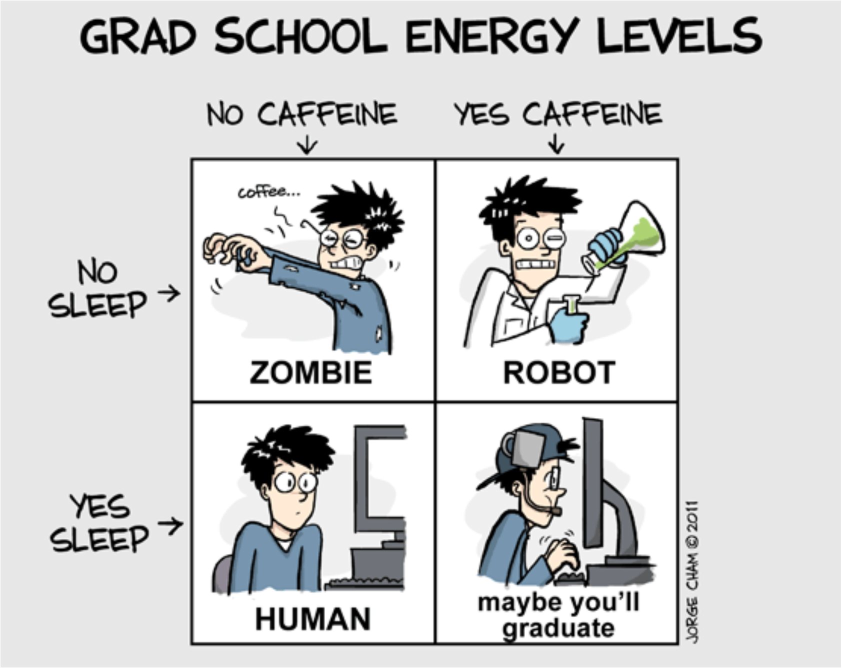
ML4NLP
Multiclass Classification



Dan Goldwasser

Purdue University

dgoldwas@purdue.edu

Social NLP

- *Last week we discussed the speed-dates paper.*
- *Interesting perspective on NLP problems-*
  - *Can we use NLP methods in social science research?*
  - *Key idea:*
    - *Find a problem in the real world that you can about.*
    - ***DEFINE THE PROBLEM FORMALLY***
    - *Identify relevant NLP tools that can measure what you are interested in.*
    - ***Evaluate whether they actually do.***
    - *Use NLP tools to process large amounts of data and say something meaningful about the world that you couldn't before.*

Some (free!) pointers for further reading

- **Linguistic Structure Prediction**. Noah Smith.
- http://www.morganclaypool.com/doi/abs/10.2200/S00361ED1V01Y201105HLT013

- **Structured Learning and Prediction in Computer Vision**. Nowozin and Lampert

http://www.nowozin.net/sebastian/papers/nowozin2011structured-tutorial.pdf

- **Speech and Language Processing.** Jurafsky and Martin

https://web.stanford.edu/~jurafsky/slp3/

Classification

- *A fundamental machine learning tool*
  - Widely applicable in NLP
- **Supervised learning**: Learner is given a collection of labeled documents
  - Emails: Spam/not spam; Reviews: Pos/Neg
- Build a **function** mapping documents to labels
  - Key property: ***Generalization***
    - function should work well on new data

Learning as Optimization

- **Discriminative Linear classifiers**
  - So far we looked **perceptron**
  - Combines model (linear representation) with algorithm (update rule)
  - Let's try to **abstract** – we want to find a linear function performing best on the data
    - What are good properties of this classifier?
    - Want to explicitly control for error + "simplicity"
  - **How can we discuss these terms separately from a specific algorithm?**
    - Search space of all possible linear functions
    - **Find a specific function that has certain properties..**

Classification

- **So far**:
  - General optimization framework for learning
  - Minimize regularized loss function

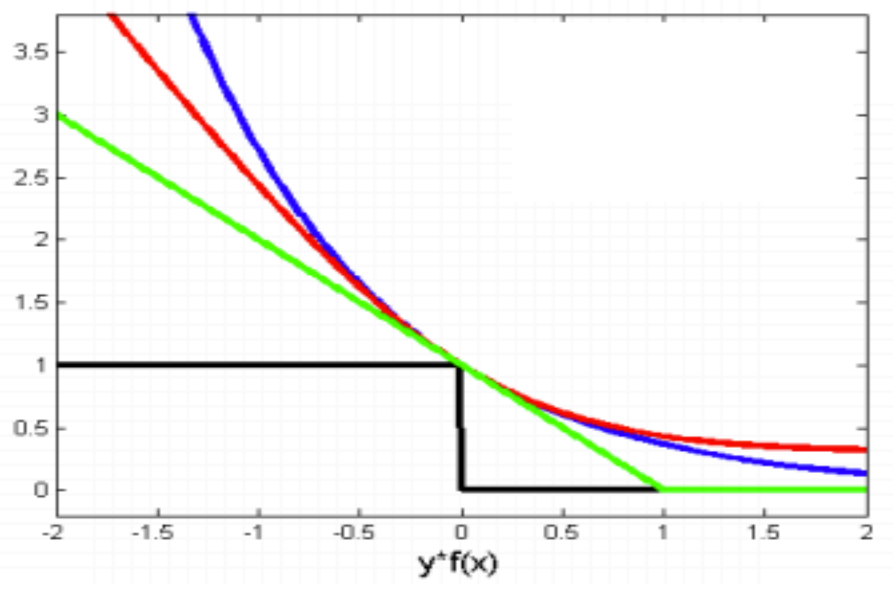
$$\min_{\mathrm{w}} = \sum_n loss(y_n, \mathrm{w}_n) + \frac{\lambda}{2}||\mathrm{w}||^2$$

  - Gradient descent is an all purpose tool
    - Computed the gradient of the *square loss function*

-

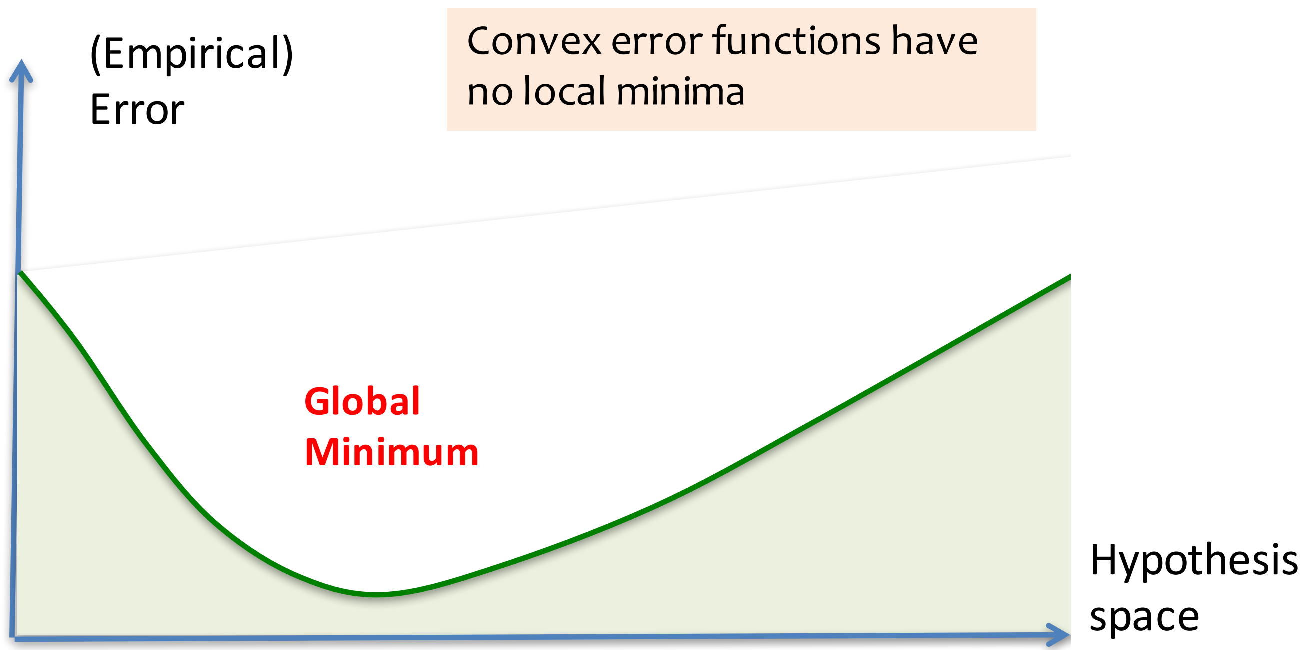
Surrogate Loss functions

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



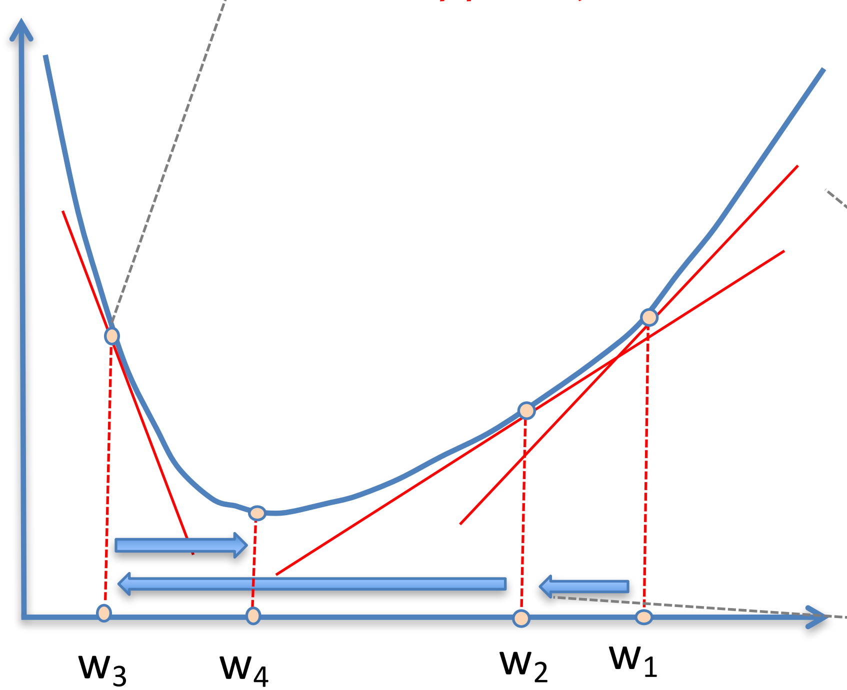
Convex Error Surfaces

- **Convex functions** have a single minimum point
  - Local minimum = global minimum
  - *Easier to optimize*



