Natural Language Processing

Lecture 4: *multiclass Classification and Introduction to Neural Nets*

Dan Goldwasser

The 10000 feet view

- **Text classification**: useful formulation of NLP tasks
  - Break text interpretation into a set of categories
- **Learning** : optimize an objective function over the training data.
- **Linear** classification vs. **Non-Linear** classification.
- So far, we've talked about linear models (LR, NB, etc.)
  - Simple (=convex) but require feature engineering effort (**why**?)
- **Questions for this lecture:**
  - Do we need to move beyond linear classifiers?
  - How can we learn representation from data, rather manual engineer them?

Learning as Optimization

- Given an objective function, find an optimal solution.
  - **Objective function**: score that correlated with correct prediction.
  - **Minimization** (correlating with misclassification) or **maximization** problem (probability of correct answer/likelihood of training data).

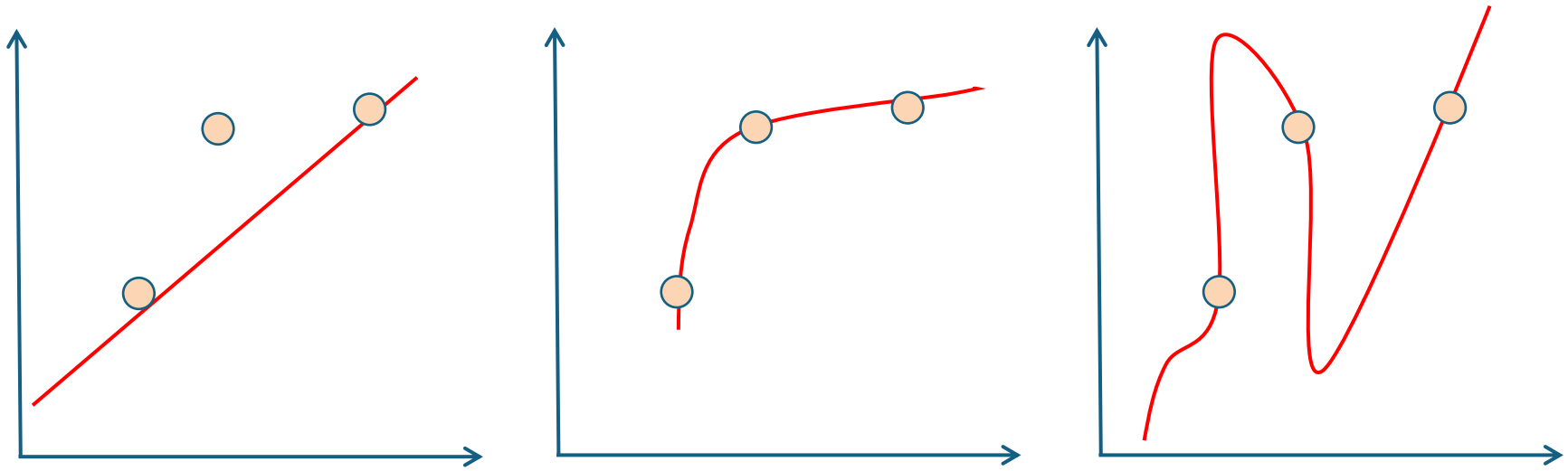
$$p(y_1, y_2, \ldots, y_N \mid x_1, x_2, \ldots, x_N) = \prod_{i=1}^{N} \sigma(w^T \Phi(x_i))^{y_i} \left(1 - \sigma(w^T \Phi(x_i))\right)^{1 - y_i}$$

  - Instead use the **log-likelihood**

$$\log p(y_1, y_2, \ldots, y_N \mid x_1, x_2, \ldots, x_N) = \sum_{i=1}^{N} y_i \log(\sigma(w^T \Phi(x_i))) + (1 - y_i) \log\left(1 - \sigma(w^T \Phi(x_i))\right)$$

Regularization, Regression example

- GD: general optimization algorithm
  - Works for classification (different loss function)
  - Incorporate polynomial features to fit a complex model
  - **Danger – overfitting!**



Regularizationn

- Add a term to the objective function that reduces complexity.
- L1 Regularization

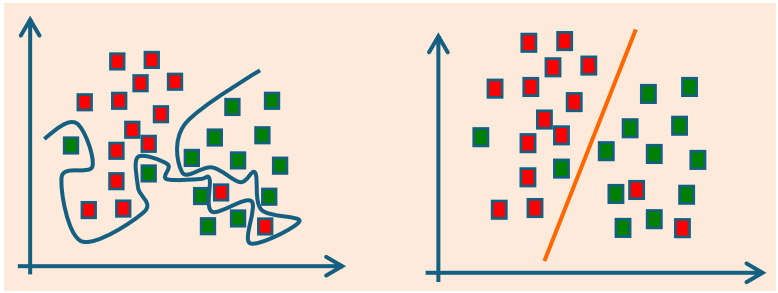
$$L(w) = \ell(w) + \lambda \sum_{i=1}^{n} |w_i|$$

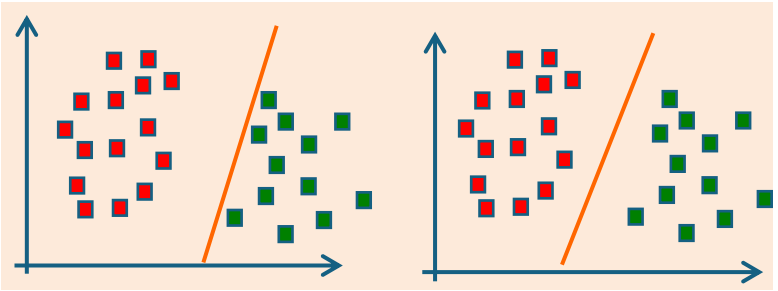
- L2 Regularization

$$L(w) = \ell(w) + \lambda \sum_{i=1}^{n} w_i^2$$

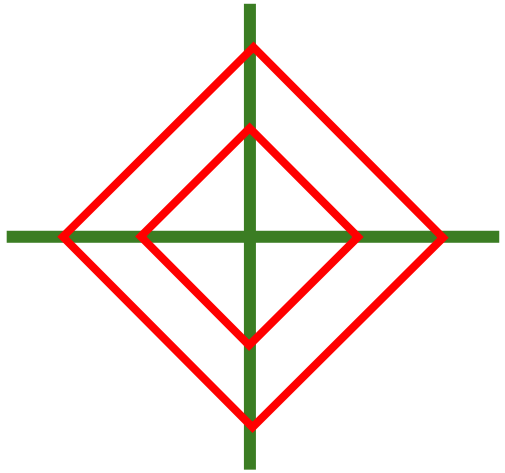
- **Why does minimizing |W| help?**

Would you rather?

