Aprendizaje Supervisado

Franco Luque - Matías Marenchino

RoadMap

- **Short review of topics from previous course**
  Classification / Regression / Perceptron
- **What supervised learning is**
- **SVMs**
- **Ensemble methods**
  Bagging: Random Forests / Boosting
- **Neural Networks**

Motivation

Example 1

- **A credit card company receives applications for new credit cards. Each one has information about an applicant:**
  - salary
  - age
  - marital status
  - Veraz
  - Credit report from BCRA
  - ...
- **Problem: determine if an application should be approved or rejected**

Example 2

- **Problem: classify an email as SPAM or not**

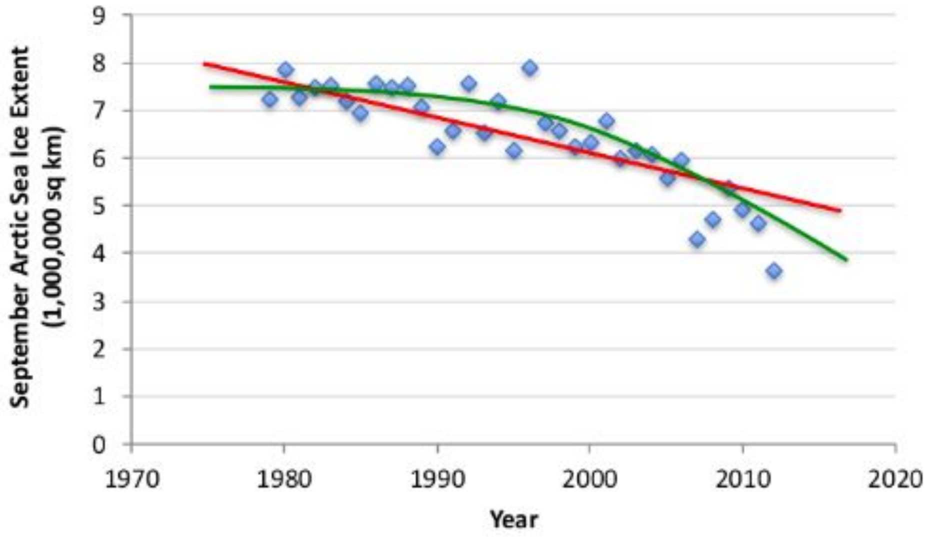
Describing the problem

- **Data:** A set of records (or samples, instances) described by n attributes: A1, A2, … An and each sample is labelled with a class (Like SPAM or NOT) or a "score" (like the credit score)

- **Goal:** To learn a model (or a function) from the data that can be used to predict the labels that the records have (and labels for new unlabelled records)

Aprendizaje supervisado: regresión

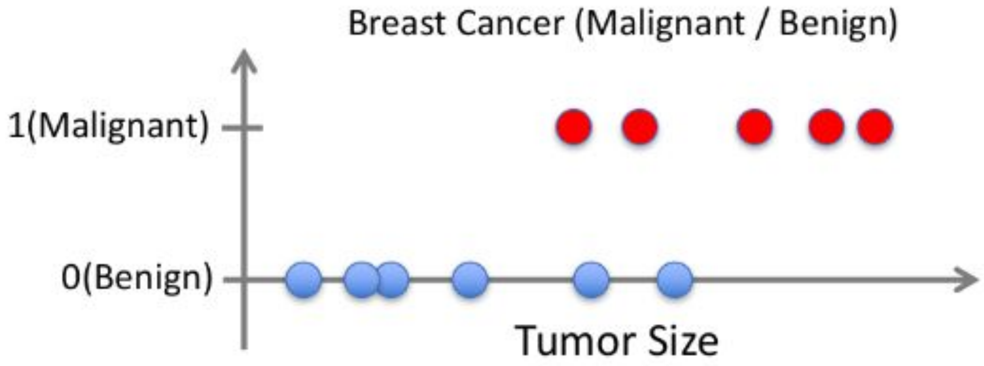
- Dados $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Aprender una $f(x)$ que permita predecir y a partir de x
    - Si y está en $\mathbb{R}^n$ $\rightarrow$ **regresión**



Slides from the previous course

Aprendizaje supervisado: clasificación

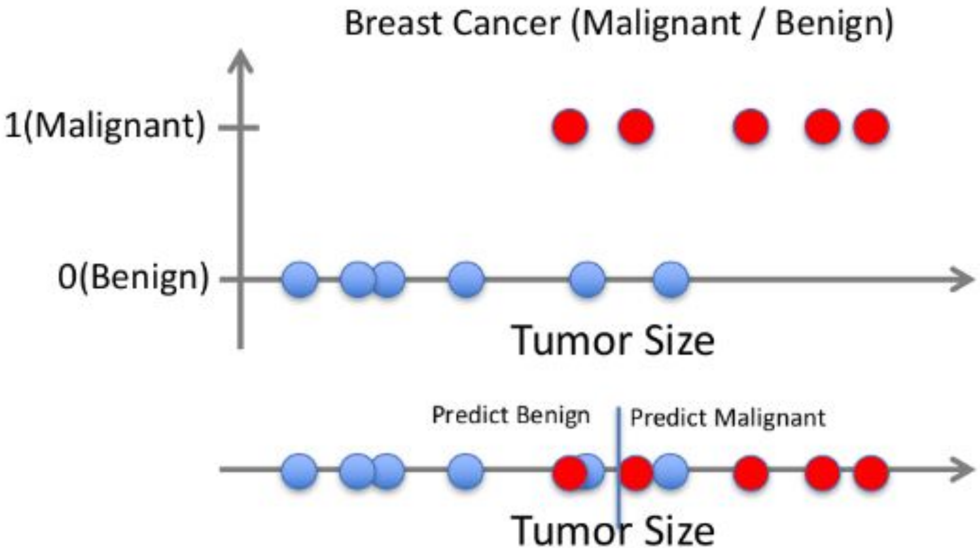
- Dados $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Aprender una $f(x)$ que permita predecir y a partir de x
    - Si y es categórica $\rightarrow$ **clasificación**



Slides from the previous course

Aprendizaje supervisado: clasificación

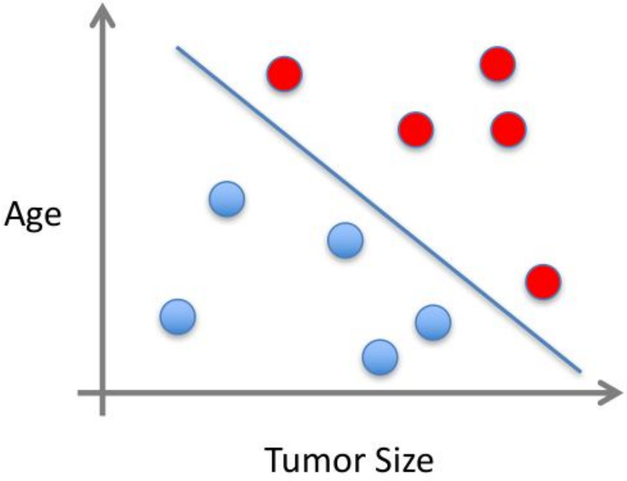
- Dados $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Aprender una $f(x)$ que permita predecir y a partir de x
    - Si y es categórica $\rightarrow$ **clasificación**



Slides from the previous course

Supervised Learning

- x can be multi-dimensional
  – Each dimension corresponds to an attribute



- Clump Thickness
- Uniformity of Cell Size
- Uniformity of Cell Shape

…

Based on example by Andrew Ng

Slides from the previous course

Aprendizaje supervisado

Functions $\mathcal{F}$

$f : \mathcal{X} \to \mathcal{Y}$

Training data

$\{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$

LEARNING

$\boxed{\begin{array}{l} \text{find } \hat{f} \in \mathcal{F} \\ \text{s.t. } y_i \approx \hat{f}(x_i) \end{array}}$