Gradient Descent Intuition

(3) We also need to determine the step size (aka learning rate).
What happens if we overshoot?



(1) The derivative of the function at w_1 is the slope of the tangent line
→ Positive slope (increasing)
Which direction should we move to decrease the value of Err(w)?

(2) The gradient determines the direction of steepest increase of Err(w) *(go in the opposite direction)*

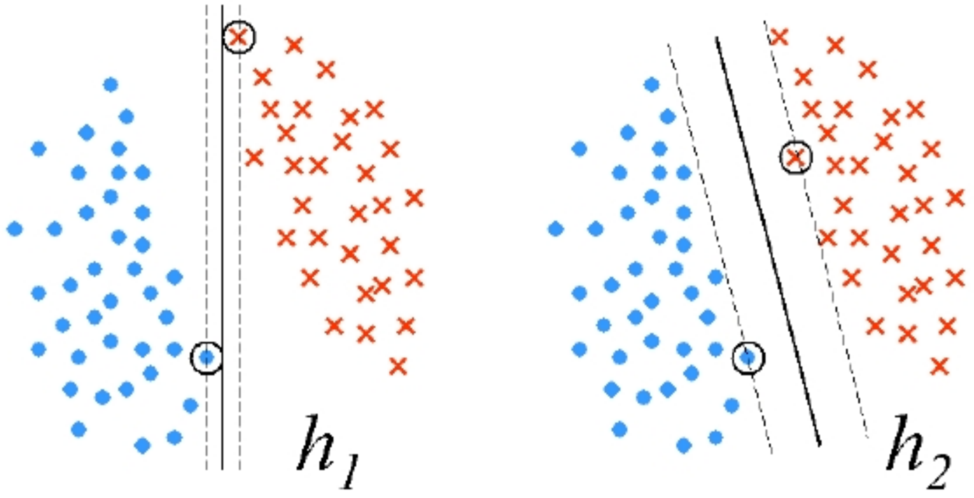
What is the gradient of Error(w) at this point?

Maximal Margin Classification

Motivation for the notion of ***maximal margin***

Defined w.r.t. a dataset S :

$$\gamma(S) = \max \quad \min_{(x,y) \in S} y\, w^T x / ||w||$$



Some Definitions

- **Margin**: distance of the closest point from a hyperplane

This is known as the *geometric margin*, the **numerator** is known as the *functional margin*

$$\gamma = \min_{x_i, y_i} \frac{y_i(\mathrm{w}^T x_i + b)}{||\mathrm{w}||}$$

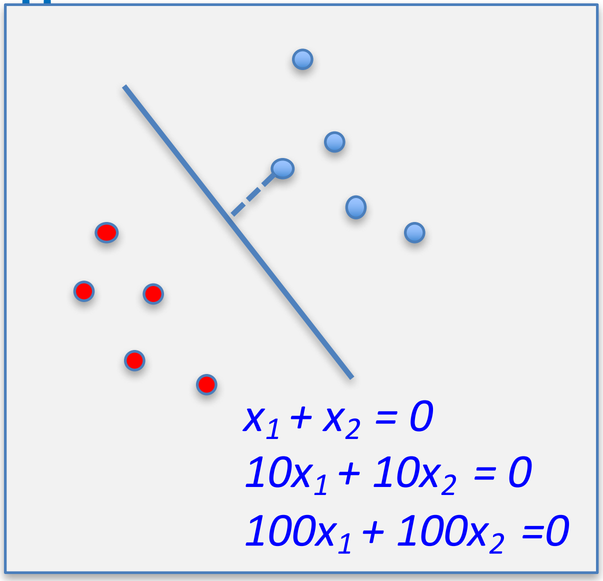
- Our new objective function
  - Learning is essentially solving:

$$\max_{\mathrm{w}} \gamma$$

Maximal Margin

- We want to find $\max_{w} \gamma$
- **Observation**: *we don't care about the magnitude of w!*
- *Set the functional margin to 1, and minimize **W***
- $\max_w \gamma$ is equivalent to $\min_w ||w||$ in this setting
- **To make life easy**:

  let's minimize $\min_w ||w||^2$



Hard vs. Soft SVM

Minimize: $\frac{1}{2}||w||^2$

Subject to: $\forall (x,y) \in S$: $y\, w^T x \geq 1$

Minimize: $\frac{1}{2}||w||^2 + c \sum_i \xi_i$

Subject to: $\forall (x,y) \in S$: $y_i\, w^T x_i \geq 1-\xi_i$; $\xi_i > 0$, $i=1\ldots m$

Objective Function for Soft SVM

- The relaxation of the constraint: $y_i w_i^T x_i \geq 1$ can be done by introducing a slack variable ξ (per example) and requiring: $y_i w_i^T x_i \geq 1 - \xi_i$; $(\xi_i \geq)$
- Now, we want to solve:

$$\text{Min } ½ ||w||^2 + c \Sigma \boldsymbol{\xi_i} \quad (\text{subject to } \xi_i \geq 0)$$

- Which can be written as:

$$\text{Min } ½ ||w||^2 + c\, \boldsymbol{\Sigma \max(0, 1 - y_i w^T x_i)}.$$

- *What is the interpretation of this?*
- This is the **Hinge loss** function

Generalization into Multiclass problems

Multiclass classification Tasks

- So far, our discussion was limited to **binary predictions**
  - Well, ***almost*** (?)
- What happens if our decision is not over binary labels?
  - *Many interesting classification problems are not!*
  - **POS**: Noun,verb, determiner,..
  - **Document classification**: sports, finance, politics
  - **Sentiment**: *Positive, negative, objective*

How can we approach these problems?

Hint: What is the computer science solution to: *"I can solve problem A, but now I have problem B, so…"*

- *Can the problem be reduced into a binary classification problem?*

Multiclass classification

- We will look into two approaches:
  - **Combining multiple binary classifiers**
    - One-vs-All
    - All-vs-All
  - **Training a single classifier**
    - Extending SVM to the multiclass case

One-Vs-All

Assumption: *Each class can be separated from the rest using a binary classifier*

- **Learning**: Decomposed to learning k independent binary classifiers, one corresponding to each class
  - An example (x,y) is considered positive for class y and negative to all others.
  - Assume m examples, k class labels (assume m/k in each)
  - Classifier f_i: m/k (positive) and (k-1)m/k (negative)
- **Decision**: *Winner Takes All*:
  - $f(x) = \text{argmax}_i\, f_i(x) = \text{argmax}_i\, (v_i x)$

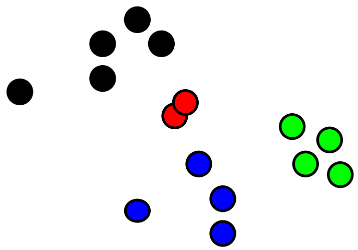
Q: Why do we need the assumption above?