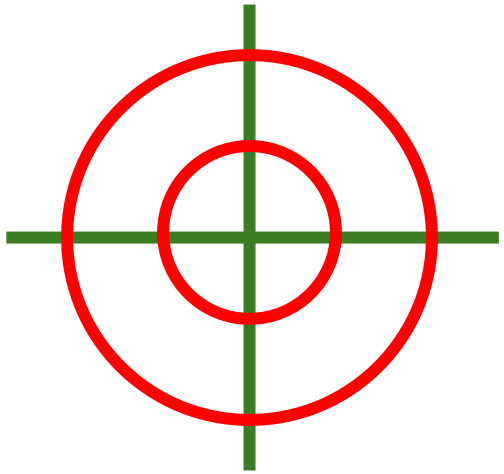




L1 and L2 Norms



$$\|x\|_1 = \sum |x_i|$$



$$\|x\|_2 = \sqrt{\sum x_i^2}$$

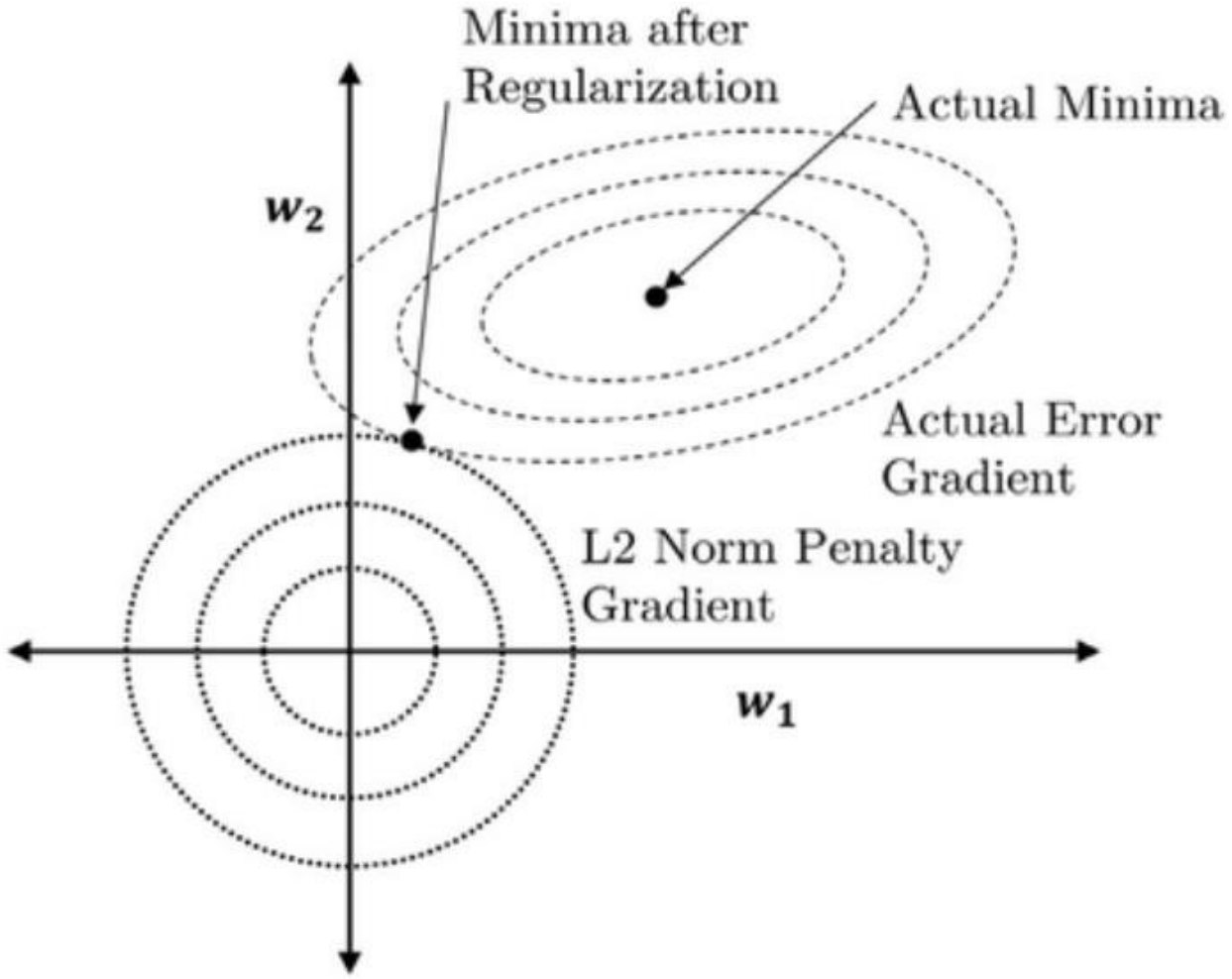
What happens when we minimize the joint objective function?