Learning machine

New data

PREDICTION

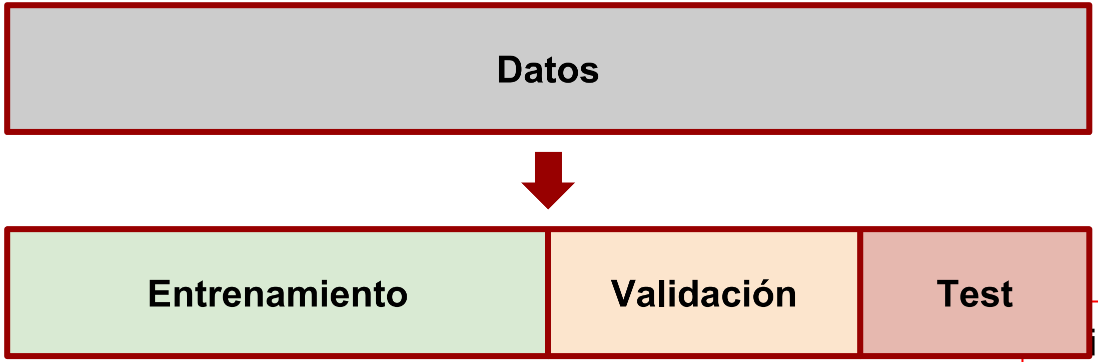
$\boxed{y = \hat{f}(x)} \Leftarrow \boxed{x}$

Slides from the previous course

Elección de *hiperparámetros*

Dividir el conjunto total de ejemplos en tres subconjuntos

- **Entrenamiento**: aprendizaje de variables del modelo
- **Validación**: ajuste/elección de hiperparámetros
- **Test**: estimación final de la performance del modelo entrenado (y con hiperparámetros elegidos adecuadamente



ides from the previous course

El algoritmo del "perceptrón"

- Propuesto por Rosemblatt en 1958
- El objetivo es encontrar un hiperplano de separación
  - Si los datos son linealmente separables, lo encuentra
- Es un algoritmo *online* (procesa un ejemplo a la vez)
- Muchas variantes ...

Slides from the previous course

El algoritmo del "perceptrón"

Entrada:

- una secuencia de pares de entrenamiento (x_1,y_1), (x_2, y_2) ...
- Una tasa de aprendizaje r

Algoritmo:

- Inicializar $w^{(0)} \in \mathbb{R}^{n}$
- Para cada ejemplo (x_i,y_i)
  - Predecir $y_i' = sign(w^{T} x_i + w_0)$
  - Si $y_i' \neq y_i$:

    $w^{(t+1)} \leftarrow w^{(t)} + r\,(y_i x_i)$

Slides from the previous course

El algoritmo del "perceptrón"

Entrada:

- una secuencia de pares de entrenamiento (x_1,y_1), (x_2, y_2) ...
- Una tasa de aprendizaje r (número pequeño y menor a 1)

Algoritmo:

- Inicializar $w^{(0)} \epsilon \mathbb{R}^n$
- Para cada ejemplo (x_i,y_i)
  - Predecir $y_i' = sign(w^T x_i)$
  - Si $y_i' \neq y_i$:

    $w^{(t+1)} \leftarrow w^{(t)} + r\,(y_i x_i)$

Actualiza solo cuando comete un error

Error en positivos:

$$w^{(t+1)} \leftarrow w^{(t)} + r\,x_i$$

Error en negativos:

$$w^{(t+1)} \leftarrow w^{(t)} - r\,x_i$$

Si $y_i w^T x_i \leq 0 \rightarrow$ error

Slides from the previous course

Demo Time (demo 1) …

Support Vector Machines

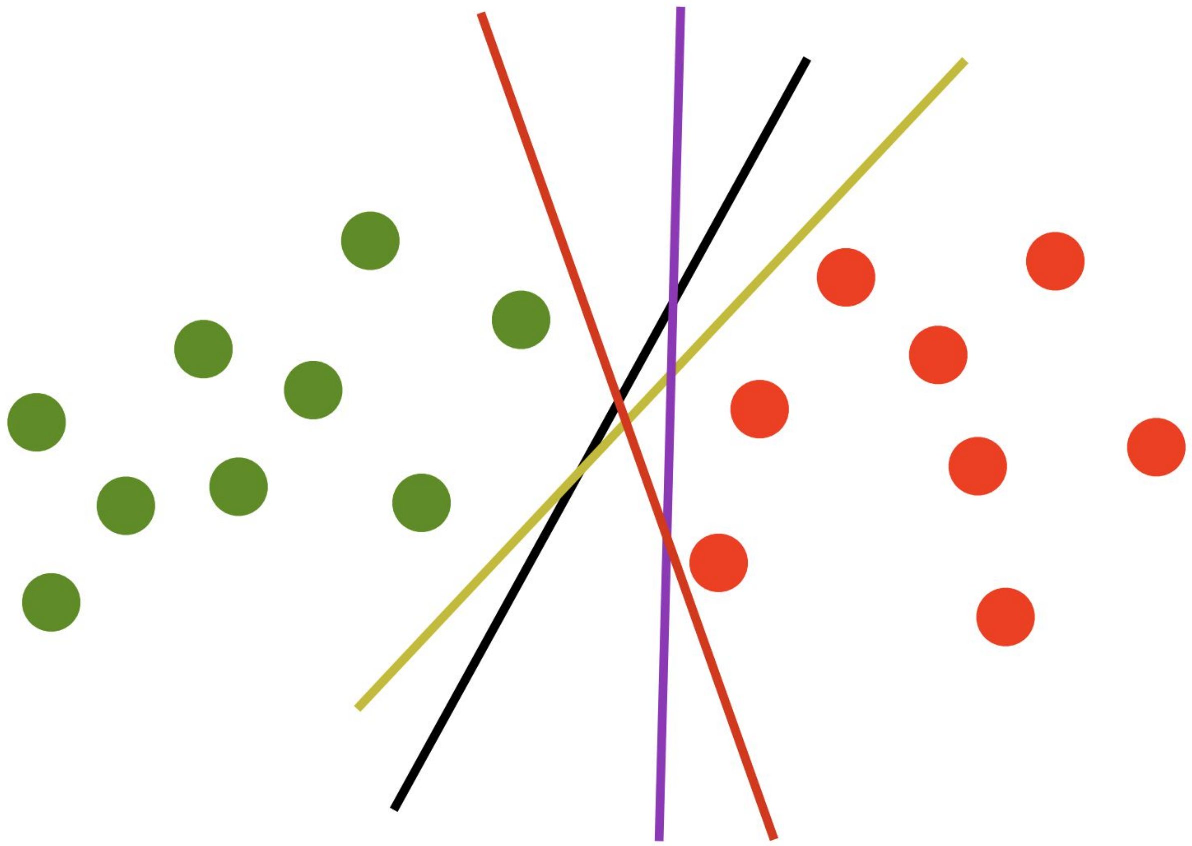


Image by Alex Smola

Support Vector Machines

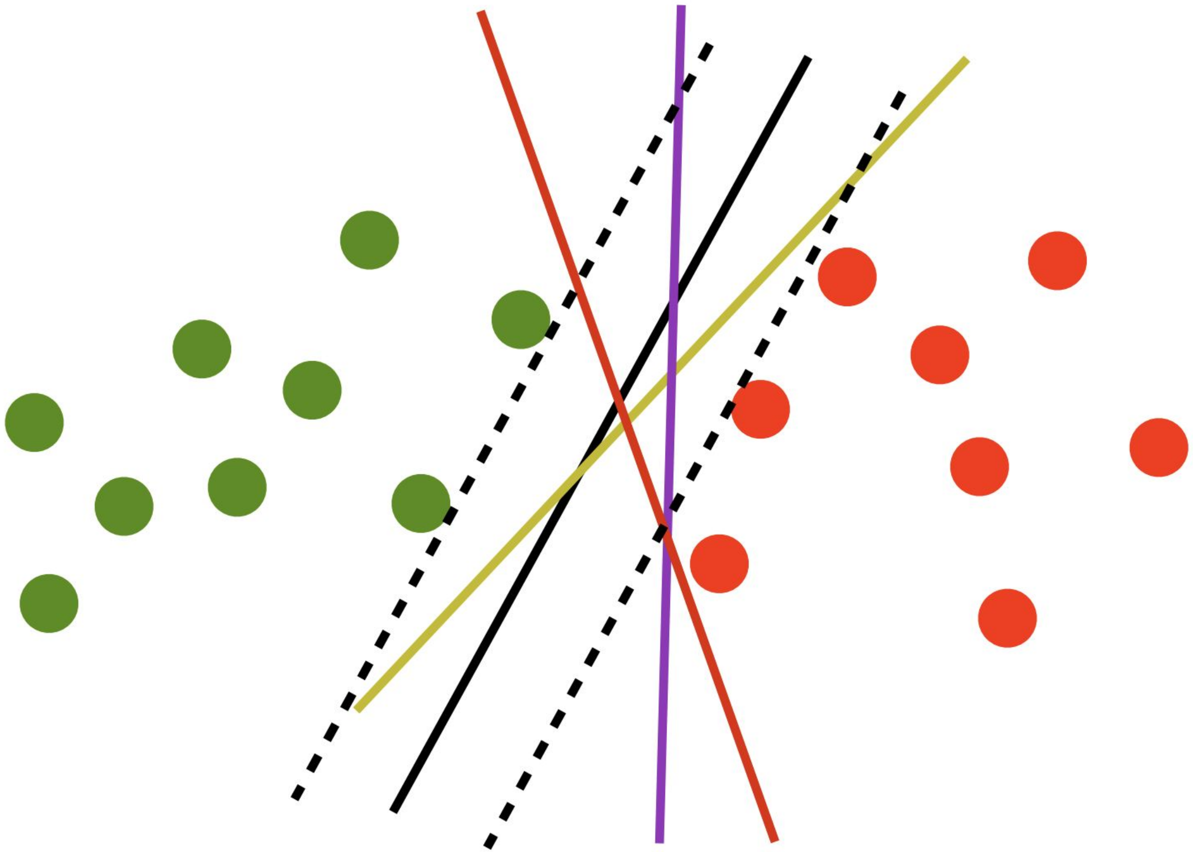


Image by Alex Smola

Support Vector Machines

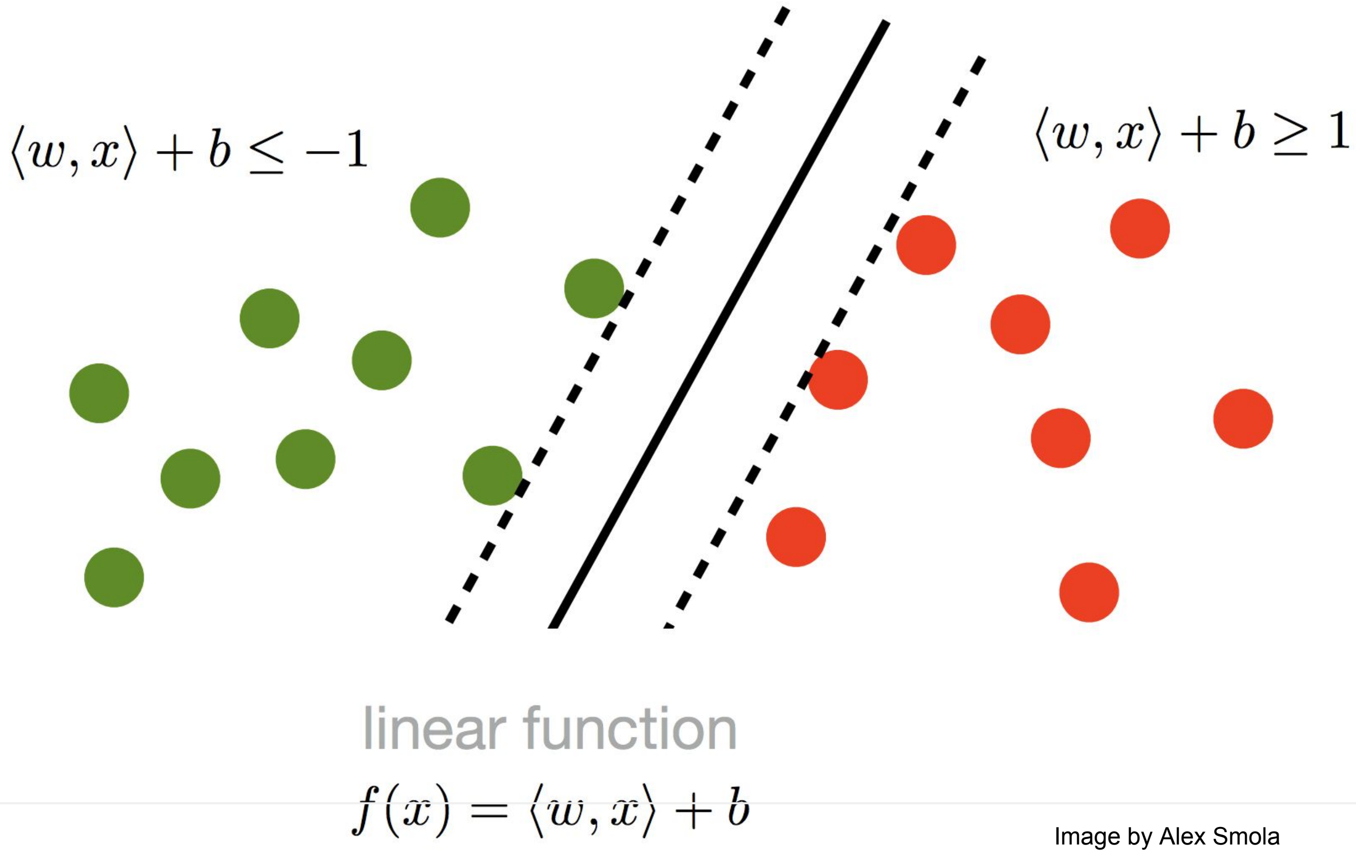


Image by Alex Smola

Whiteboard Time

•••

Support Vector Machines, Deriving the Equations

Support Vector Machines

Some Interesting Properties we have found:

- $w = \sum_i \alpha_i y_i x_i$
- Just the points on the margin used to define the hyperplane

Demo Time (demo 2)

•••

SVMs: slack variables

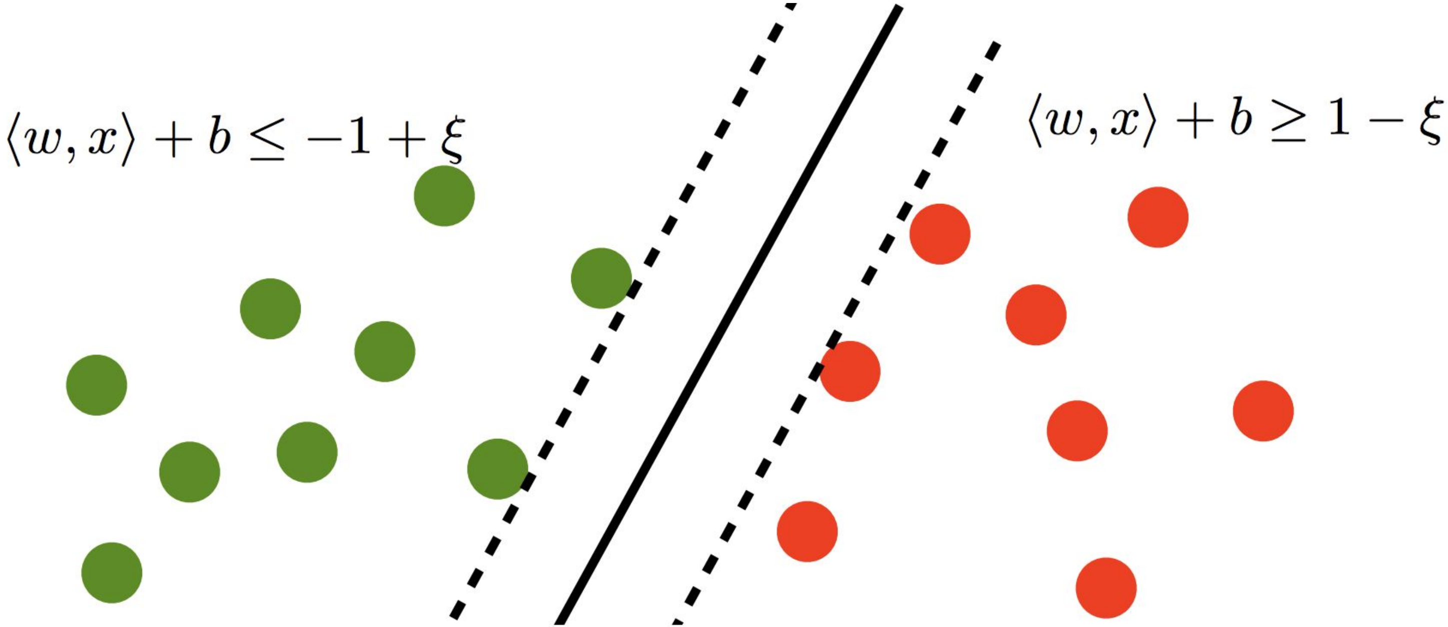


Image by Alex Smola

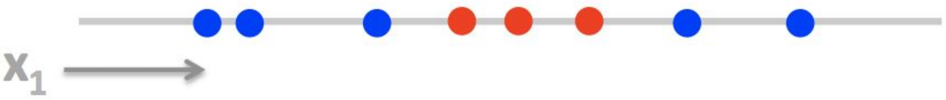
Demo Time (demo 3)