Example: *One-vs-All*

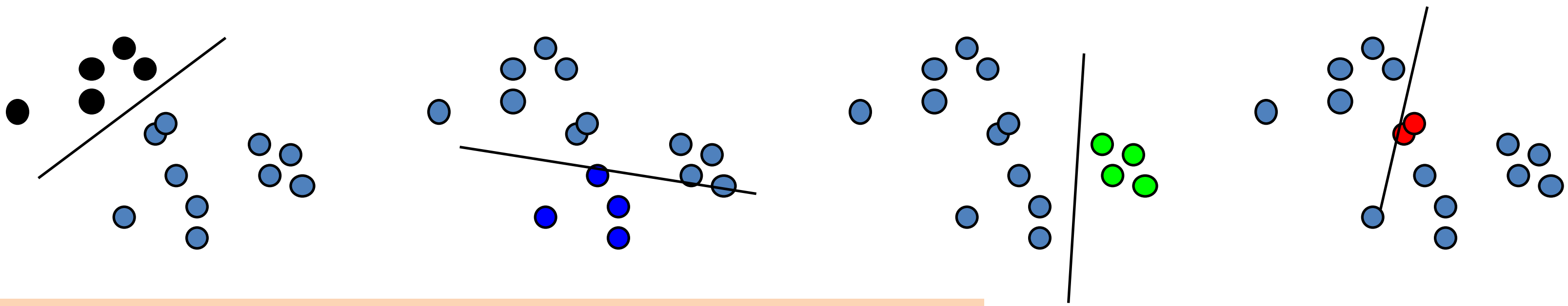
From the full dataset, construct three binary classifiers, one for each class

Solving Multi-Class with binary learning

- **Multi-Class classifier**
  - **Function** f : x → {1,2,3,...,k}



- **Decompose into binary problems**



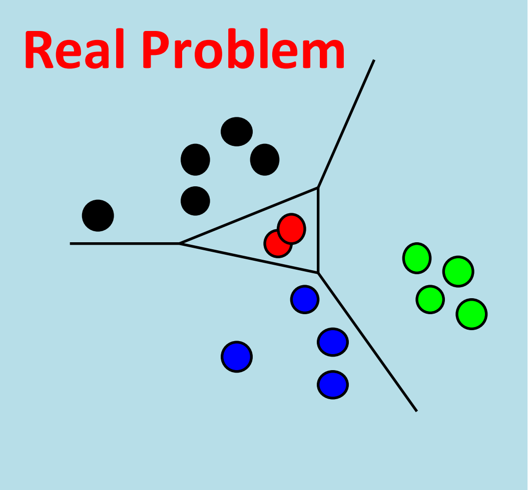
Not always possible to learn

Learning via One-Versus-All

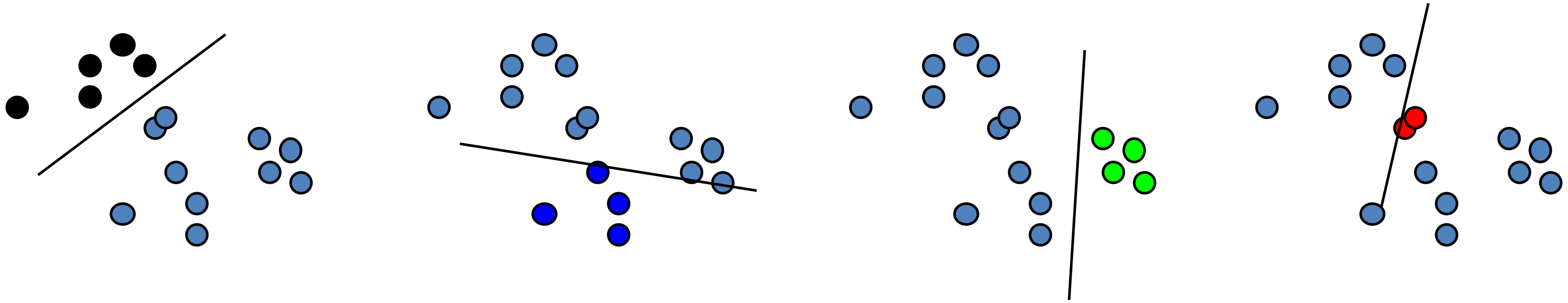
- Find $v_r, v_b, v_g, v_y \in \mathbf{R}^n$ such that

  $\mathbf{H} = \mathbf{R}^{kn}$

  - $v_r\, x > 0$ iff y = red ☹
  - $v_b\, x > 0$ iff y = blue ☺
  - $v_g\, x > 0$ iff y = green ☺
  - $v_y\, x > 0$ iff y = yellow ☺



- **Classification:** $f(x) = \mathrm{argmax}_i\, (v_i\, x)$



All-vs-All

Assumption: There is a separation between **every pair of classes** using a binary classifier in the hypothesis space.

- **Learning**: Decomposed to learning [k choose 2] ~ k^2 independent binary classifiers, separating between every two classes
  - Assume m examples, k class labels. For simplicity, say, m/k in each.
  - Classifier f_{ij}: m/k (positive) and m/k (negative)

- **Decision**: Winner Decision procedure is more involved since output of binary classifier may not cohere (transitivity no ensured) :
  - **Majority**: classify example x to label i if i wins on it more often that j (j=1,…k)
  - ***Tournament***: start with n/2 pairs; continue with winners

All-vs-All

- Every pair of labels is linearly separable here
  - When a pair of labels is considered, all others are ignored
- Problems
  1. $O(K^2)$ weight vectors to train and store
  2. Size of training set for a pair of labels could be very small, leading to overfitting
  3. Prediction is often ad-hoc and might be unstable

     Eg: What if two classes get the same number of votes? For a tournament, what is the sequence in which the labels compete?

Multiclass by reduction to Binary

- Decompose the multiclass learning problem into multiple binary learning problems

- **Prediction**: combine binary classifiers
- **Learning**: optimize local correctness
  - No global view
  - *Separation between learning and prediction procedures*

- We can train the classifier to meet the **REAL objective**
  - **Real objective**: final performance, not local metric

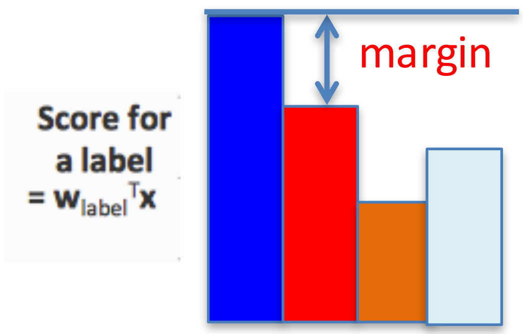
Multiclass SVM

- **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***.

Hard Multiclass SVM

Regularization

$$\min_{\mathbf{w}_1,\mathbf{w}_2,\cdots,\mathbf{w}_K} \quad \frac{1}{2}\sum_k \mathbf{w}_k^T \mathbf{w}_k$$

$$\text{s.t.} \quad \mathbf{w}_{\mathbf{y}_i}^T \mathbf{x} - \mathbf{w}_k^T \mathbf{x} \geq 1 \qquad \forall(\mathbf{x}_i, \mathbf{y}_i) \in D, \; k \in \{1, 2, \cdots, K\}, k \neq \mathbf{y}_i,$$

The score of the true label has to be higher than 1, *for **any** label*

Soft Multiclass SVM

Regularizer

Slack Variables

$$
\begin{aligned}
\min_{\mathbf{w}_1,\mathbf{w}_2,\cdots,\mathbf{w}_K,\xi} \quad & \frac{1}{2}\sum_k \mathbf{w}_k^T\mathbf{w}_k + C \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in D} \xi_i \\
\text{s.t.} \quad & \mathbf{w}_{\mathbf{y}_i}^T\mathbf{x} - \mathbf{w}_k^T\mathbf{x} \geq 1 - \xi_i, \qquad \forall(\mathbf{x}_i,\mathbf{y}_i) \in D, \\
& \qquad\qquad\qquad\qquad\qquad\quad k \in \{1,2,\cdots,K\}, k \neq \mathbf{y}_i, \\
& \xi_i \geq 0, \qquad\qquad\qquad\qquad\quad \forall i.
\end{aligned}
$$

The score of the true label should have a margin of $1-\xi_i$

Relax hard constraints using slack variables

Positive slack

K. Crammer, Y. Singer: "On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines", JMLR, 2001

Alternative Notation

- For examples with label *i* we want: $w_i^T x > w_j^T x$
- **Alternative notation**: *Stack all weight vectors*

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

x in the i^th block, zeros everywhere else

- *Define features jointly over the input and output*

$$\mathbf{w}^T \phi(\mathbf{x}, i) > \mathbf{w}^T \phi(\mathbf{x}, j)$$ *is equivalent to* $w_i^T x > w_j^T x$

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 $i = 1, \ldots, n$,

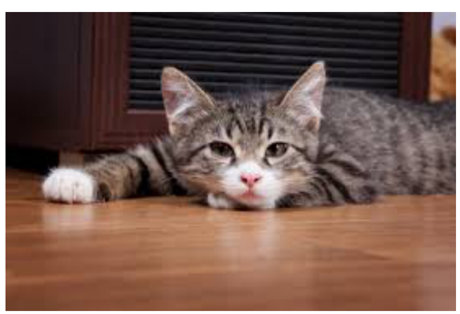
$$\langle w, \phi(x^n, y^n)\rangle - \langle w, \phi(x^n, y)\rangle \geq 1 - \xi^n \quad \text{for all } y \in \mathcal{Y} \setminus \{y^n\}.$$

- **Prediction**

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

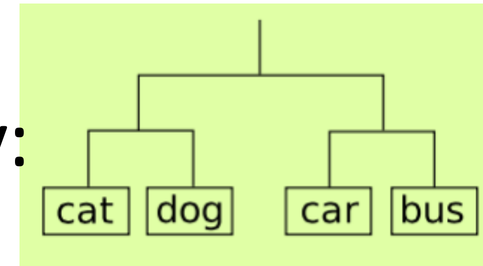
Cost Sensitive Multiclass Classification

- Sometime we are willing to “tolerate” some mistakes more than others



Cost Sensitive Multiclass Classification

- We can think about it as a hierarchy:



- Define a distance metric:
  - $\Delta(y,y')$ = tree distance between y and y'

We would like to incorporate that into our learning

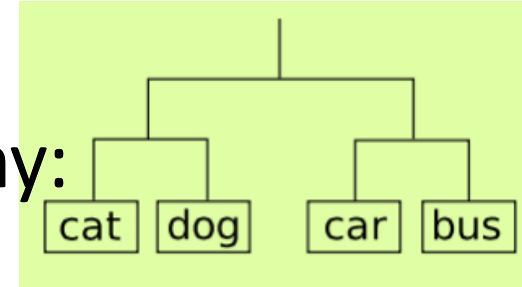
Solve: $$\min_{w,\xi} \frac{1}{2}\|w\|^2 + \frac{C}{N}\sum_{n=1}^{N}\xi^n$$

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

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

Cost Sensitive Multiclass Classification

- We can think about it as a hierarchy:



- Define a distance metric:
  - $\Delta(y,y')$ = tree distance between y and y'

We would like to incorporate that into our learning model

$\Delta(y^n, y)$

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

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

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

Cost Sensitive Multiclass Classification

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

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

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

Instead we can have an unconstrained version -

Question: What is sub-gradient of this loss function?

