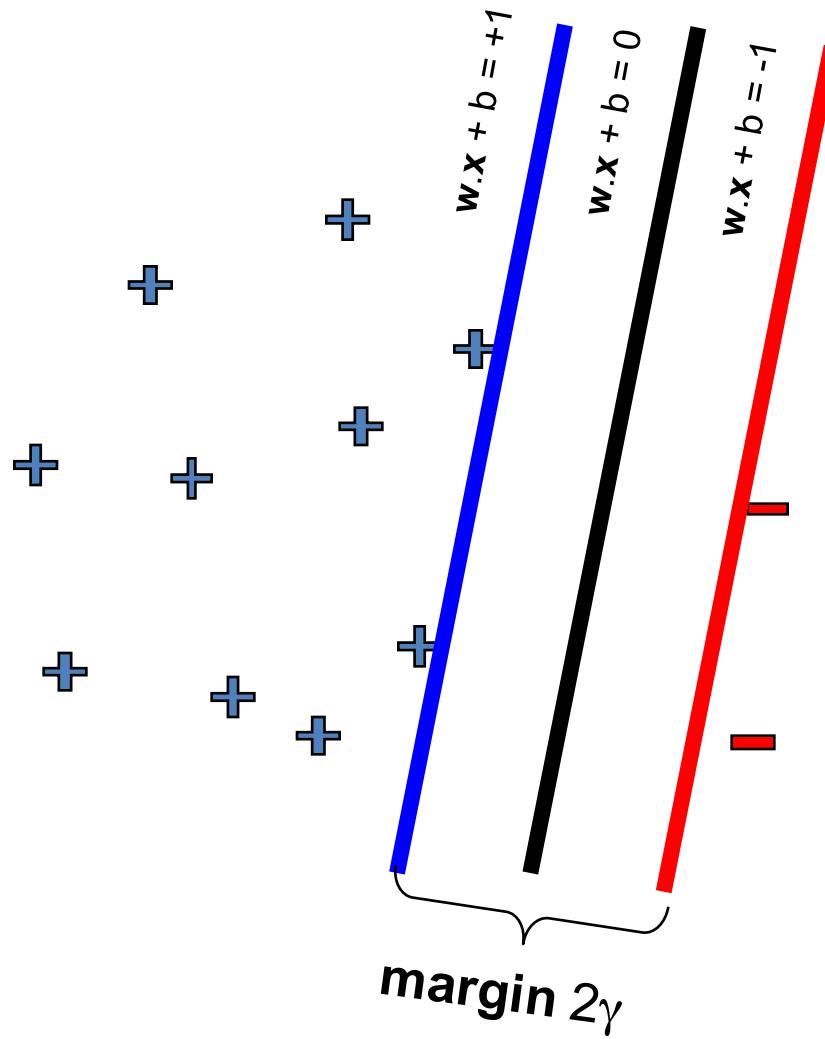
$$b = y_k - \mathbf{w} \cdot \mathbf{x}_k$$

for any  $k$  where  $C > \alpha_k > 0$

# Why did we learn about the dual SVM?

- Builds character!
- Exposes structure about the problem
- There are some quadratic programming algorithms that can solve the dual faster than the primal
- The “**kernel trick**”!!!

# Dual SVM interpretation: Sparsity



$$\mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

# Dual formulation only depends on dot-products, not on $\mathbf{w}$ !

$$\text{maximize}_{\alpha} \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

$$\text{maximize}_{\alpha} \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

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

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

# Common kernels

- Polynomials of degree d

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d$$

- Polynomials of degree up to d

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^d$$

- Gaussian kernel / Radial Basis Function

$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{\|\mathbf{u} - \mathbf{v}\|^2}{2\sigma^2}\right)$$