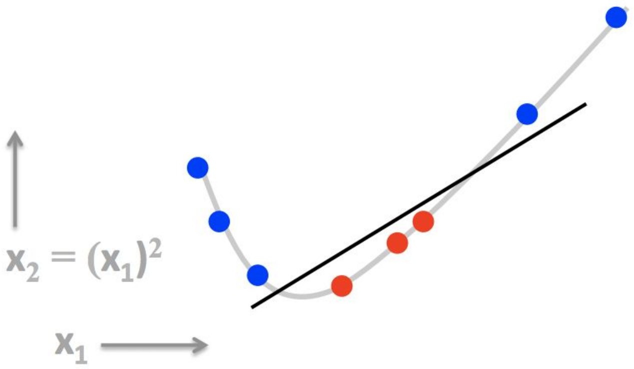
...

SVMs: Kernels

Images by Alex Ihler

SVMs: Kernels





Images by Alex Ihler

Whiteboard Time

...

SVMs: Kernels

Some common kernels:

- Polynomial
  $$K(x,z) = (1 + \sum_j x_j z_j)^d$$
- Radial Basis Functions (RBF aka Gaussian Similarity Functions)
  $$K(x,z) = \exp(-(x-z)^2/2\sigma^2)$$
- Sigmoid
  $$K(x,z) = \tanh(c < x,z > +h)$$
- Specific for certain types of problems

Demo Time (demo 4)

...

Multiclass SVMs

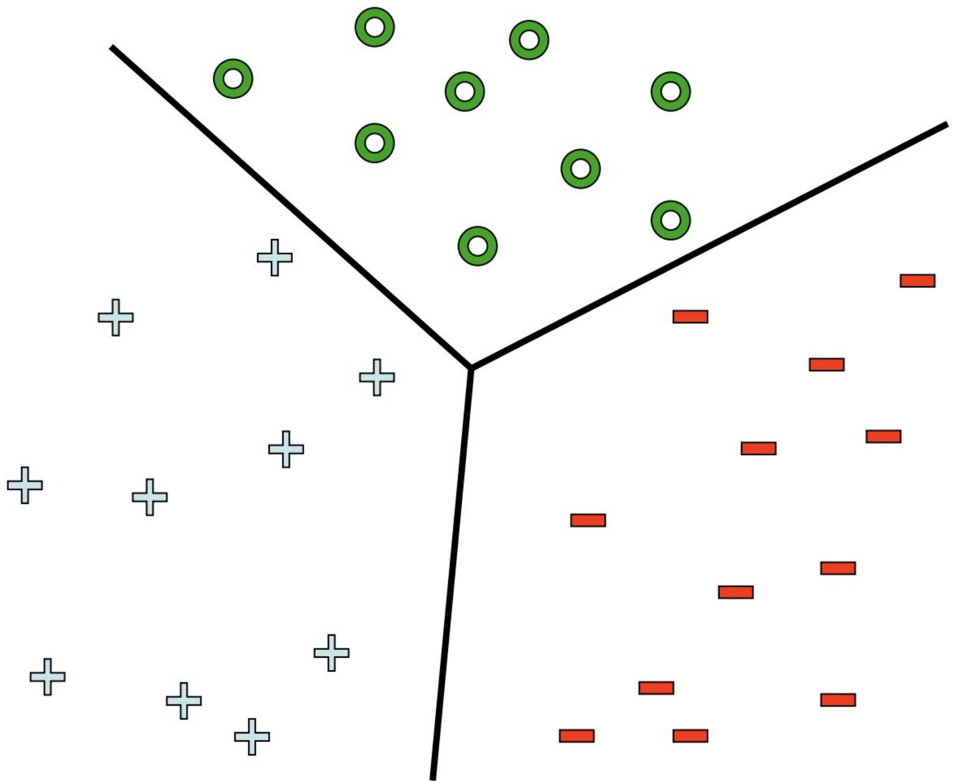


Image by David Sontag

Multiclass SVMs: one vs the rest (one vs all)

- **Training:** how could we do it?

Multiclass SVMs: one vs the rest (one vs all)

- **Training:** For M classes:

  construct a hyperplane between class k and the other M − 1 classes => M SVMs

- **Classification:** how could we do it?

Multiclass SVMs: one vs the rest

- **Training:** For M classes:

  construct a hyperplane between class k and the other $M - 1$ classes => M SVMs

- **Classification:** make M predictions (one for each SVMs) and find out the one getting more hits into its positive region.

Multiclass SVMs: one vs one (all vs all)

Multiclass SVMs: one vs one (all vs all)

- **Training:** For M classes:

  construct a hyperplane between class i and class j. => M(M-1)/2 SVMs

- **Classification:** If f_{ij} is the classifier where i are the positive samples and j the negative ones,, the classification of x is given by

$$f(x) = \text{argmin}_i \left(\sum_j f_{ij}(x)\right)$$

Support Vector Regressions

- Idea based on support vectors like in SVMs, but now y_i is a real number.
- Uses soft margins in the regression process instead of classification
- Additional parameter ε, to compute the loss function

Support Vector Regressions

- Idea based on support vectors like in SVMs, but now y_i is a real number.
- Uses soft margins in the regression process instead of classification
- Additional parameter ε, to compute the loss function
- Not frequently used, logistic regression is more popular

Whiteboard Time

•••

Decision Trees (review)

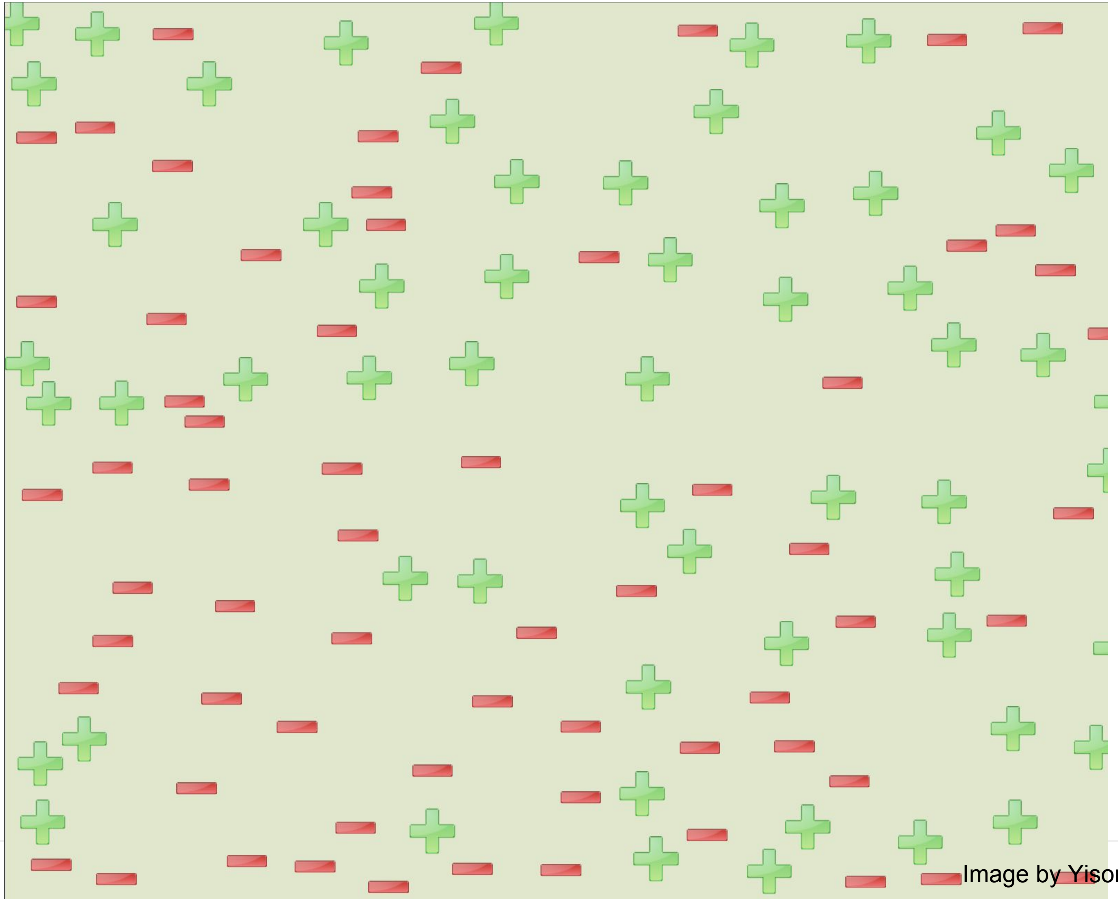


Image by Yisong Yue

Decision Trees (review)