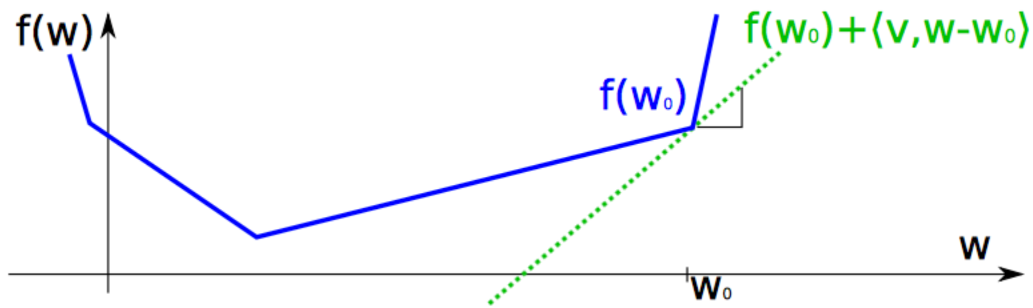
Solve: $$\min_{w,\xi} \frac{1}{2}\|w\|^2 + \frac{C}{N}\sum_{n=1}^{N} l(\mathbf{w};(\mathbf{x}^n, y^n))$$

$$l(\mathbf{w};(\mathbf{x}^n, y^n)) = \max_{y' \in Y} \Delta(y, y') - \langle \mathbf{w}, \phi(\mathbf{x}^n, y^n)\rangle + \langle \mathbf{w}, \phi(\mathbf{x}^n, y')\rangle$$

Reminder: Subgradient descent

Let $f : \mathbb{R}^D \to \mathbb{R}$ be a convex, not necessarily differentiable, function.
A vector $v \in \mathbb{R}^D$ is called a **subgradient** of f at w_0, if

$$f(w) \geq f(w_0) + \langle v, w - w_0 \rangle \quad \text{for all } w.$$

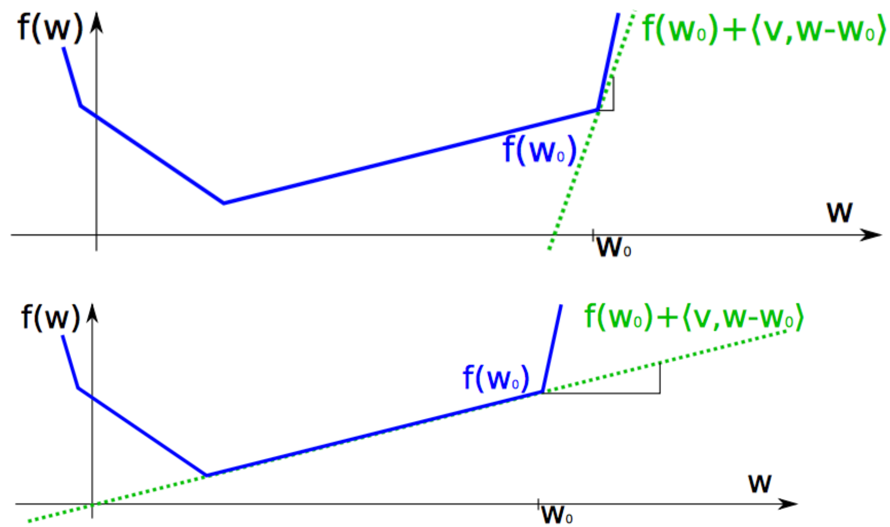


Slides by Sebastian Nowozin and Christoph H. Lampert "structured models in computer vision" tutorial CVPR 2011

Reminder: Subgradient descent

Let $f : \mathbb{R}^D \to \mathbb{R}$ be a convex, not necessarily differentiable, function.
A vector $v \in \mathbb{R}^D$ is called a **subgradient** of f at w_0, if

$$f(w) \geq f(w_0) + \langle v, w - w_0 \rangle \quad \text{for all } w.$$



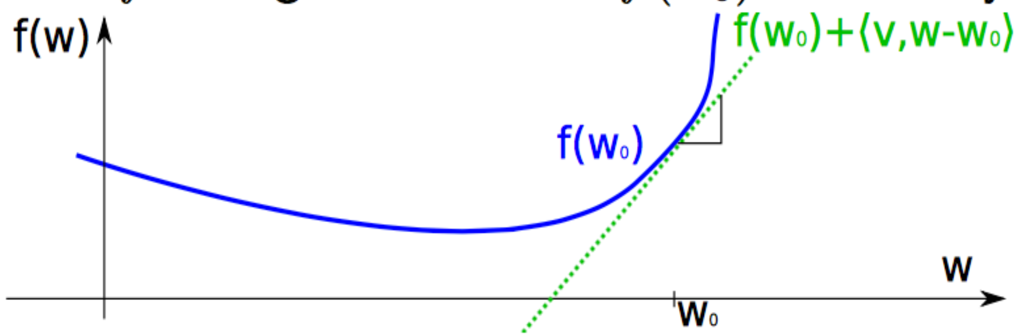
Slides by Sebastian Nowozin and Christoph H. Lampert "structured models in computer vision" tutorial CVPR 2011

Reminder: Subgradient descent

Let $f : \mathbb{R}^D \to \mathbb{R}$ be a convex, not necessarily differentiable, function.
A vector $v \in \mathbb{R}^D$ is called a **subgradient** of f at w_0, if

$$f(w) \geq f(w_0) + \langle v, w - w_0 \rangle \quad \text{for all } w.$$

For differentiable f, the gradient $v = \nabla f(w_0)$ is the only subgradient.



Slides by Sebastian Nowozin and Christoph H. Lampert "structured models in computer vision" tutorial CVPR 2011

Reminder: Subgradient descent

Subgradient Descent Minimization – minimize $F(w)$

- **require:** tolerance $\epsilon > 0$, stepsizes η_t
- $w_{cur} \leftarrow 0$
- **repeat**
  - $v \in \nabla^{\text{sub}}{}_{w} F(w_{cur})$
  - $w_{cur} \leftarrow w_{cur} - \eta_t v$
- **until** F changed less than ϵ
- **return** w_{cur}

Converges to global minimum, but rather inefficient if F non-differentiable.

Slides by Sebastian Nowozin and Christoph H. Lampert "structured models in computer vision" tutorial CVPR 2011

Subgradient for the MC case

Computing a subgradient:

$$\min_{w} \frac{1}{2}\|w\|^2 + \frac{C}{N}\sum_{n=1}^{N} \ell^n(w)$$

with $\ell^n(w) = \max_y \ell^n_y(w)$, and

$$\ell^n_y(w) := \Delta(y^n, y) + \langle w, \phi(x^n, y)\rangle - \langle w, \phi(x^n, y^n)\rangle$$

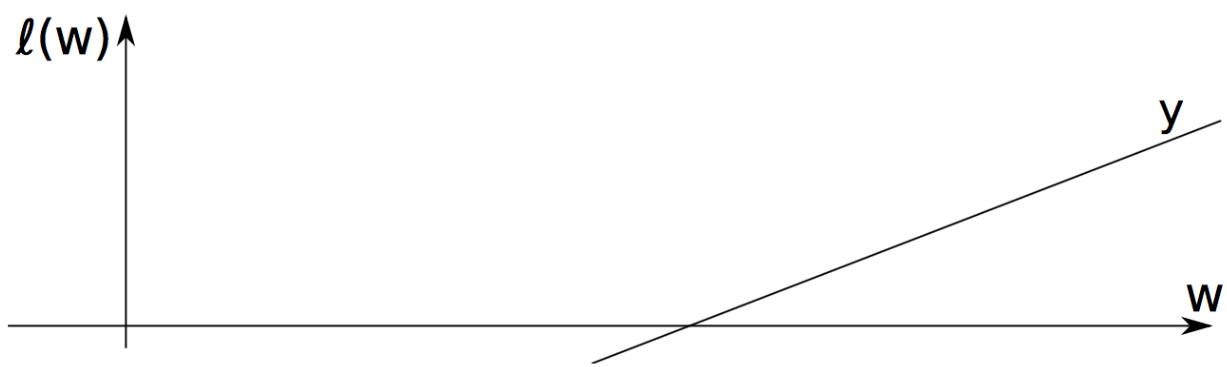
Subgradient for the MC case

Computing a subgradient:

$$\min_w \frac{1}{2}\|w\|^2 + \frac{C}{N}\sum_{n=1}^{N} \ell^n(w)$$

with $\ell^n(w) = \max_y \ell^n_y(w)$, and

$$\ell^n_y(w) := \Delta(y^n, y) + \langle w, \phi(x^n, y)\rangle - \langle w, \phi(x^n, y^n)\rangle$$



For each $y \in \mathcal{Y}$, $\ell_y(w)$ is a linear function.