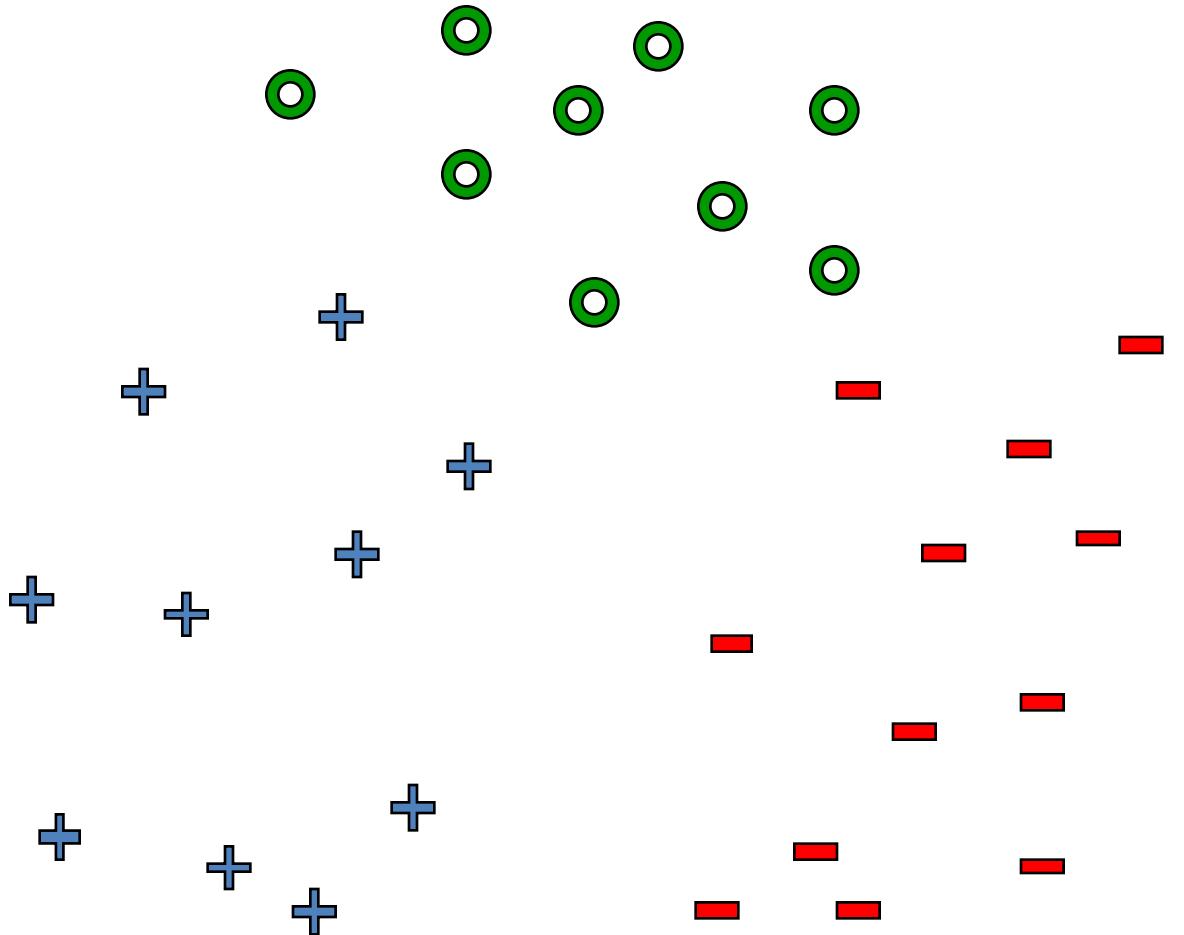
- Sigmoid

$$K(\mathbf{u}, \mathbf{v}) = \tanh(\eta \mathbf{u} \cdot \mathbf{v} + \nu)$$

# Plan for Today

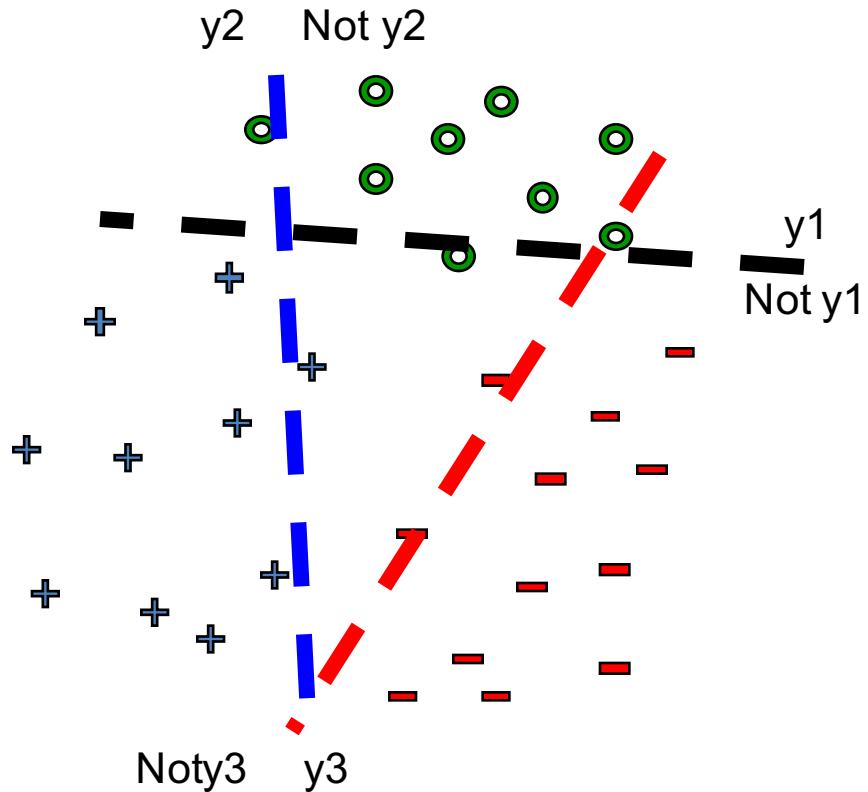
- SVMs
  - Multi-class
- Neural Networks