Image by Yisong Yue

Decision Trees (review)

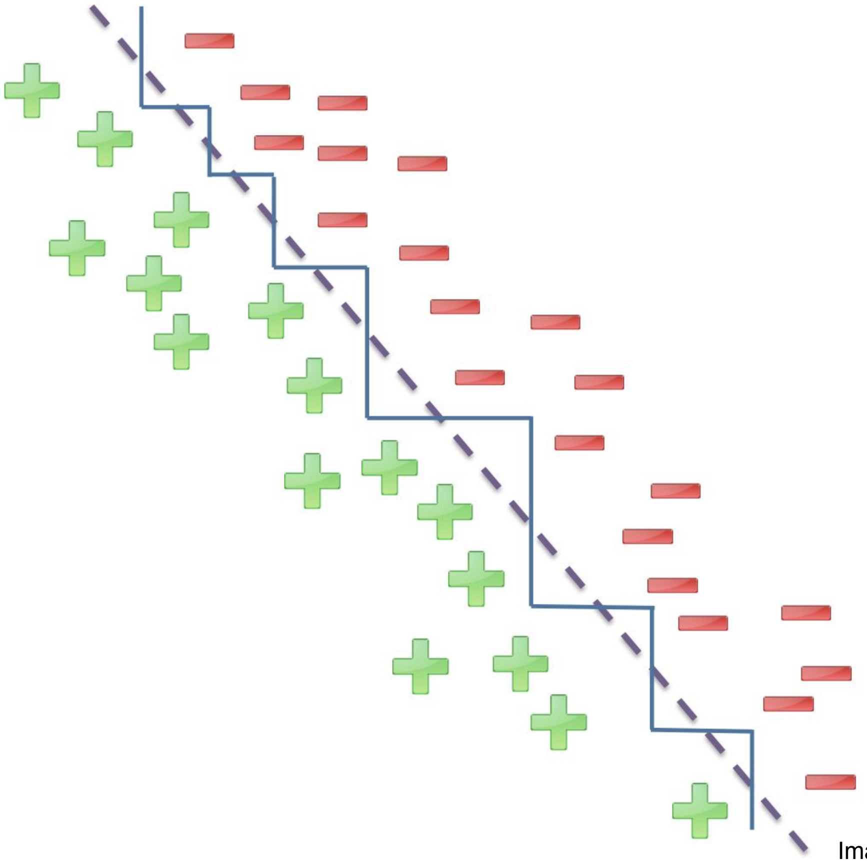


Image by Yisong Yue

Demo Time (demo 5)

…

Ensemble Learning

- Generate a set of "learners" that, when combined, have higher accuracy.
- Assuming we have three learners: L1, L2, L3
- The predictions from them may differ
- What would we do? Who do we trust?
    - Believe the model that we know is best?
    - Go with the majority?

Ensemble Learning

- **An ensemble model is a model that is a combination of several different models**
- **Usually, an ensemble is more accurate than all its constituent models**
- **Why?**
  - **Intuition:** "two know more than one"

Ensemble Learning. First approach: Voting

- Given n classifiers m1, m2, … mn
- Consider a new classifier M that, given a datum x, M computes m1(x), m2(x), …, counts the predictions and returns the most predicted class
- How well would M work?

Ensemble Learning. First approach: Voting

- To answer the question, let's make some assumptions:
    - All the classifiers m1, …, mn are equally accurate (with p being their accuracy)
    - The errors in the classification made by each classifier are independent
        - P(mj wrong | mk wrong) = P(mj wrong)
- How well would M work?

Ensemble Learning. First approach: Voting

- **Example**
  - **p = 0.8**
  - **n = 5**
  - **P(majority of 5 models is correct) =**

$$= \binom{5}{5}0.8^5 0.2^0 + \binom{5}{4}0.8^4 0.2^1 + \binom{5}{3}0.8^3 0.2^2$$
$$= 0.942$$

- **Need empirical validation if assumptions not valid**

Ensemble Learning. First approach: Voting

- **Example**
  - **p = 0.8**
  - **n = 5**
  - **P(majority of 5 models is correct) =**

$$
\begin{aligned}
&= \binom{5}{5} 0.8^5 0.2^0 + \binom{5}{4} 0.8^4 0.2^1 + \binom{5}{3} 0.8^3 0.2^2 \\
&= 0.942
\end{aligned}
$$

- **Need empirical validation if assumptions not valid**

Ensemble Learning

- We know how to build decision trees
- How can we build different trees for the same data?
  - This may result on the same tree
- Do we introduce some variations?

Bagging

- Let D be the dataset
- Repeat k times:
  - Create D' from D by randomly selecting |D| instances of D with replacement
  - Learn a new model m
- Return a model that selects the most frequent prediction among m1, …, mk predictions