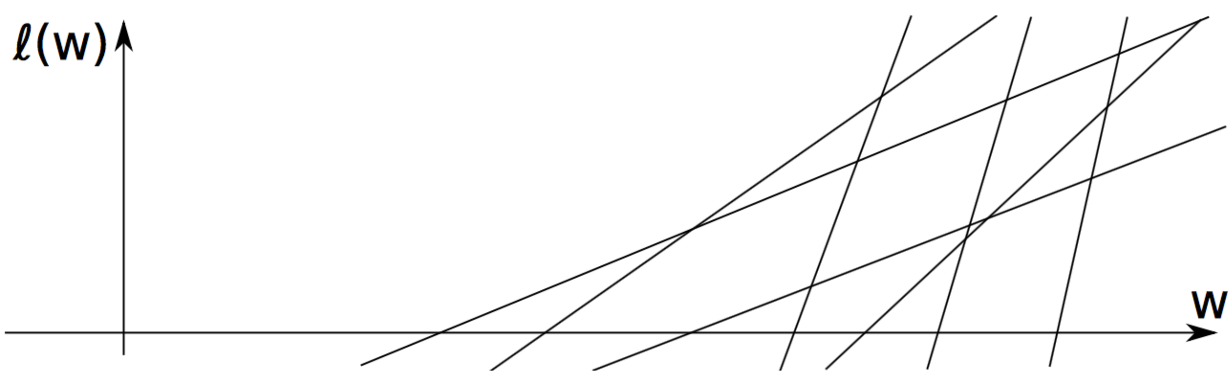
Subgradient for the MC case

Computing a subgradient:

$$\min_{w} \ \frac{1}{2}\|w\|^2 + \frac{C}{N}\sum_{n=1}^{N} \ell^n(w)$$

with $\ell^n(w) = \max_y \ell_y^n(w)$, and

$$\ell_y^n(w) := \Delta(y^n, y) + \langle w, \phi(x^n, y)\rangle - \langle w, \phi(x^n, y^n)\rangle$$



For each $y \in \mathcal{Y}$, $\ell_y(w)$ is a linear function.

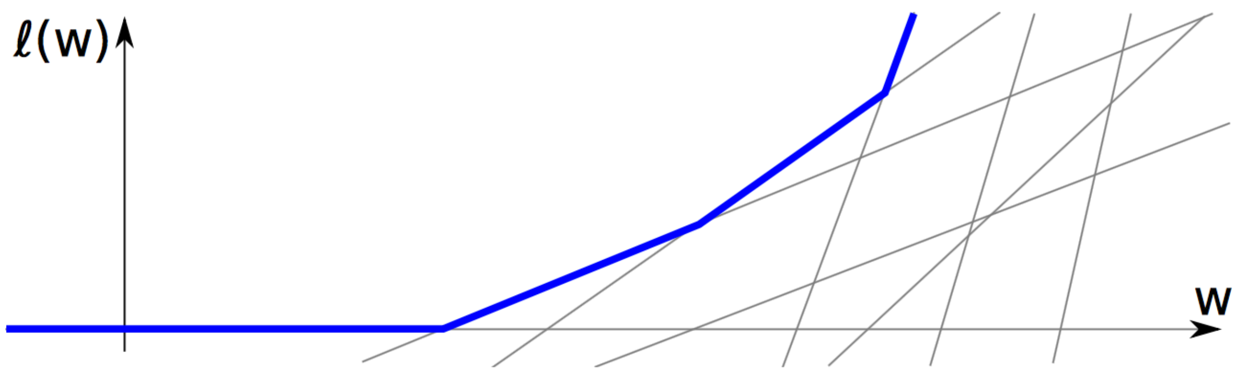
Subgradient for the MC case

Computing a subgradient:

$$\min_w \frac{1}{2}\|w\|^2 + \frac{C}{N}\sum_{n=1}^{N} \ell^n(w)$$

with $\ell^n(w) = \max_y \ell^n_y(w)$, and

$$\ell^n_y(w) := \Delta(y^n, y) + \langle w, \phi(x^n, y)\rangle - \langle w, \phi(x^n, y^n)\rangle$$



$\ell(w) = \max_y \ell_y(w)$: maximum over all $y \in \mathcal{Y}$.

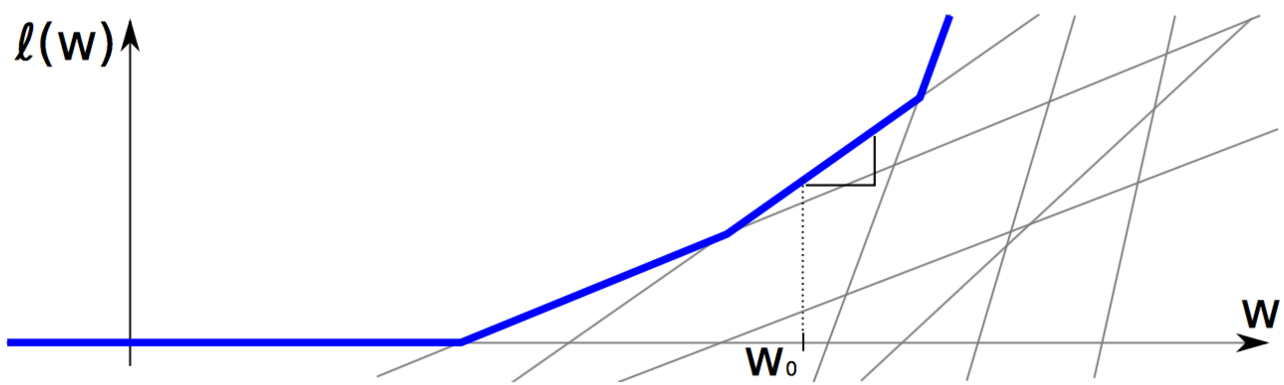
Subgradient for the MC case

Computing a subgradient:

$$\min_w \frac{1}{2}\|w\|^2 + \frac{C}{N}\sum_{n=1}^{N} \ell^n(w)$$

with $\ell^n(w) = \max_y \ell^n_y(w)$, and

$$\ell^n_y(w) := \Delta(y^n, y) + \langle w, \phi(x^n, y)\rangle - \langle w, \phi(x^n, y^n)\rangle$$



Subgradient of ℓ^n at w_0: find maximal (active) y, use $v = \nabla \ell^n_y(w_0)$.

Subgradient descent for the MC case

Subgradient Descent S-SVM Training

input training pairs $\{(x^1, y^1), \dots, (x^n, y^n)\} \subset \mathcal{X} \times \mathcal{Y}$,
input feature map $\phi(x, y)$, loss function $\Delta(y, y')$, regularizer C,
input number of iterations T, stepsizes η_t for $t = 1, \dots, T$

$w \leftarrow \vec{0}$
for t=1,...,T **do**
  for i=1,...,n **do**
    $\hat{y} \leftarrow \mathrm{argmax}_{y \in \mathcal{Y}} \Delta(y^n, y) + \langle w, \phi(x^n, y) \rangle - \langle w, \phi(x^n, y^n) \rangle$
    $v^n \leftarrow \phi(x^n, \hat{y}) - \phi(x^n, y^n)$
  end for
  $w \leftarrow w - \eta_t (w - \frac{C}{N} \sum_n v^n)$
end for

output prediction function $f(x) = \mathrm{argmax}_{y \in \mathcal{Y}} \langle w, \phi(x, y) \rangle$.

Observation: each update of w needs 1 argmax-prediction per example.

Subgradient descent for the MC case

Stochastic Subgradient Descent S-SVM Training

input training pairs $\{(x^1, y^1), \ldots, (x^n, y^n)\} \subset \mathcal{X} \times \mathcal{Y}$,
input feature map $\phi(x, y)$, loss function $\Delta(y, y')$, regularizer C,
input number of iterations T, stepsizes η_t for $t = 1, \ldots, T$

1: $w \leftarrow \vec{0}$
2: **for** t=1,...,T **do**
3: $(x^n, y^n) \leftarrow$ randomly chosen training example pair
4: $\hat{y} \leftarrow \operatorname{argmax}_{y \in \mathcal{Y}} \Delta(y^n, y) + \langle w, \phi(x^n, y)\rangle - \langle w, \phi(x^n, y^n)\rangle$
5: $w \leftarrow w - \eta_t(w - \frac{C}{N}[\phi(x^n, \hat{y}) - \phi(x^n, y^n)])$
6: **end for**

output prediction function $f(x) = \operatorname{argmax}_{y \in \mathcal{Y}} \langle w, \phi(x, y)\rangle$.

Question: What is the difference between this algorithm and the perceptron variant for multiclass classification?

Can you define NER as a multiclass classification problem?