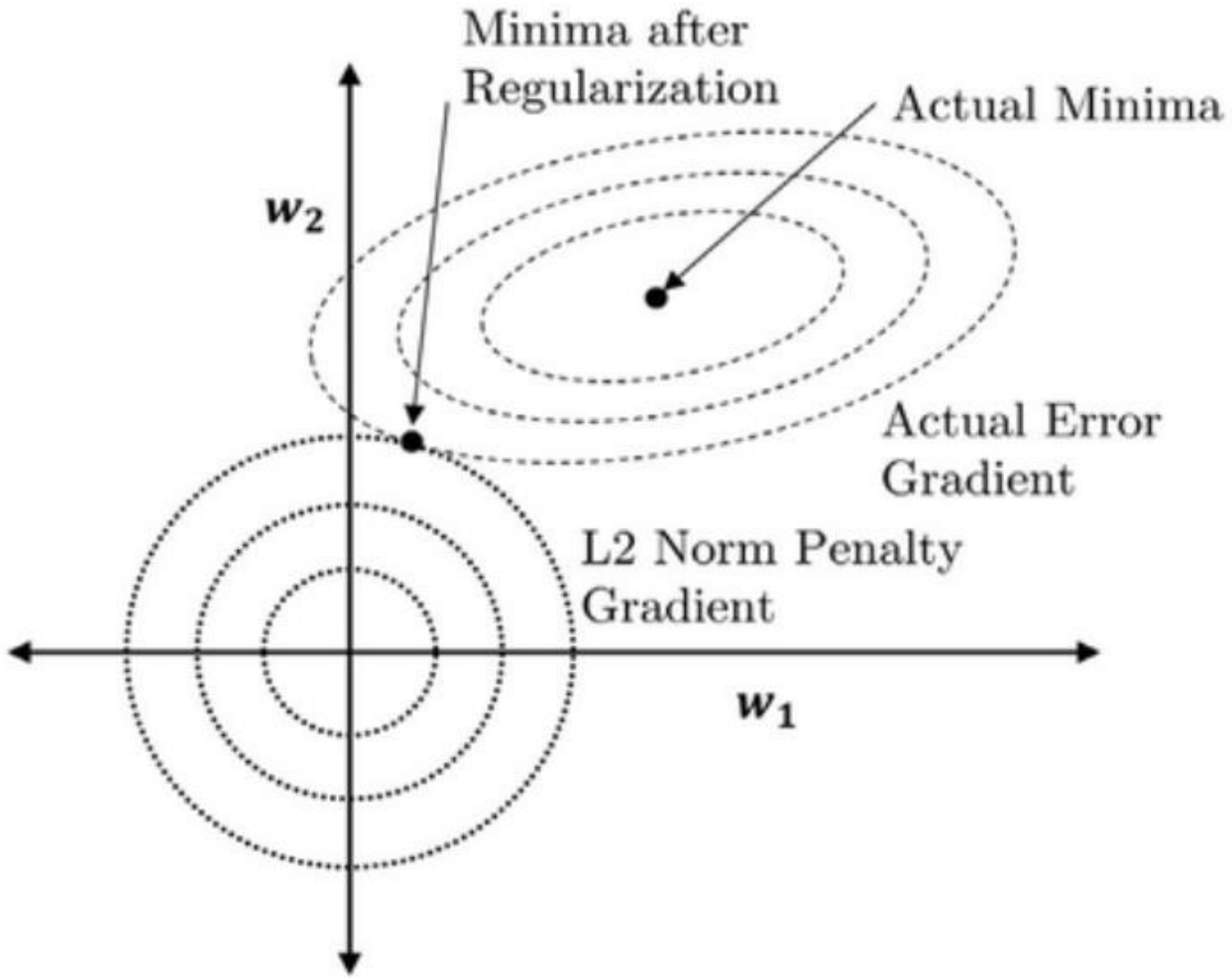
$$\text{Argmin}_w \ell(w) + \lambda R(w)$$

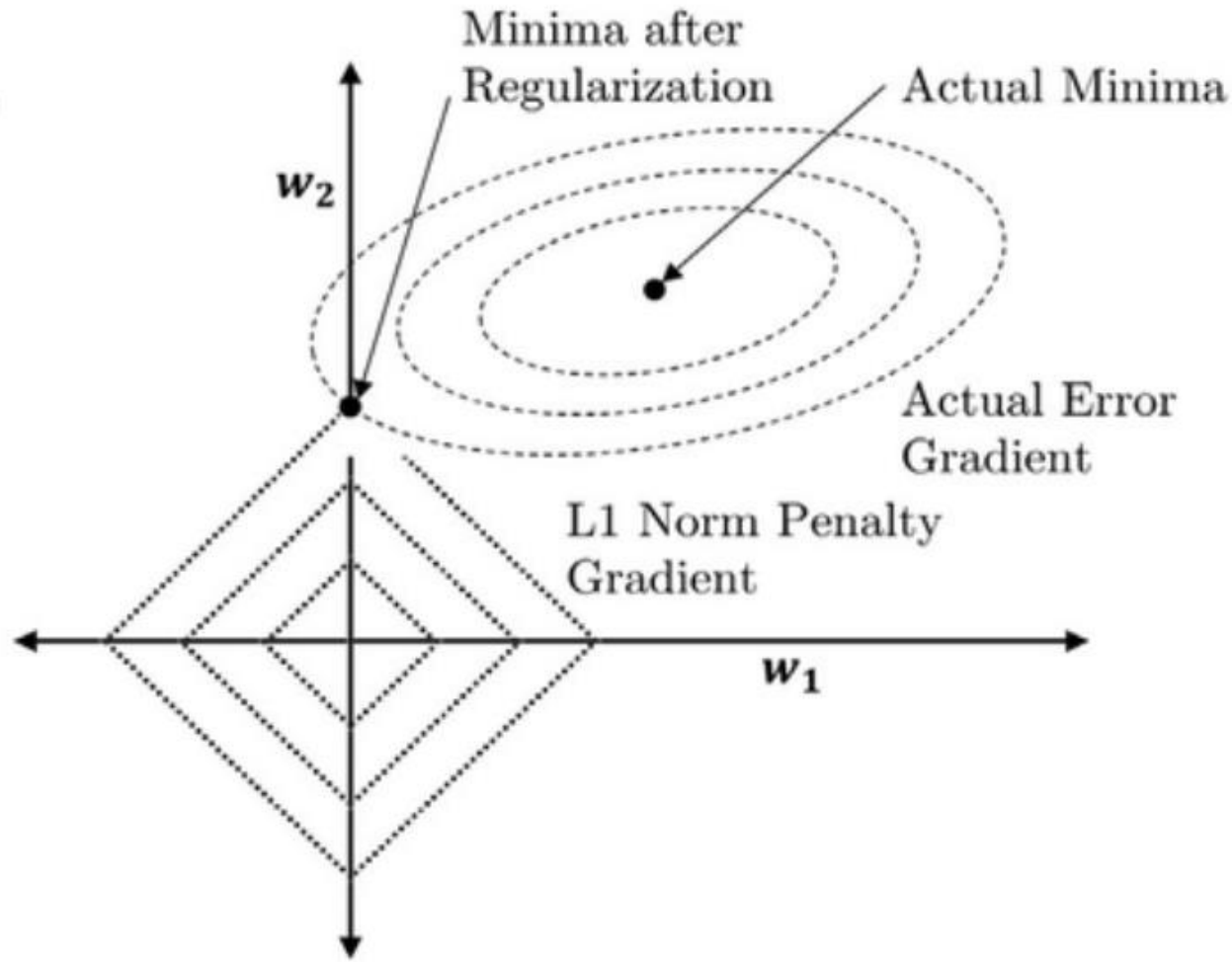
We look at the combined objective: a **weighted sum** of a loss *term* and a *regularizer.*

Each one pushing in a *different* direction.