# What about multiple classes?



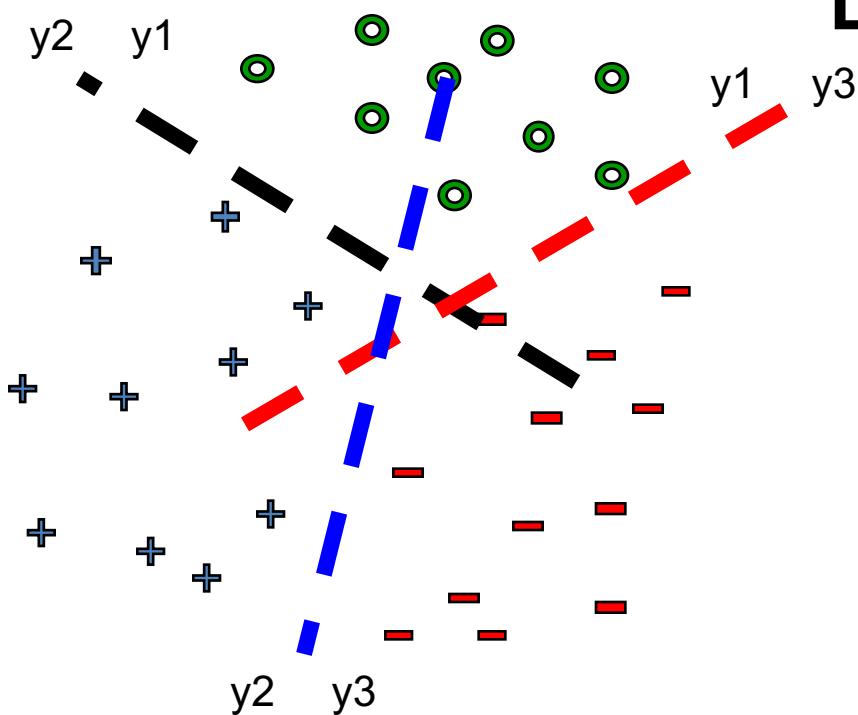
# One against All (Rest)