Bagging

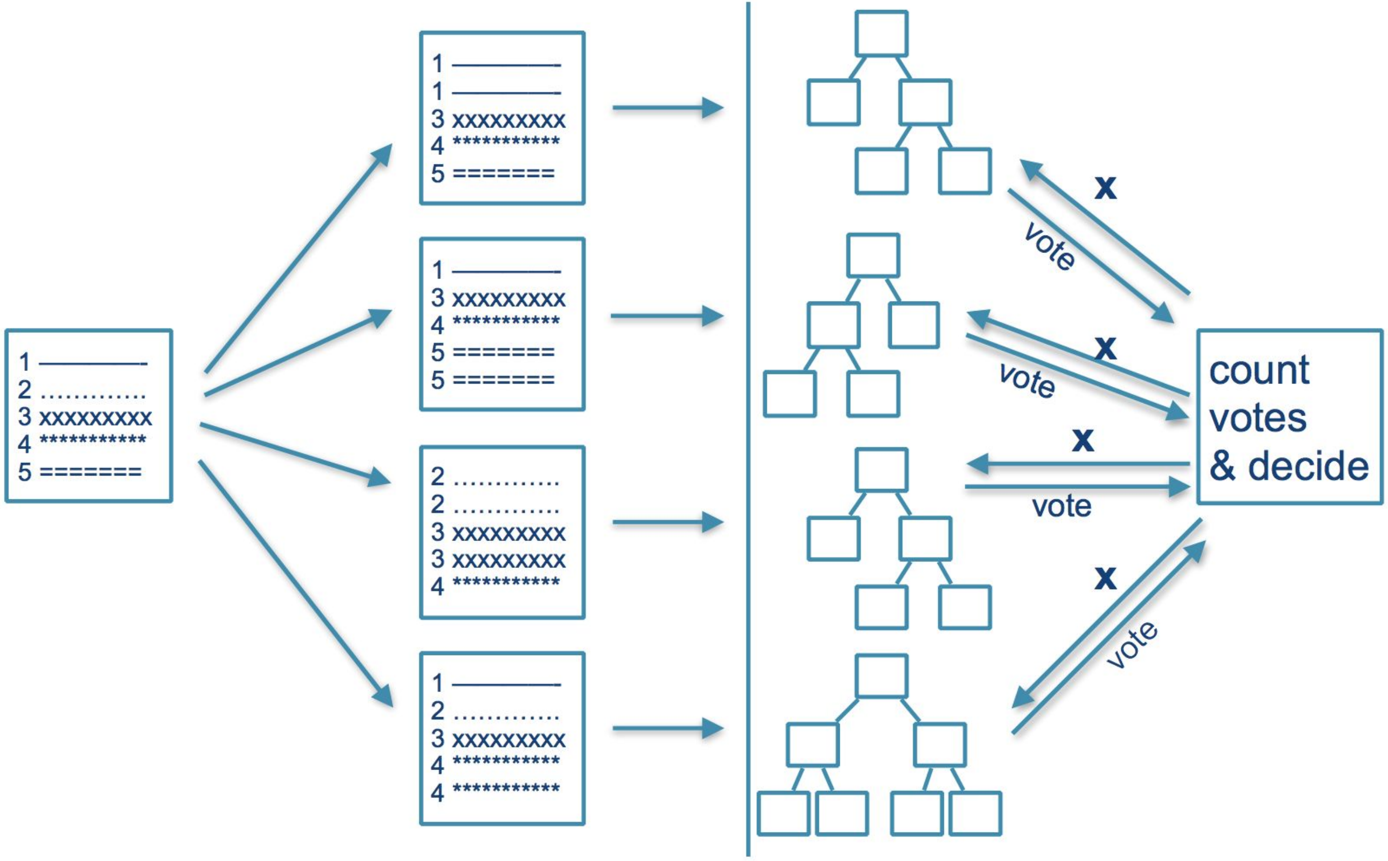


Image by Hendrik Blockeel

Bagging for Decision Trees

- Bagging generally works well for unstable learners
  - A learner is unstable if small changes in the dataset can give very different resulting models
  - It turns out that decision tree learners are indeed unstable

- Disadvantage: learning k trees is k times as expensive as learning one tree

Random Forests

- Like bagging, with one improvement
  - For trees, ALL the features are considered to create a split node (inner node)
  - For random forests, at each node consider only M randomly chosen attributes (not all)
  - Usually take $M = \sqrt{\text{number of attributes}}$

Random Forests

Common Steps

- Build a random forest considering M attributes
- Predict the value of "Out-of-bag" samples using the random forest
- Estimate the accuracy
- Determine the optimal M (hyperparameter)

Random Forests

- **Random Forests is one of the most efficient and most accurate learning methods to date (2008)** (Caruana+: An empirical evaluation of supervised learning in high dimensions. ICML 2008)

- **Easy to use with little parameter tuning**

- **Easy to debug, but, compared with Decision Trees, the model is less interpretable**

Demo Time (demo 6)

...

Boosting

- **Bagging goal:** fit large trees to resampled versions of the training data, and classify by majority vote

- **Boosting:** fit large or small trees to "**reweighted**" versions of the training data and classify by **weighted** majority vote

Boosting

- **Each model, defines the features that the next model will focus on**
- **Uses bootstrapping like bagging, but here we weight each sample of data**
    - **Some samples will be used more frequently**
- **Process:**
    - Given a model, track the samples that are more "erroneous" and give them heavier weights (considered to be data that have more complexity and requires more steps)
    - Given a model, track the error rate so that better models are given more weights

Neural Networks Warm-up

Why neural?

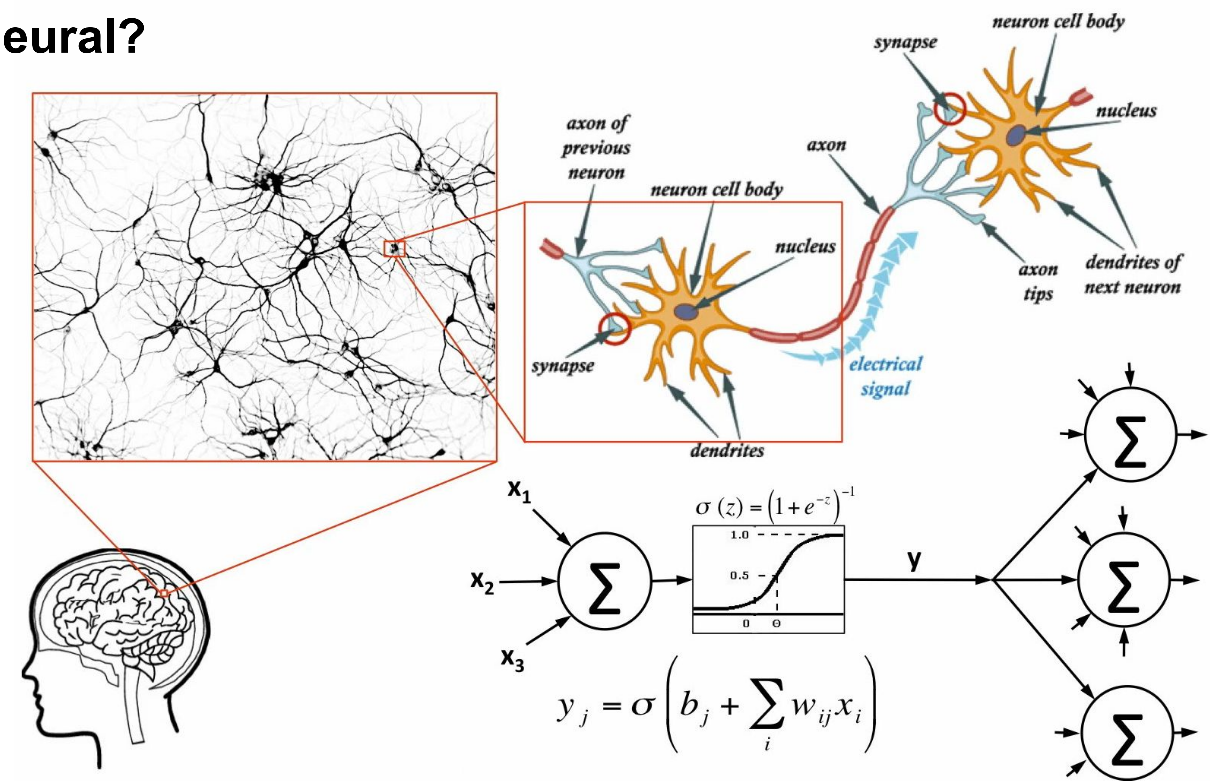


Image by Victor Lavrenko

Neural Networks Warm-up

Logistic Regression Review

Given x, we would like to find $\hat{y} = P(y = 1|x)$

What is the easiest way to transform x?

Neural Networks Warm-up

Logistic Regression Review

Given x, we would like to find $\hat{y} = P(y = 1|x)$

What is the easiest way to transform x?

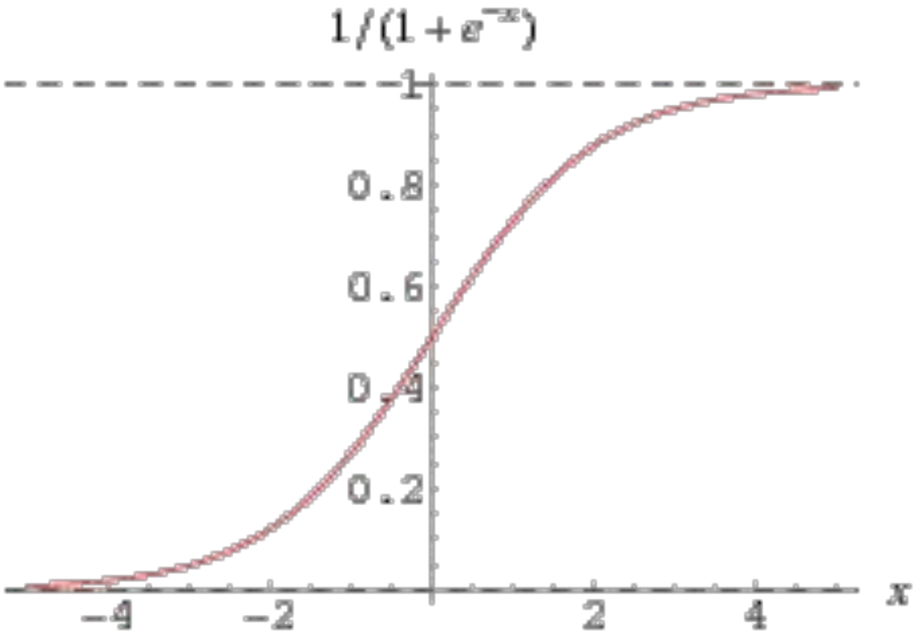
$$\hat{y} = w^T x + b$$

But we would like ŷ to be a probability: $0 \leq \hat{y} \leq 1$

Neural Networks Warm-up

Sigmoid function

$$f(x) = \frac{1}{1+\exp(-x)}$$



Neural Networks Warm-up

Cost function (for Logistic Regression)