Given different values of lambda, the optimal point will shift.




Minima after
Regularization
Actual Minima
w_2
Actual Error
Gradient
L2 Norm Penalty
Gradient
w_1

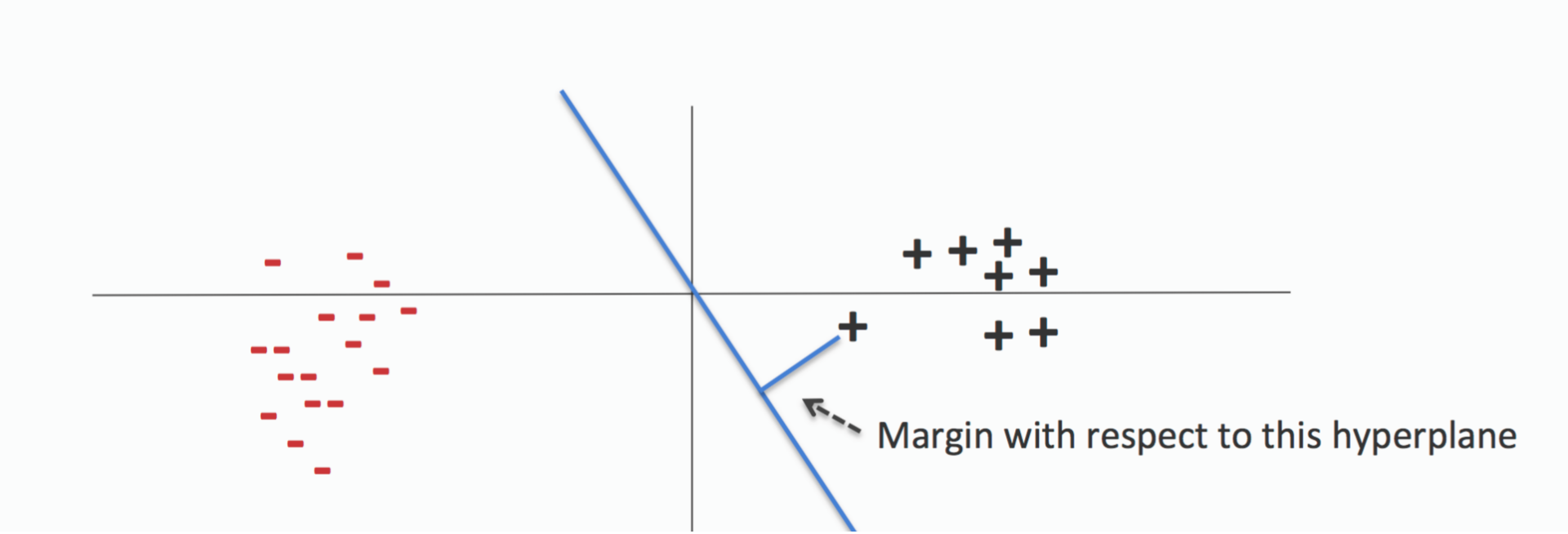

Minima after
Regularization
Actual Minima
w_2
Actual Error
Gradient
L1 Norm Penalty
Gradient
w_1

Reminder: Margin

- The margin of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.

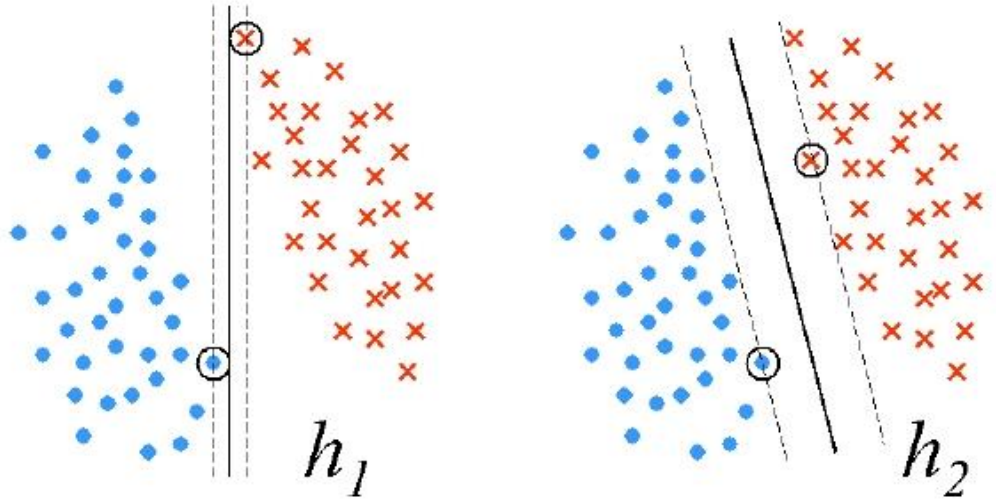
Distance: $\frac{|\mathbf{w}\mathbf{x}+b|}{\|\mathbf{w}\|}$

To help the discussion, let's consider the signed distance: **(why?)** $\frac{(\mathbf{w}\mathbf{x}_i+b)y_i}{\|\mathbf{w}\|}$



Maximal Margin Classifier

- Motivation for *maximal margin*
  - *We'll introduce an optimization problem for maximizing the margin directly*
  - We will then generalize the idea of margin into a broad preference towards simpler models



Hard SVM Intuition

The margin of a classifier: *the distance of the nearest point.* ***Recall:*** $\frac{y\,(w^T x)}{||w||}$

We want to find the max margin classifier: $argmax_w \left[\frac{y\,(w^T x)}{||w||} \right]$

If we **fix** $||w|| = 1$, we can focus on maximizing the **functional margin**

$$w^* = \text{argmax}_{||w||=1} \min_{(x,y) \in S} y\,(w^T x)$$

Or, ***fix*** *the functional margin* $y(w^T x) \geq 1$*, and focus on minimizing* $||w||$

$$w^* = \text{argmin} ||w||$$
$$\text{s.t. } y(w^T x) \geq 1$$

Hard SVM Optimization

- We have shown that the sought after weight vector w is the solution of the following optimization problem:

SVM Optimization:

$$\text{Minimize: } \tfrac{1}{2}\ \|w\|^2$$

$$\text{Subject to: } \forall\ (x,y)\ \forall\ S: \quad y\ w^T x \geq 1$$

- This is an optimization problem in (n+1) variables, with |S|=m inequality constraints.