**Learn N classifiers:**

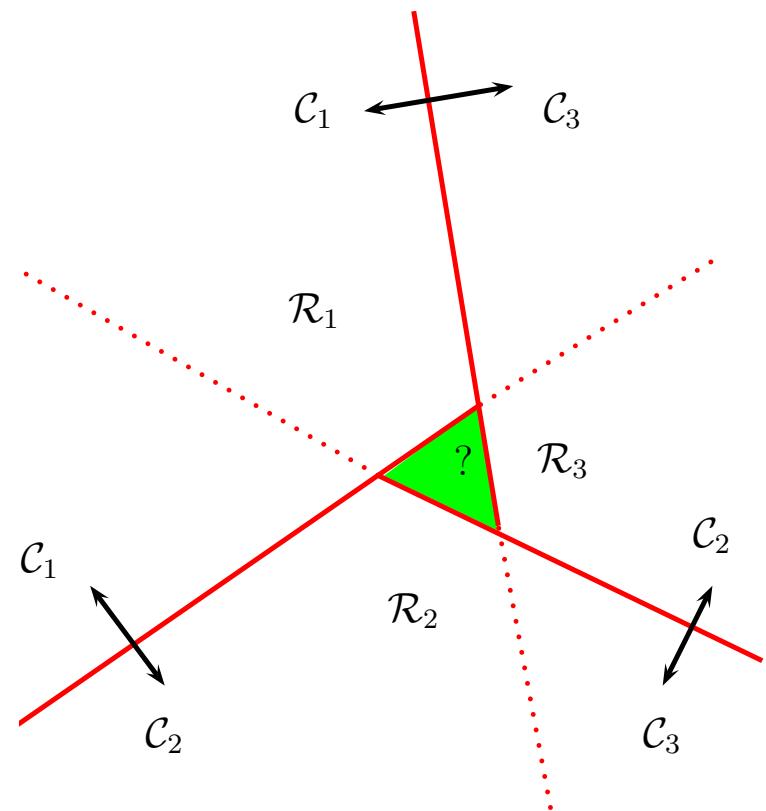
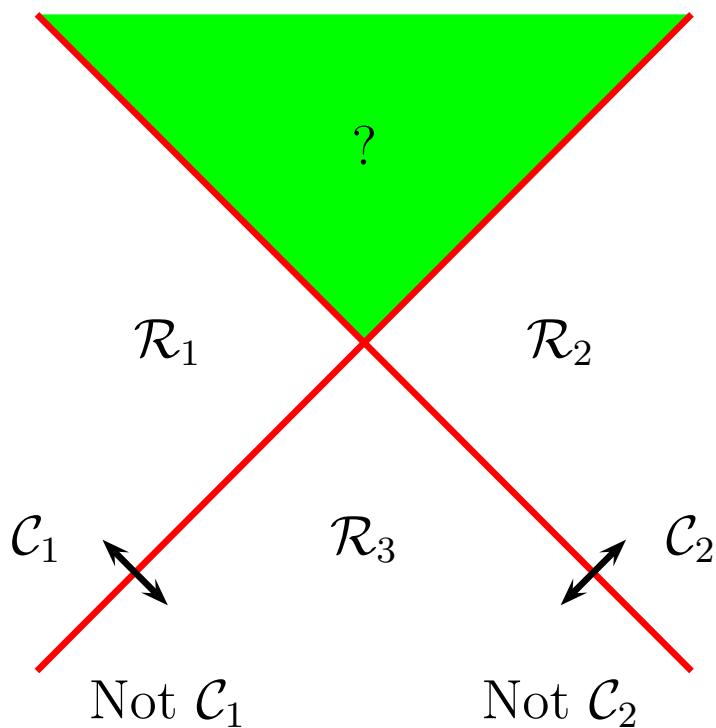


# One against One

**Learn N-choose-2 classifiers:**

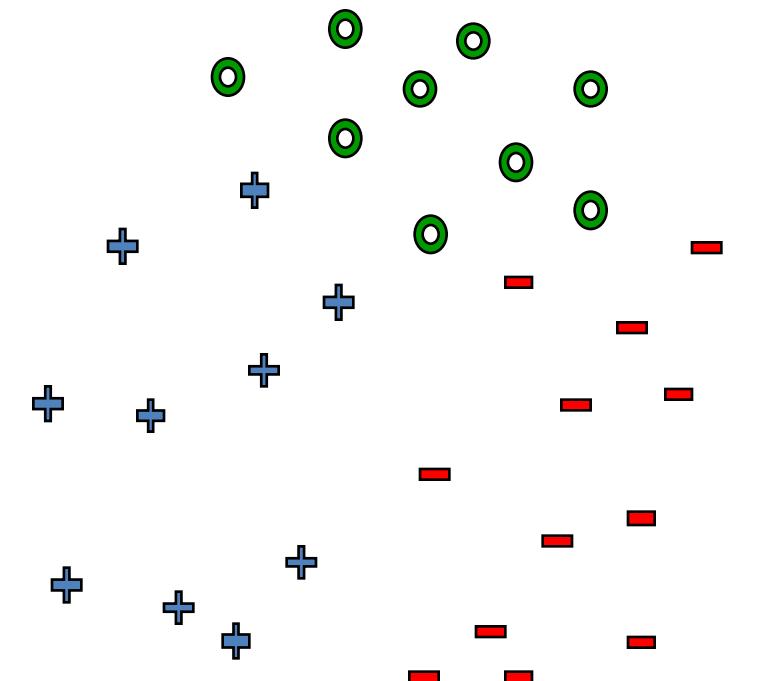


# Problems



# Learn 1 classifier: Multiclass SVM

**Simultaneously learn 3 sets of weights**



$$\mathbf{w}^{(y_j)} \cdot \mathbf{x}_j + b^{(y_j)} \geq \mathbf{w}^{(y')} \cdot \mathbf{x}_j + b^{(y')} + 1, \quad \forall y' \neq y_j, \quad \forall j$$