- Named entity recognition (NER)
  - Identify mentions of named entity in text
  - People (PER), places (LOC) and organizations (ORG)

Begin_p

Barak Obama visited Mount Sinai hospital in New York

PER ORG LOC

Conceptually two considerations:

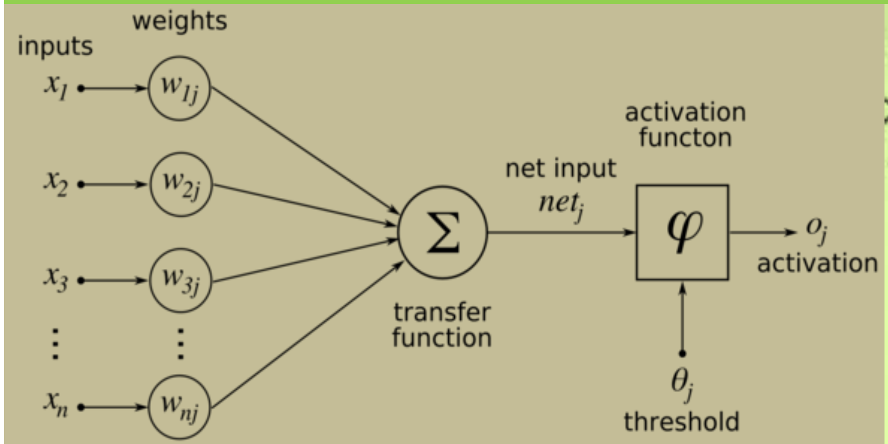
- Entity **boundary (Begin, Inside ,Outside)**
- Entity **type (PER,ORG,LOC)**

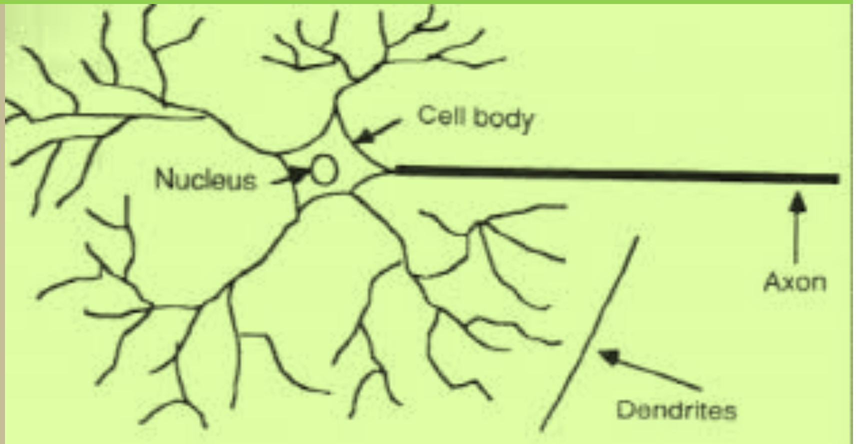
Our Labels:

Begin_p,Begin_L,Begin_o
Inside_p,Inside_L,Inside_o
Outside

Given a sentence, are consecutive predictions independent of each other?

Introduction to Deep Learning





Deep Learning in NLP

- So far we used **linear models**
  - They work fine, but we made some assumptions
- The problems are linear, or we are willing to work in order to make them linear
- Feature engineering is a form of expressing domain knowledge
- We are fine working with **very very high** dimensional data
  - It's easy to get there.
  - Everything is mostly linear at that point.
- *What could go wrong?*

Deep Learning Architectures for NLP

- The key question we follow – how can you **build complex (compositional?) meaning representation**, for **larger units than words**, to support **advanced classification tasks**?
- We will look at several popular architectures.
  - We will build on a *"recently introduced model from the 70's"*
    - Maybe even before that..
    - NN made a come-back in the last 5 years.

Neural Networks

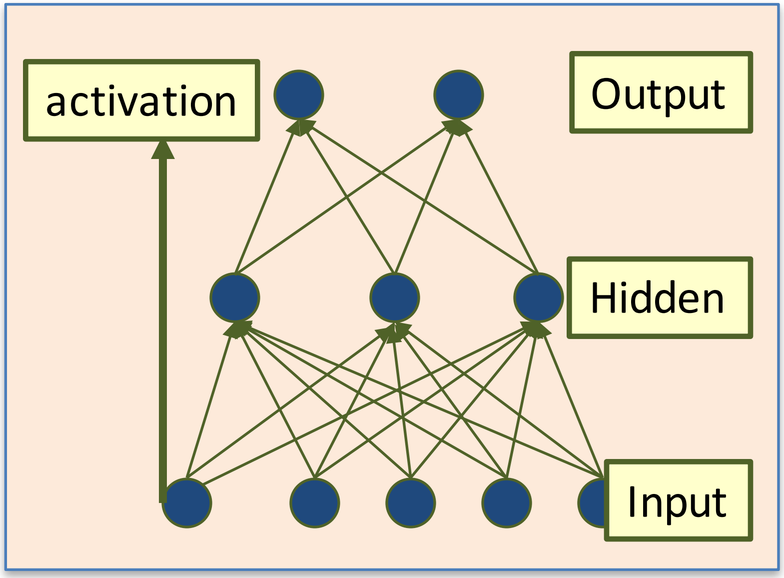
- Robust approach for approximating functions
  - *Functions can be real-valued, discrete or vector valued*
- One of the most effective general purpose learning methods
  - A lot of attention in the 90’s, making a comeback!
- Especially useful for complex problems, where the input data is hard to interpret
  - Sensory data (speech, vision, etc)
- Many successful application domains
- **Interesting spin**: *Learning input representation*
  - *So far we thought about the feature representation as being fixed*

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

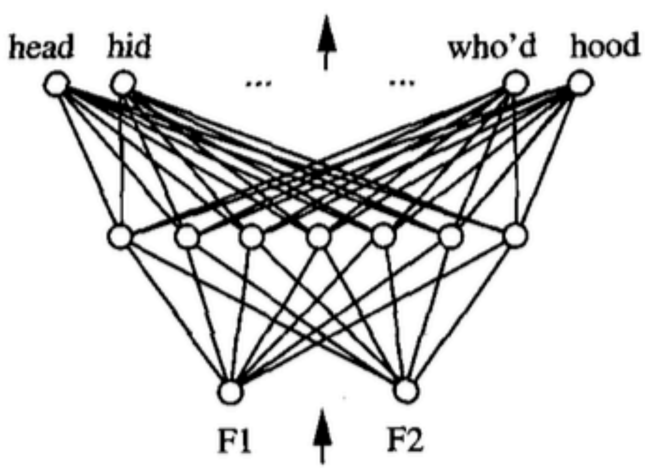
Multi Layer Neural Networks

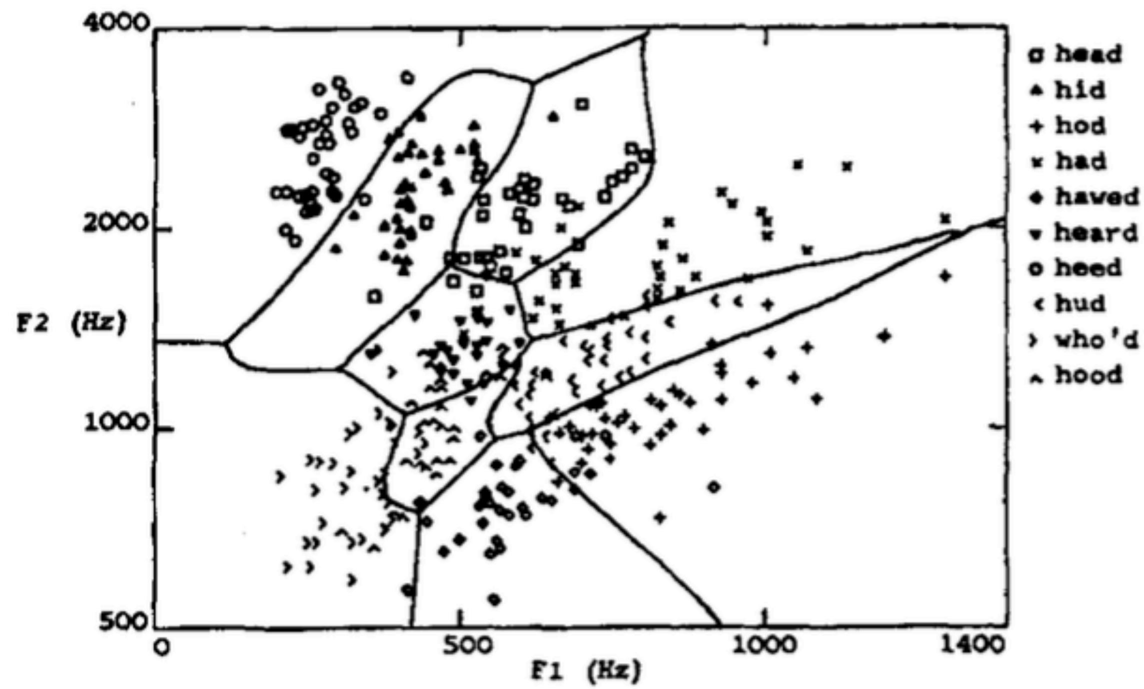
- Multi-layer network were designed to overcome the computational (**expressivity**) limitation of a single threshold element.
- The idea is to **stack** several layers of threshold elements, *each layer using the output of the previous layer as input*



Multi-layer networks **can represent arbitrary functions**, but building effective learning methods for such network was [thought to be] difficult.