Non-Separable Case

Want to relax the constraints:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1.$$

Introduce slack variables ξ_i:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \text{Where } \xi_i \geq 0, \text{ an error occurs when } \xi_i > 0$$

Thus we can assign an extra cost for errors, as follows:

$$\begin{aligned} &\text{Minimize} && f(\mathbf{w}, b, \boldsymbol{\xi}) \equiv \tfrac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \xi_i \\ &\text{subject to} && y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i; \quad \xi_i \geq 0, \quad i = 1, \ldots, m \end{aligned}$$

Soft SVM, Unconstrained version

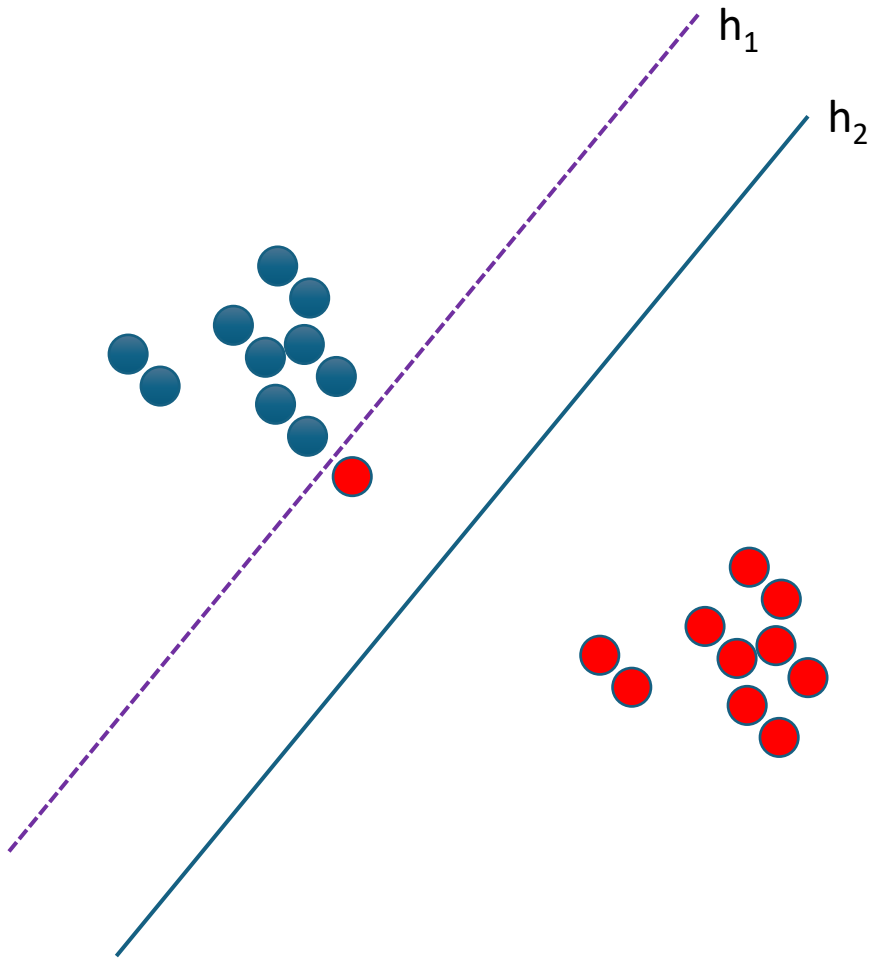
- $$\text{Min } \tfrac{1}{2} \| w \|^2 + c \sum_i \xi_i$$

$$\text{s.t. } \xi_i \geq 0;\ y_i(w_i^T x_i) \geq 1 - \xi_i$$

- The SoftSVM can be written as an unconstrained optimization problem--

$$\text{Min } \tfrac{1}{2} \| w \|^2 + c \sum_i \max(0, 1 - y_i\, w_i^T x_i)$$

Generalization vs Error Min.



Which classifier has a better margin?

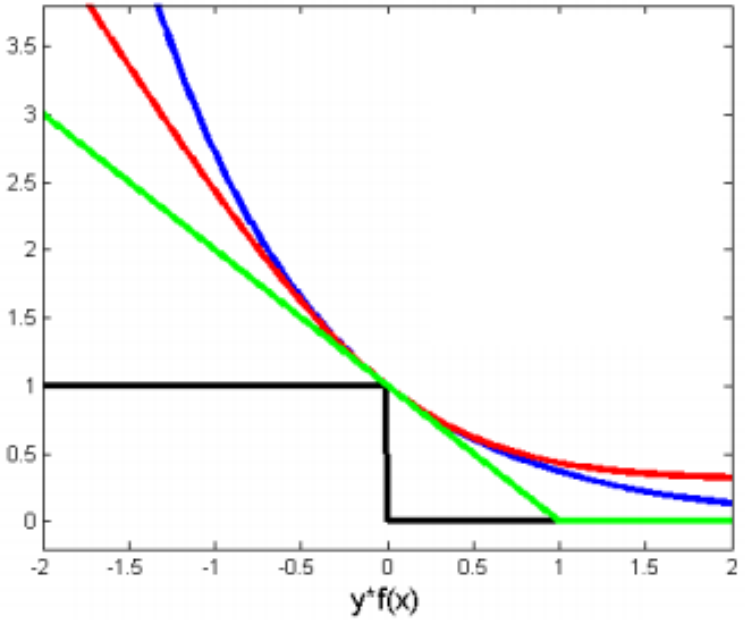
Which one is likely to work better?

How can we encode the preference towards h_1 or h_2 using our objective function?

Surrogate Loss functions

- The **hinge loss** function: $\text{Min}_w \frac{1}{2} \|w\|^2 + c \sum_{i\ (x,y) \in S} \max(0, 1 - y\, w^t x)$
  - Smooth, but not differentiable at x=1
  - Use sub-gradient descent

Surrogate loss function: smooth approximation to the 0-1 loss
Upper bound to 0-1 loss



Multiclass Classification

- So far, our discussion was limited to **binary predictions**

- What happens if our decision is **not** over binary labels?
  - **POS**: Noun,verb, determiner,..
  - **Document classification**: sports, finance, politics
  - **Sentiment**: *Positive, negative, neutral*

- **Most problems have more than just two choices!**

One-vs-All

- **Reduce multiclass classification to multiple binary problems.**

- Given K labels, train K binary classifiers
  - Assuming V feature, how many parameters?