- **m training samples:** $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots (x^{(m)}, y^{(m)})\}$
- **want to find:** $\hat{y}^{(i)} \approx y^{(i)}$
- **Popular loss function:**

$$\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

Neural Networks Warm-up

Cost function (for Logistic Regression)

- **m training samples:** $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots (x^{(m)}, y^{(m)})\}$
- **want to find:** $\hat{y}^{(i)} \approx y^{(i)}$
- **~~Popular~~ loss function:**
$$\mathcal{L}(\hat{y}, y) = -[y \log \hat{y} + (1-y) \log (1-\hat{y})]$$
- **Cost function:**
$$\mathcal{J}(w, b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

Neural Networks Warm-up

Logistic Regression

$$z = w^T x + b$$

$$\hat{y} = \sigma(w^T x + b)(\text{activation function; we usually use } a \text{ instead of } \hat{(}y)$$

$$\mathcal{L}(a, y) = -[y \log a + (1 - y) \log (1 - a)]$$

The goal is to minimize $\mathcal{J}$, to do so we compute

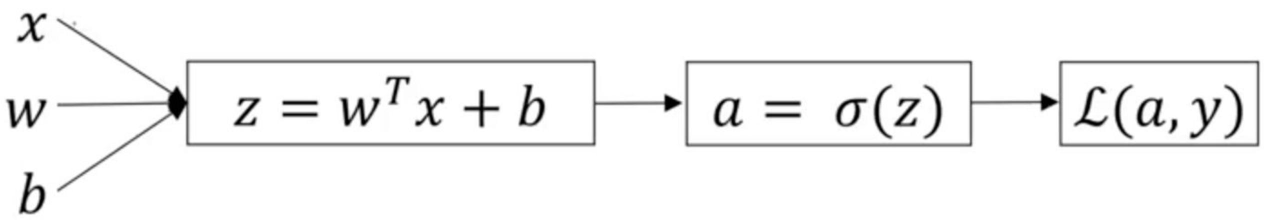
$$\frac{\partial}{\partial w_1} \mathcal{J}(w, b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial w_1} \mathcal{L}(a^{(i)}, y^{(i)})$$

Whiteboard Time

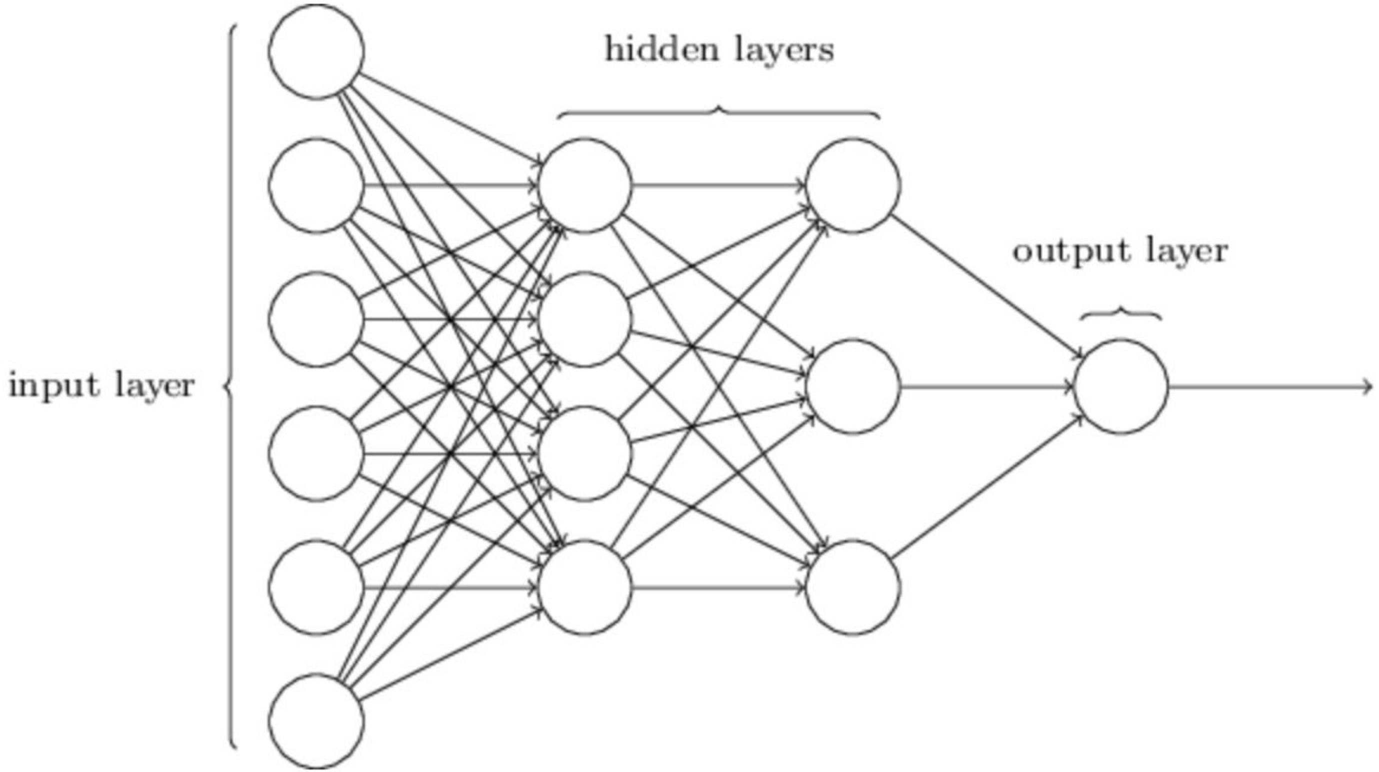
Computation graph for backpropagation
&
Logistic Regression minimization

Neural Networks Warm-up

Logistic Regression Gradient Descent



Neural Networks



Neural Networks

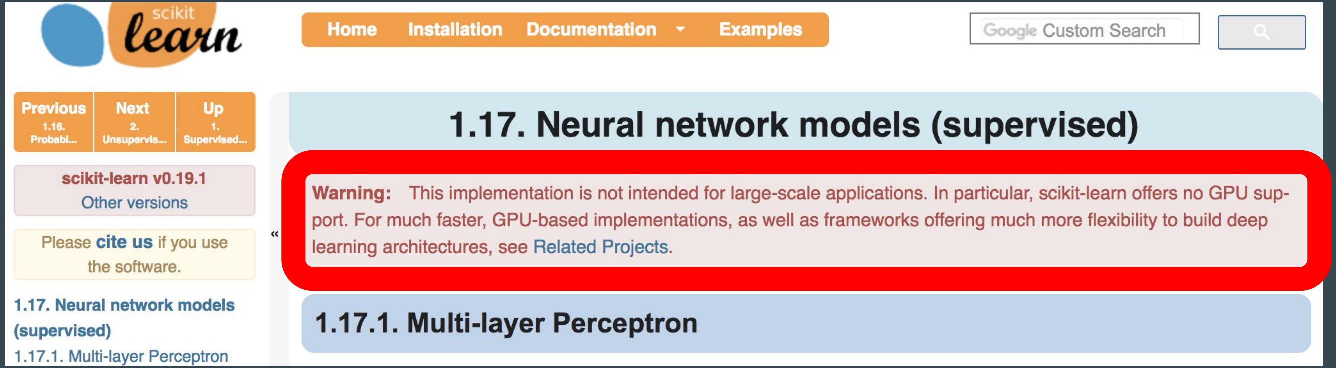
Activation Functions

- **Sigmoid function (as in Logistic regression)**
- **tanh:** $\frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$
- **Rectified Linear Unit (ReLU):** $\max(0, x)$
- **Leaky ReLU**