# Learn 1 classifier: Multiclass SVM

$$\begin{aligned} & \text{minimize}_{\mathbf{w}, b} \sum_y \mathbf{w}^{(y)} \cdot \mathbf{w}^{(y)} + C \sum_j \xi_j \\ & \mathbf{w}^{(y_j)} \cdot \mathbf{x}_j + b^{(y_j)} \geq \mathbf{w}^{(y')} \cdot \mathbf{x}_j + b^{(y')} + 1 - \xi_j, \quad \forall y' \neq y_j, \quad \forall j \\ & \xi_j \geq 0, \quad \forall j \end{aligned}$$

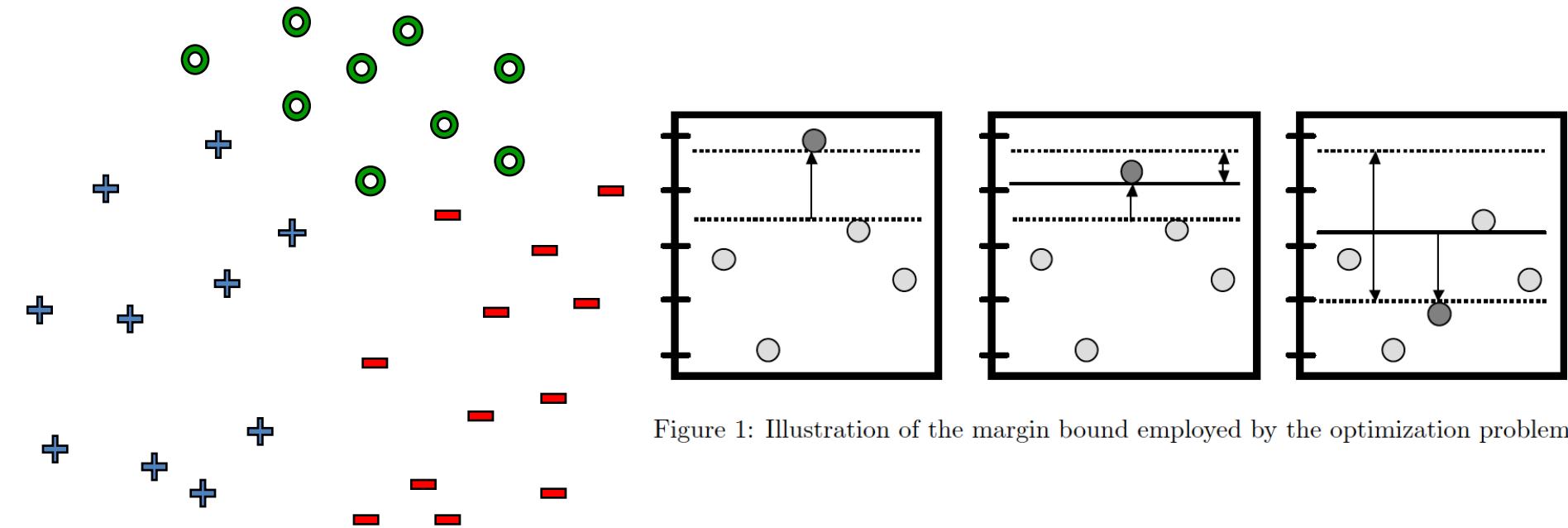


Figure 1: Illustration of the margin bound employed by the optimization problem.

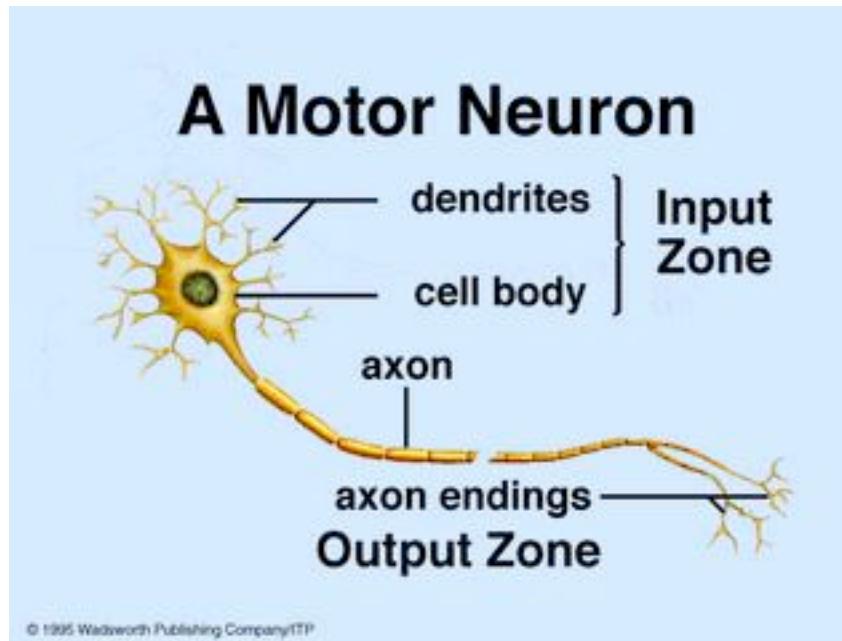
# Addressing non-linearly separable data – Option 1, non-linear features