- **Potential problems**:
  - Very imbalanced data for each classifier.
  - No ”global view”, each classifier is trained separately.

- At test time, use all K classifiers and pick the highest scoring label

Training a single classifier

- **Instead** of training K classifiers, each with V parameters, we train a single classifier with KV parameters.

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_K \end{bmatrix}_{nK \times 1} \qquad \phi(\mathbf{x}, i) = \begin{bmatrix} \mathbf{0}_n \\ \vdots \\ \mathbf{x} \\ \vdots \\ \mathbf{0}_n \end{bmatrix}_{nK \times 1}$$

$\mathbf{x}$ in the i^{th} block, zeros everywhere else

- At test time, find the highest scoring label: $\hat{y} = \underset{y}{\operatorname{argmax}}\, \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}, y),$

Example

- The same pattern is encoded as different features associated with different classes.
- The weights capture the relationship between the pattern and the output class.

Var	Definition	Wt
$f_1(0,x)$	1 if "!" $\in$ doc 0 otherwise	−4.5
$f_1(+,x)$	1 if "!" $\in$ doc 0 otherwise	2.6
$f_1(-,x)$	1 if "!" $\in$ doc 0 otherwise	1.3

Training a single classifier

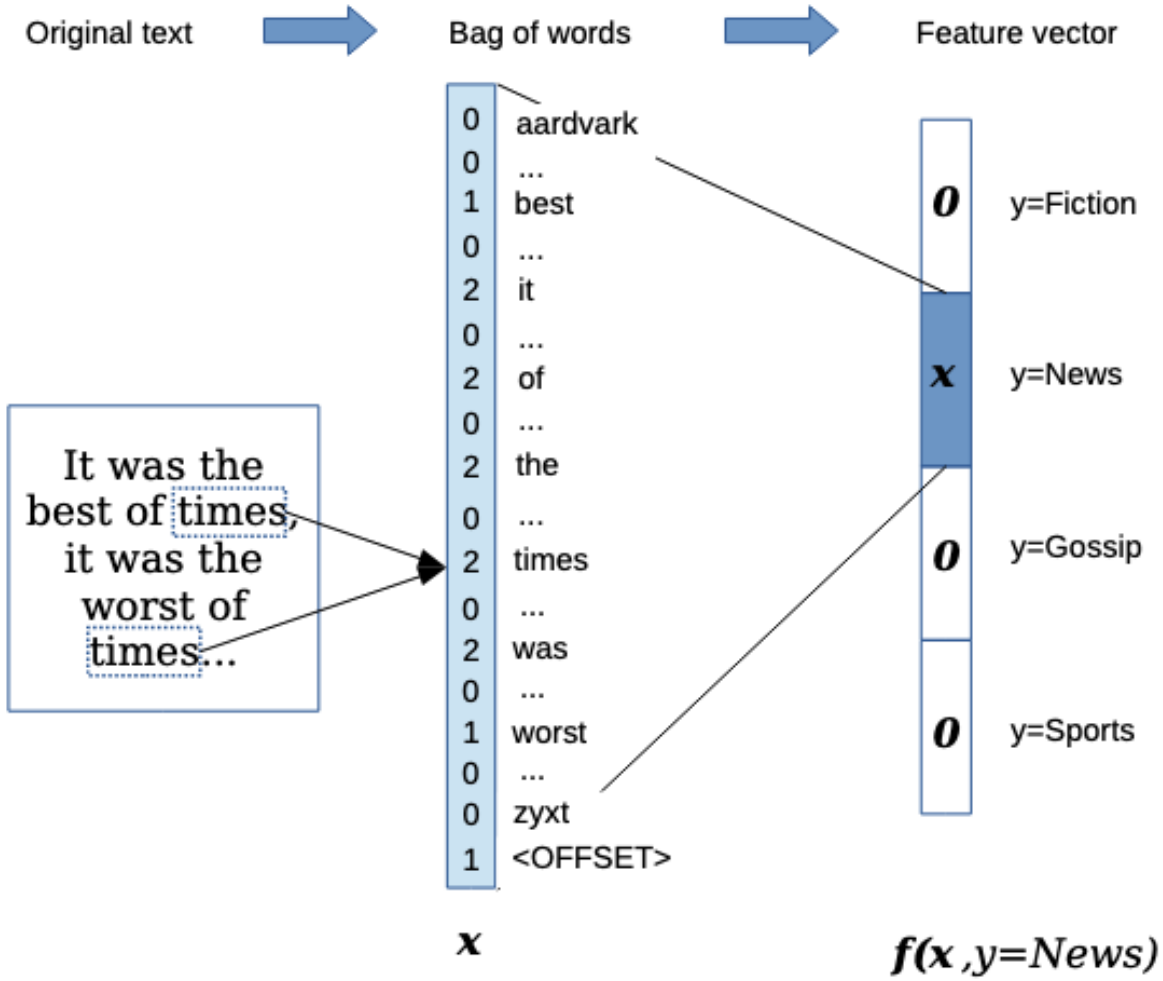


Image from: Jacob Eisenstein: An Introduction To Natural Language Processing

Multiclass Logistic Regression

- The multiclass version can be rewritten as -

$$\mathrm{p}(y \mid \boldsymbol{x}; \boldsymbol{\theta}) = \frac{\exp\left(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}, y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}, y')\right)}.$$

Multiclass Logistic Regression

- The training objective- find w that maximizes the *conditional likelihood* of the data $\{(x,y)_i\}$

$$\log \mathrm{p}(\boldsymbol{y}^{(1:N)} \mid \boldsymbol{x}^{(1:N)}; \boldsymbol{\theta}) = \sum_{i=1}^{N} \log \mathrm{p}(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

$$= \sum_{i=1}^{N} \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y^{(i)}) - \log \sum_{y' \in \mathcal{Y}} \exp\left(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y')\right).$$

$$\ell_{\text{LOGREG}} = -\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y^{(i)}) + \log \sum_{y' \in \mathcal{Y}} \exp\left(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y')\right)$$

Minimize the negative log-likelihood of the data

Multiclass SVM

- **Single classifier optimizing a global objective**
  - *Extend the SVM framework to the multiclass settings*