Example: NN for speech vowel recognition





ALVINN: autonomous land vehicle in a NN

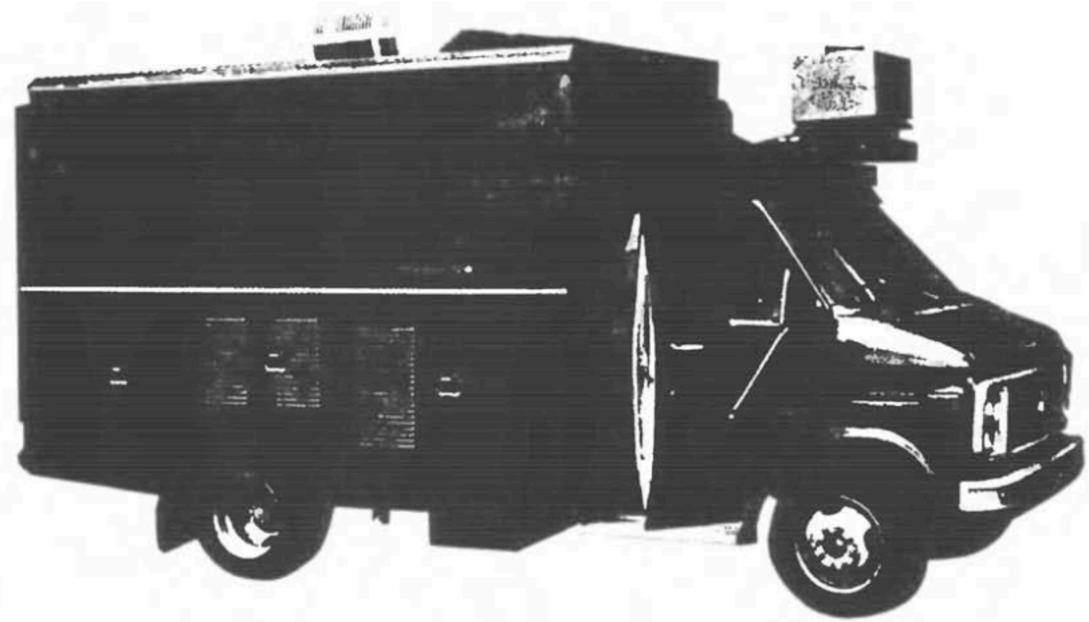
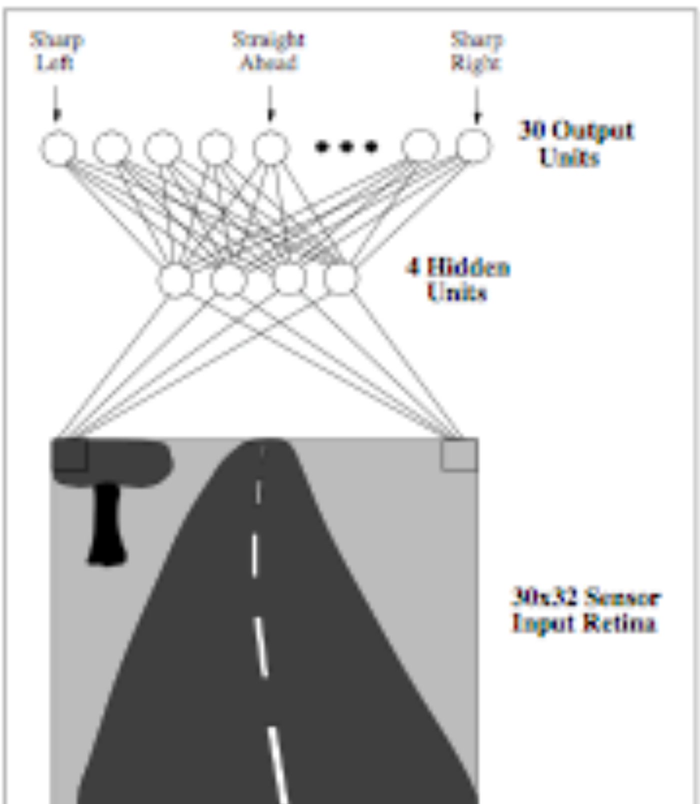
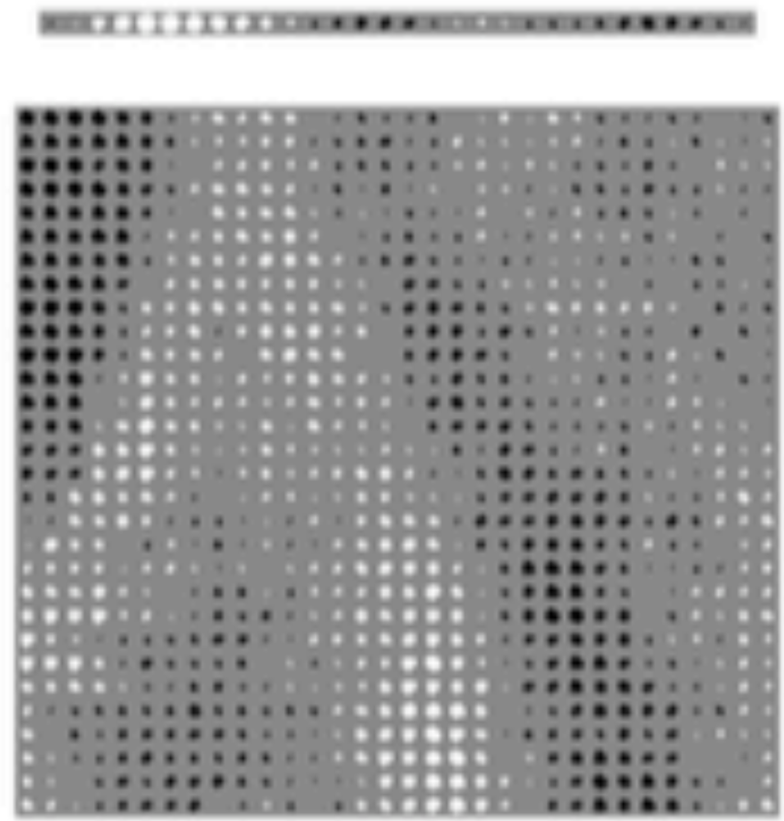


Figure 3: NAVLAB, the CMU autonomous navigation test vehicle.

Pomerleau '89

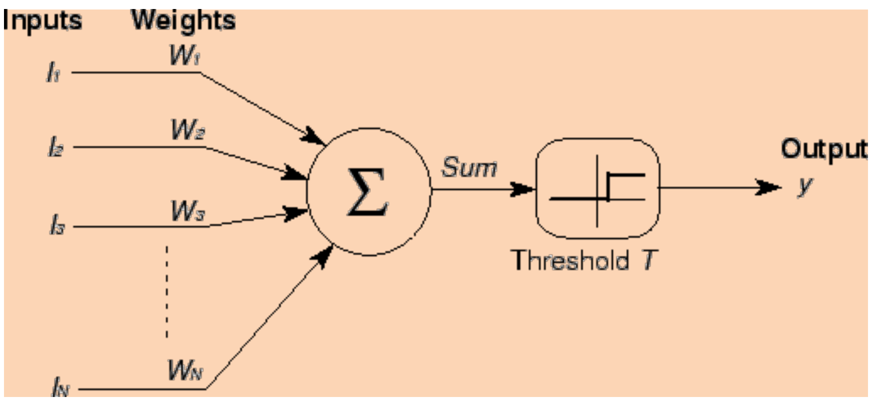
ALVINN: autonomous land vehicle in a NN

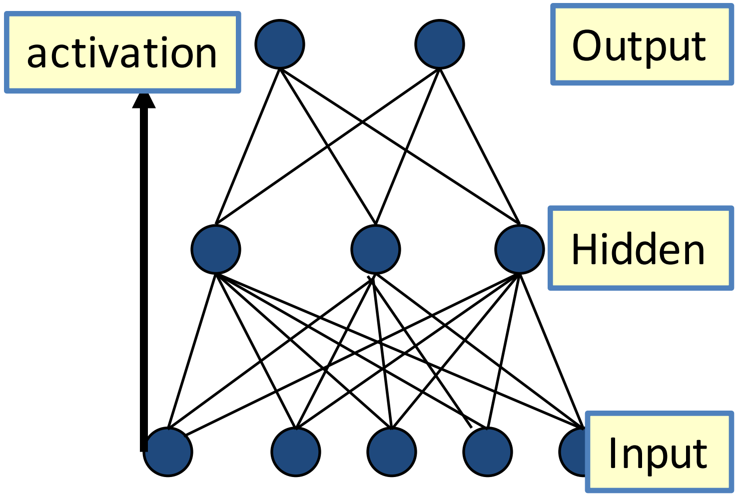




Basic Units in Multi-Layer NN

- Basic element: **linear unit**
  - But, we would like to represent nonlinear functions
    - Multiple layers of linear functions are still linear functions
  - Threshold units are not smooth (we would like to use gradient-based algorithms)