- Choose non-linear features, e.g.,
  - Typical linear features:  $w_0 + \sum_i w_i x_i$
  - Example of non-linear features:
    - Degree 2 polynomials,  $w_0 + \sum_i w_i x_i + \sum_{ij} w_{ij} x_i x_j$
- Classifier  $h_w(\mathbf{x})$  still linear in parameters  $\mathbf{w}$ 
  - As easy to learn
  - Data is linearly separable in higher dimensional spaces
  - Express via kernels

# Addressing non-linearly separable data – Option 2, non-linear classifier

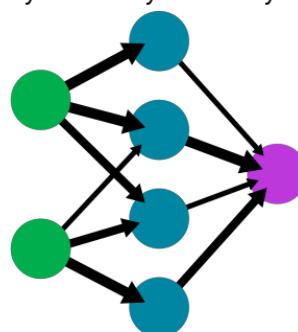
- Choose a classifier  $h_w(\mathbf{x})$  that is non-linear in parameters  $w$ , e.g.,
  - Decision trees, neural networks,...
- More general than linear classifiers
- But, can often be harder to learn (non-convex optimization required)
- Often very useful (outperforms linear classifiers)
- In a way, both ideas are related

# New Topic: Neural Networks



A simple neural network

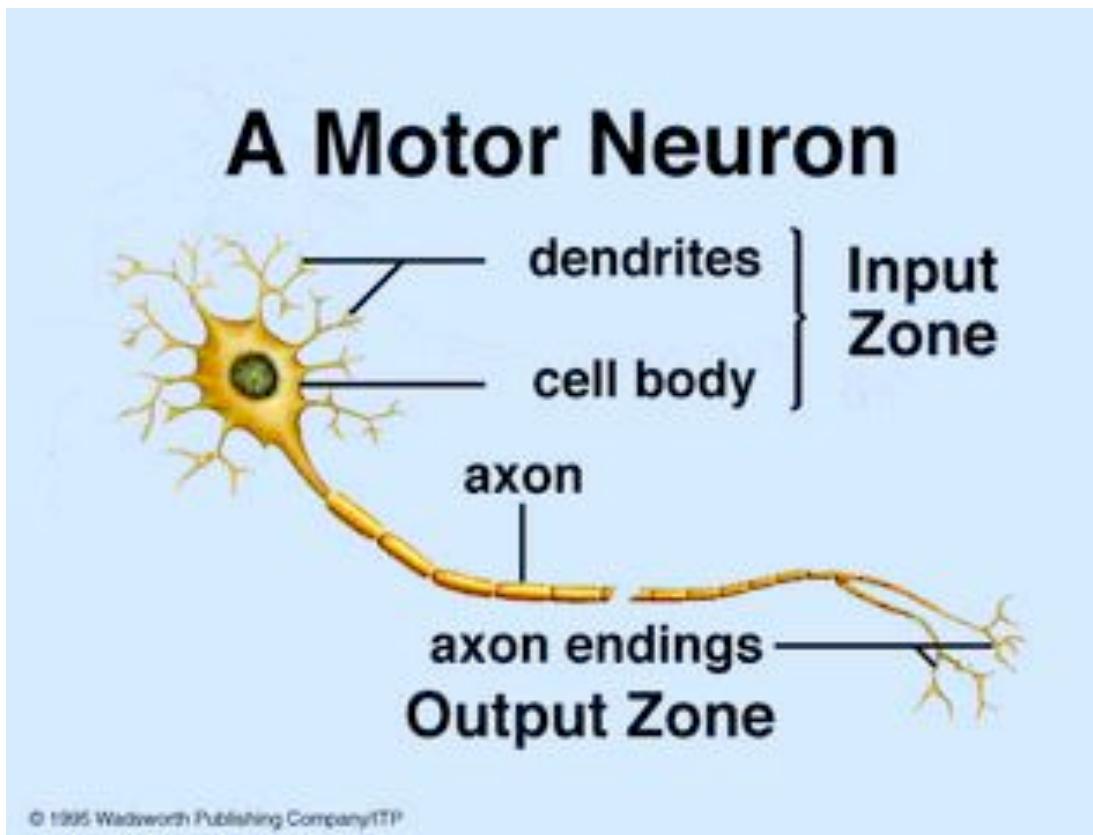
input layer      hidden layer      output layer