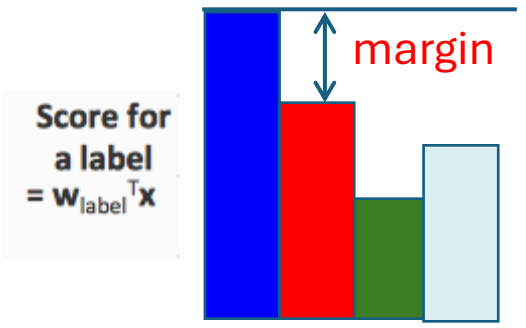
- **Binary SVM:**
  - *Minimize ||W|| such that the closest points to the hyperplane have a score of +/- 1*

- **Multiclass SVM**
  - *Each label has a **different** weight vector*
  - *Maximize **multiclass margin***

Margin in the Multiclass case

Revise the definition for the multiclass case:

- The difference between the score of the correct label and the scores of competing labels



Colors indicate different labels

SVM Objective: Minimize total norm of weights s.t. the ***true label is scored at least 1 more than the second best*.**

Multiclass classification so far

- **Learning**:

Solve: $$\min_{w,\xi} \frac{1}{2}\|w\|^2 + \frac{C}{N}\sum_{n=1}^{N} \xi^n$$

subject to, for $i = 1, \ldots, n$,

$$\langle w, \phi(x^n, y^n)\rangle - \langle w, \phi(x^n, y)\rangle \geq 1 - \xi^n \quad \text{for all } y \in \mathcal{Y} \setminus \{y^n\}.$$

- **Prediction**

$$f(x) = \text{argmax}_{y \in \mathcal{Y}} \ \langle w, \phi(x, y)\rangle$$

Linear vs. Non-Linear Classifiers

- Up to this point we focused on **linear classifiers.**
- They depend on ***engineering*** expressive features which define simple (=convex) learning problems.
- **Simple solutions often break.**
  - BoW is hard to beat, and works great for simple problems
    - Harder cases: nuanced problems, domain differences
    - Tends to blowup the feature space.
- **Non-Linear classifiers** define complex decision boundaries
  - Decision trees, **neural-nets**,..
  - NN dynamically learn a representation over the original feature space;

NN Tradeoffs

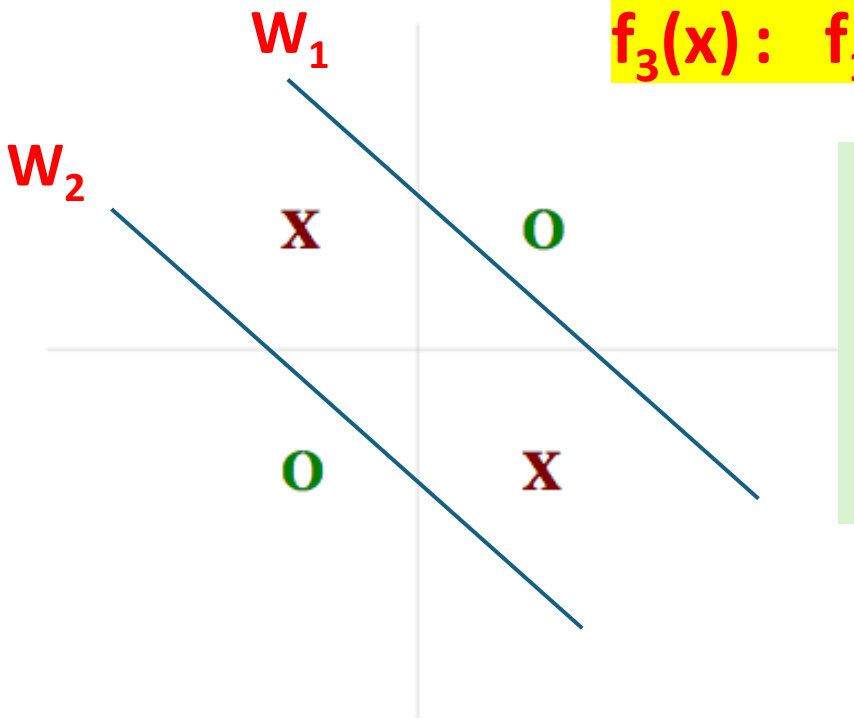
- Our goal is to build an apple-sentiment classifier from text. Here are two examples that the classifier should work well on.

I like apples but not oranges I like oranges but not apples

- Can we use a BoW linear model to tell the two apart?
- What would be the solution? What is the "cost" associated with it?

- NN can be viewed as representation learners, one of the questions we will try to answer is how this representation is constructed and what kind of patterns can/cannot be represented (given different NN architectures).

Limitations of Linear Models



$f_3(x)$: $f_1(x)$ OR $f_2(x)$

Can you find a way to combine linear models to solve this problem?
(i.e., chain their predictions)

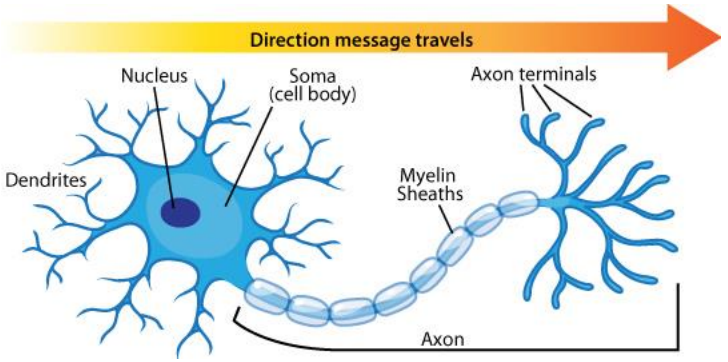
Learn *a model for **representing** the data, that would simplify the problem,* **now build a simple classifier over it.**

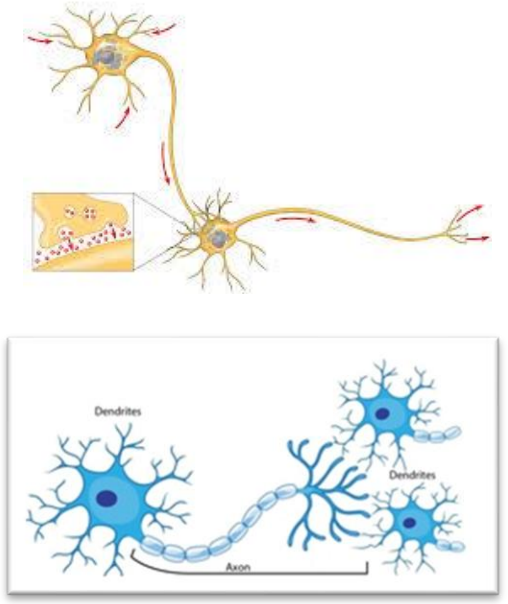
Big idea behind neural net – *jointly learn the representation to make classification easier with the classification problem.*