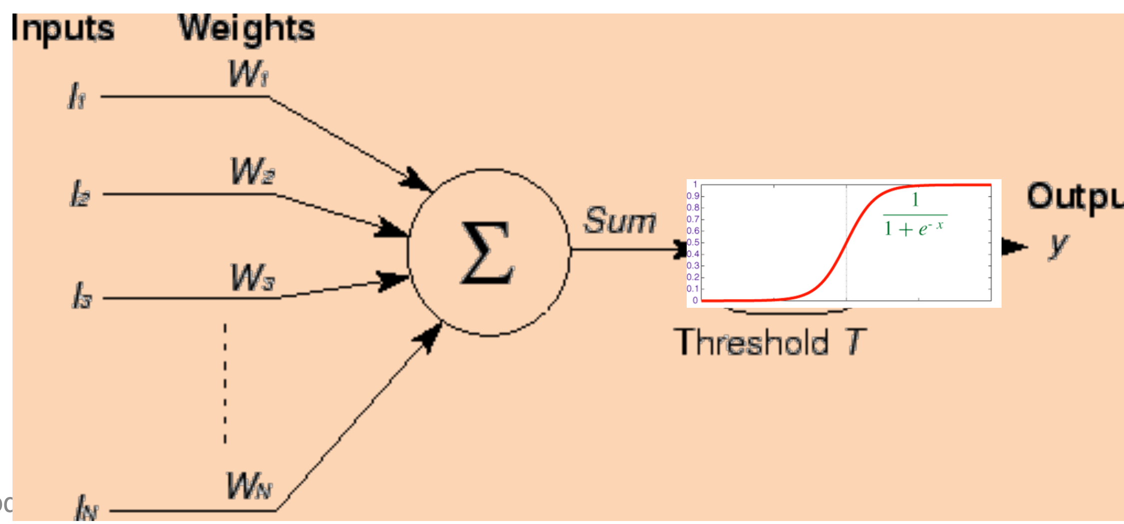




Basic Units in Multi-Layer NN

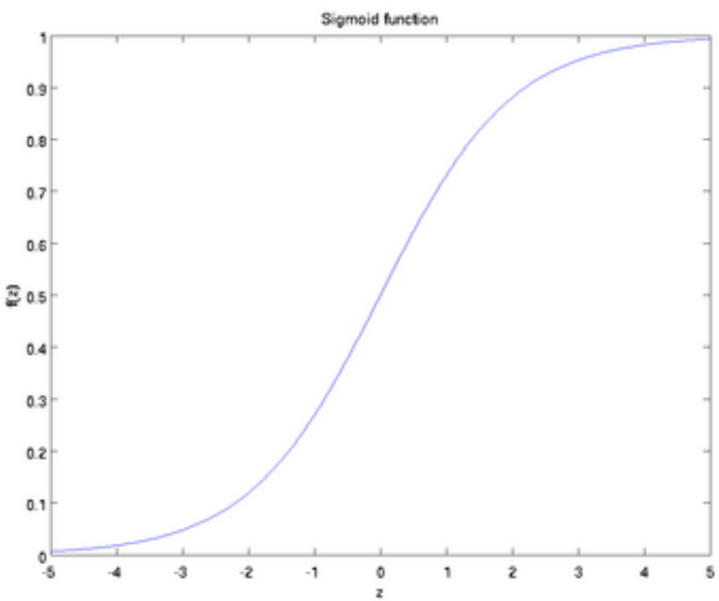
- Basic element: **sigmoid unit**
  - Input to a unit *j* is defined as: $\Sigma w_{ij} x_i$
  - Output is defined as : $\sigma\,(\Sigma w_{ij} x_i)$
    - σ is simply the logistic function: $$\frac{1}{1+e^{-x}}$$

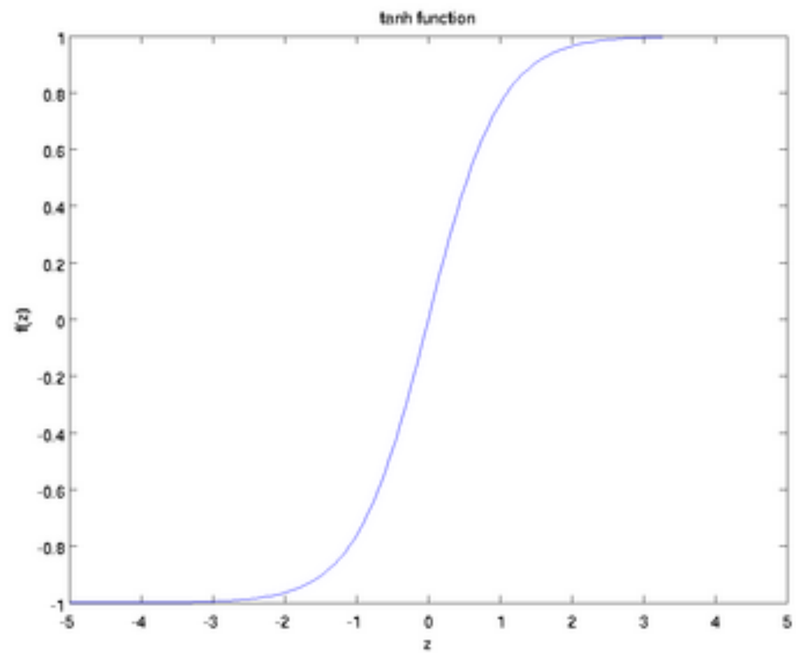
Note: similar to previous algorithms, We encode the bias/threshold, as a "fake" Feature that is always active



Basic Units in Multi-Layer NN

- Basic element: **sigmoid unit**
  - You can also replace the logistic function with other smooth activation functions

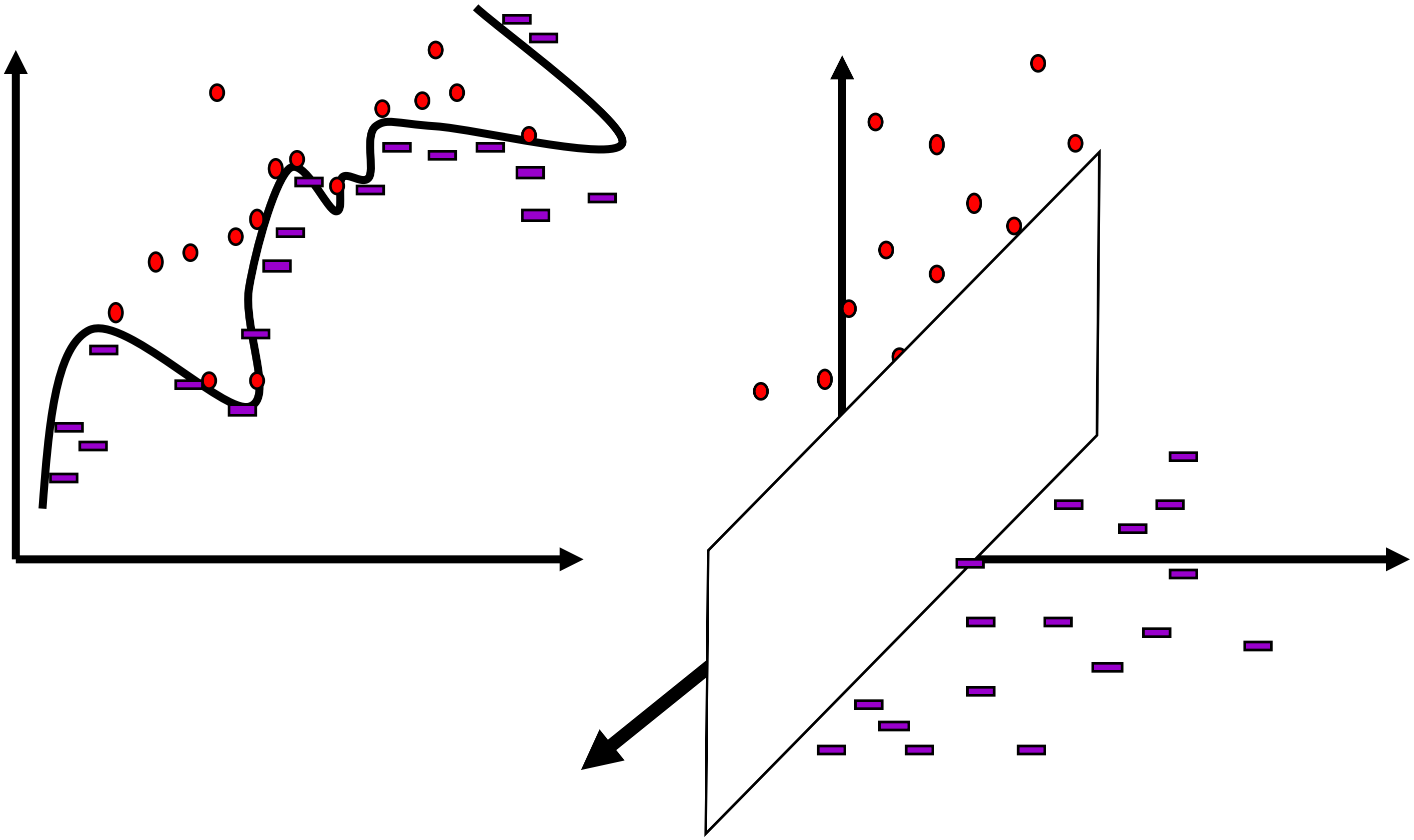




Basic Units in Multi-Layer NN

- **Key issue**: limited expressivity!
  - Minsky and Papert (1969) published an influential books showing what cannot be learned using perceptron
- These observation discouraged research on NN for several years
- But.. we really like linear functions!
- How did we deal with these issues so far?

Basic Units in Multi-Layer NN



In fact , Rosenblatt (1959) asked: *“What pattern recognition problems can be transformed so as to become linearly separable”*

Multi Layer NN

- Another approach for increasing expressivity:
  Stacking multiple sigmoid units to form a network
- Compute the output of the network using a 'feed-forward' computation
- Learn the parameters of the network using the backpropagation algorithm
- Any Boolean function can be represented using a two layer network
- Any bounded continuous function can be approximated using a two layer network