# Synonyms

- Neural Networks
- Artificial Neural Network (ANN)
- Feed-forward Networks
- Multilayer Perceptrons (MLP)
- Types of ANN
  - Convolutional Nets
  - Autoencoders
  - Recurrent Neural Nets
- [Back with a new name]: Deep Nets / Deep Learning

# Biological Neuron



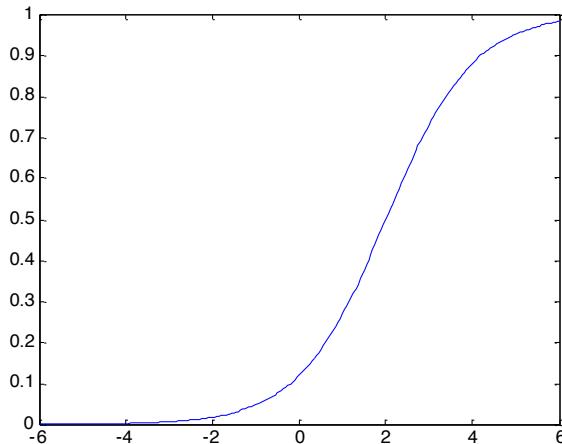
# Artificial “Neuron”

- Perceptron (with step function)
- Logistic Regression (with sigmoid)

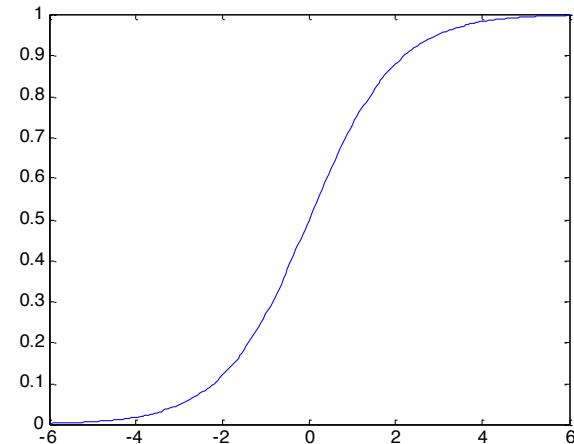
# Sigmoid

$$g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{-(w_0 + \sum_i w_i x_i)}}$$