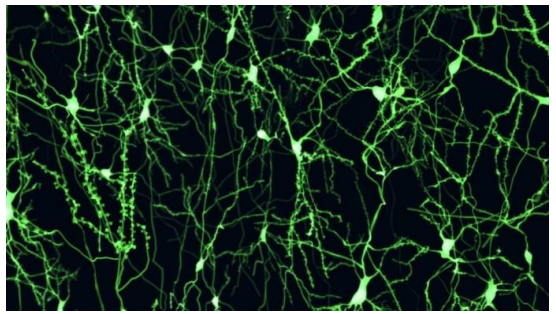
Neurons

Neuron

- takes inputs from many other sources
  - (neurotransmitters into dendrites)
- Each input signal is attenuated by some learned amount
- If the aggregate of these inputs exceeds some threshold, the neuron “fires”, sending out a signal (neurotransmitters from its axon terminals) to all the other neurons connected to it

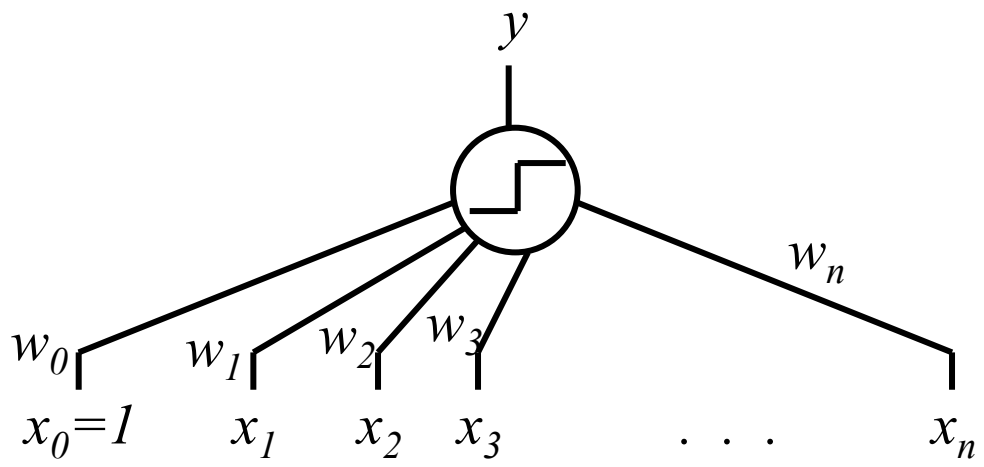






Neurons

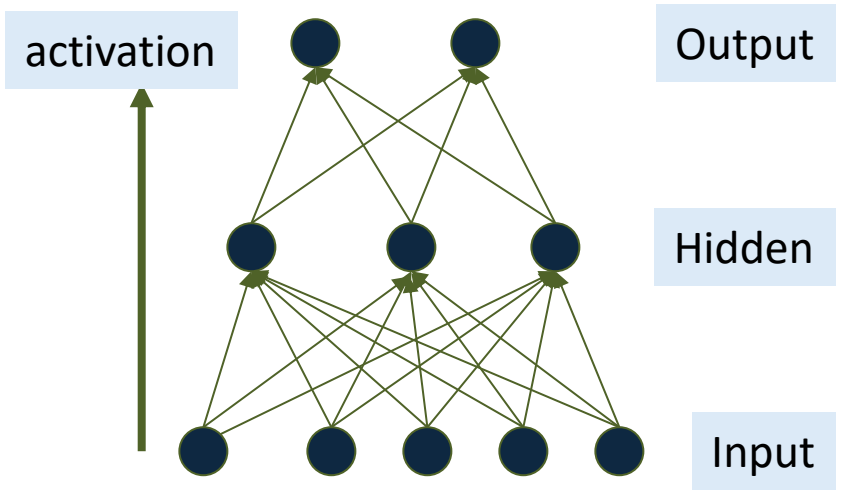
- Abstracting the biological neuron, a mathematical version:
- Each input signal is weighted by a learned real value
- If the sum of weighted inputs exceeds a threshold, neuron outputs
- Notice: **A neuron is a linear classifier!**



$$y = \begin{cases} 1 & \text{if } \sum_{i=0}^{n} w_i x_i \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Multi Layer Neural Networks

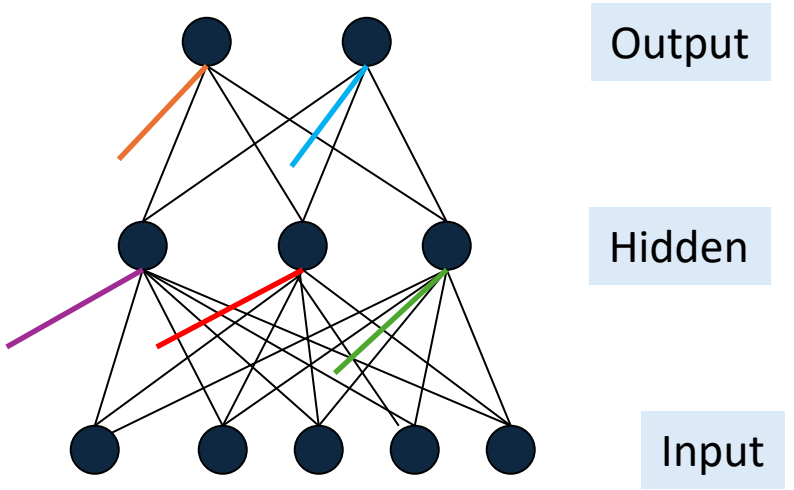
- Multi-layer network were designed to overcome the computational (**expressivity**) limitation of a single threshold element.
- The idea is to **stack** several layers of threshold elements, *each layer using the output of the previous layer as input*



Multi-layer networks **can represent arbitrary functions**, but building effective learning methods for such network was/is difficult.

Note on Bias

- Similar to other models, each (non-input) unit in a NN has a bias term. **We usually do not mention it explicitly.**
- Each bias term is weighted **differently.**



Neural Network

- Simply put, NN's are functions **f: X→Y**
  - *f is a **non-linear** function*
  - *X is a **vector** of continuous or discrete variables*
  - *Y is a **vector** of continuous or discrete variables*
- **Very expressive classifier**
  - In fact, NN can be used to represent any function
- The function f is represented using a network of logistic units