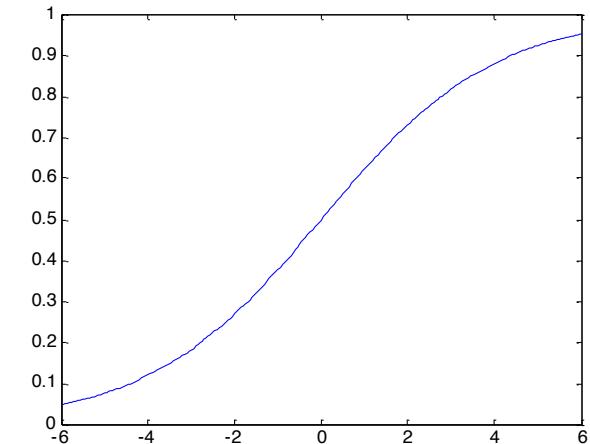
$w_0=2, w_1=1$



$w_0=0, w_1=1$



$w_0=0, w_1=0.5$

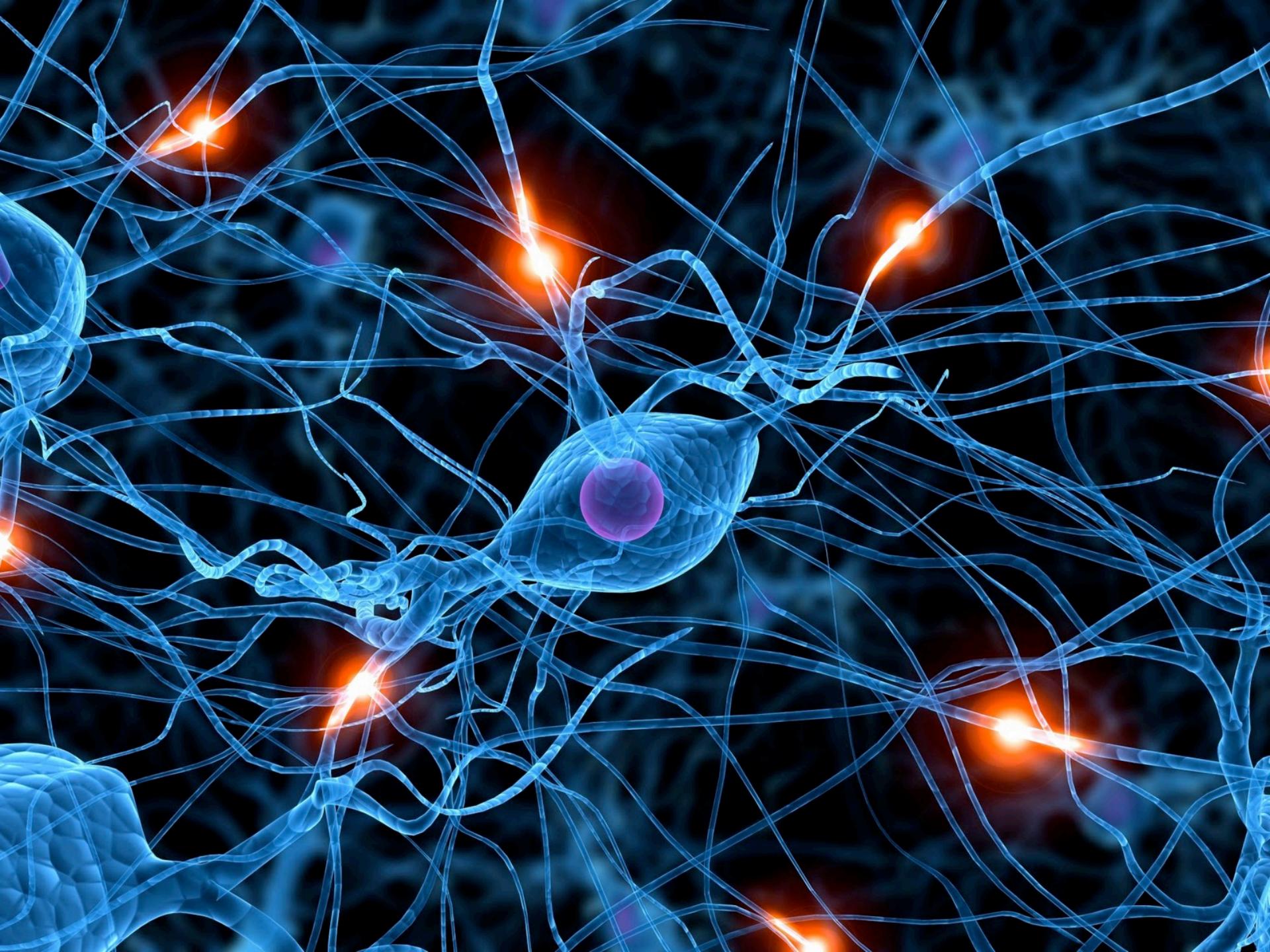


# Many possible response functions

- Linear
- Sigmoid
- Exponential
- Gaussian
- ...

# Limitation

- A single “neuron” is still a linear decision boundary
- What to do?



# Limitation

- A single “neuron” is still a linear decision boundary
- What to do?
- Idea: Stack a bunch of them together!

# Hidden layer

- 1-hidden layer feed-forward network:
  - On board

# Neural Nets

- Best performers on OCR
  - <http://yann.lecun.com/exdb/lenet/index.html>
- NetTalk
  - Text to Speech system from 1987
  - <http://youtu.be/tXMaFhO6dIY?t=45m15s>
- Rick Rashid speaks Mandarin
  - <http://youtu.be/Nu-nlQqFCKg?t=7m30s>

# Universal Function Approximators

- Theorem
  - 3-layer network with linear outputs can uniformly approximate any continuous function to arbitrary accuracy, given enough hidden units [Funahashi '89]

# Neural Networks

- Demo
  - <http://playground.tensorflow.org/>