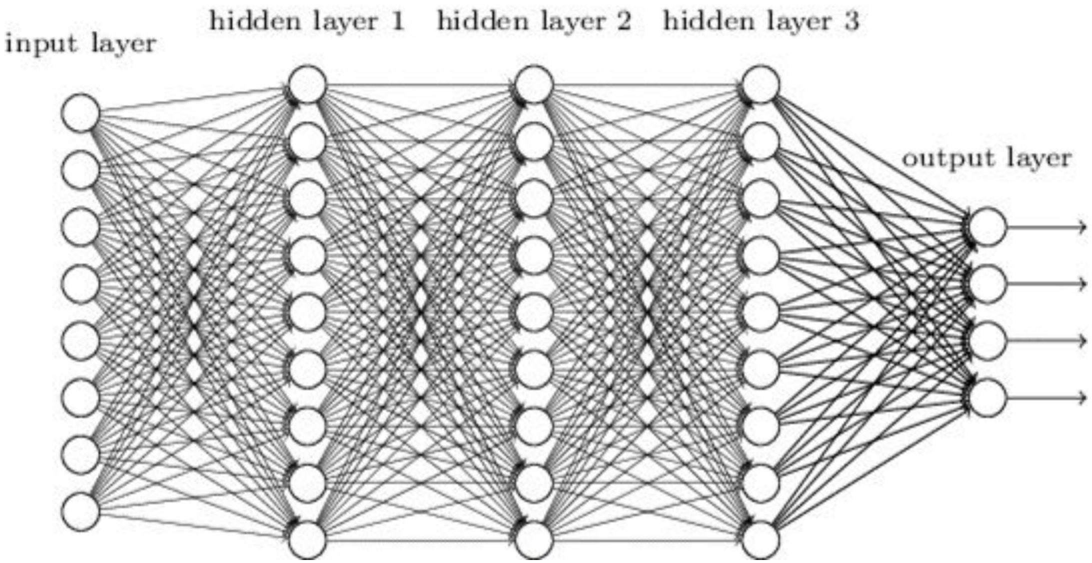
Demo Time (demo 7)

...

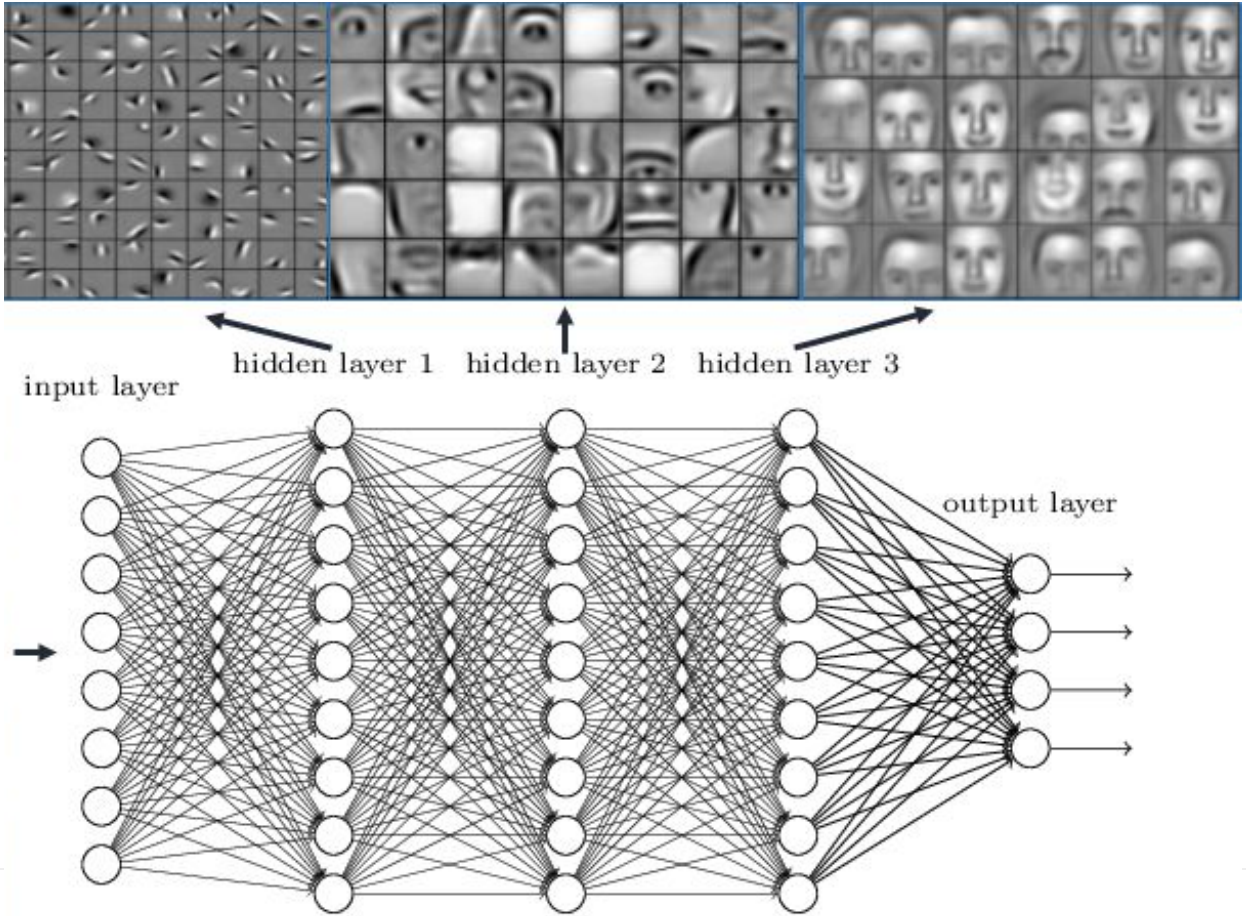
Demo Time (demo 7) …



Deep Neural Networks



Deep Neural Networks



Deep Neural Networks

How to split the data?

- **Train / Test / Validation**

Deep Neural Networks

How to split the data?

- **Train / Test / Validation**

- **Now?**

Deep Neural Networks

How to split the data?

- **Train / Test / Validation**

- **Now?**
  - **Too much data (> 10.000.000 samples)**

Deep Neural Networks

How to split the data?

- **Train / Test / Validation**

- **Now?**
  - **Too much data (> 10.000.000 samples)**

Make sure your train/ test / validation come from the same distribution

Deep Neural Networks

Rule of thumb to deal with bias/variance?

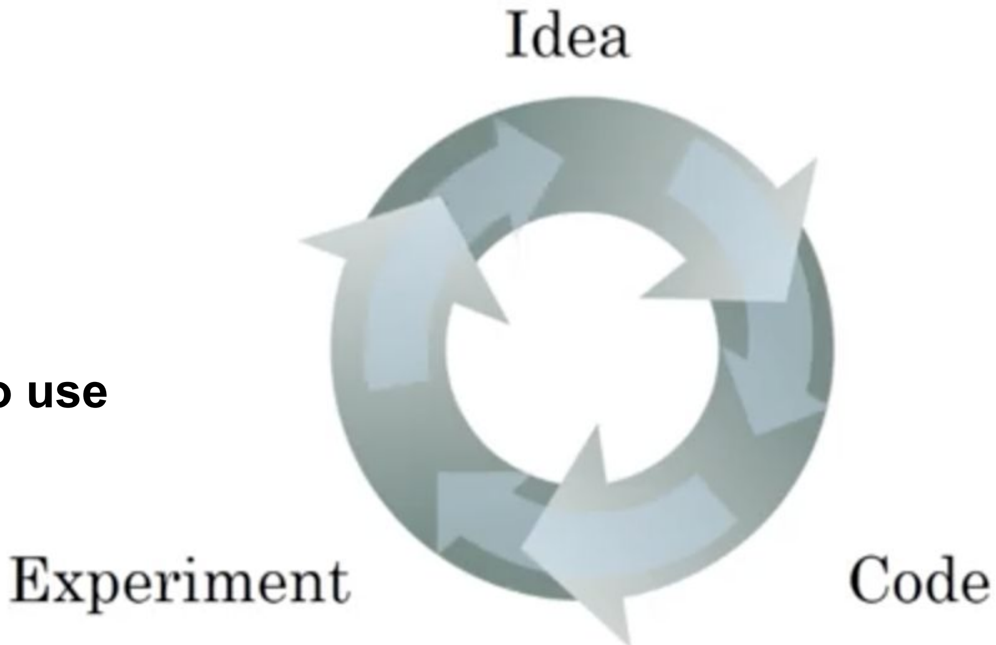
- **High Bias:**
  - **Bigger Network**
  - **Different Network Architecture**

- **High Variance:**
  - **More data**
  - **Regularization**
  - **Different Network Architecture**

Neural Networks

How to decide the size?

- **# hidden layers**
- **# hidden units**
- **What activation function to use**



Neural Networks Regularization

Logistic regression:

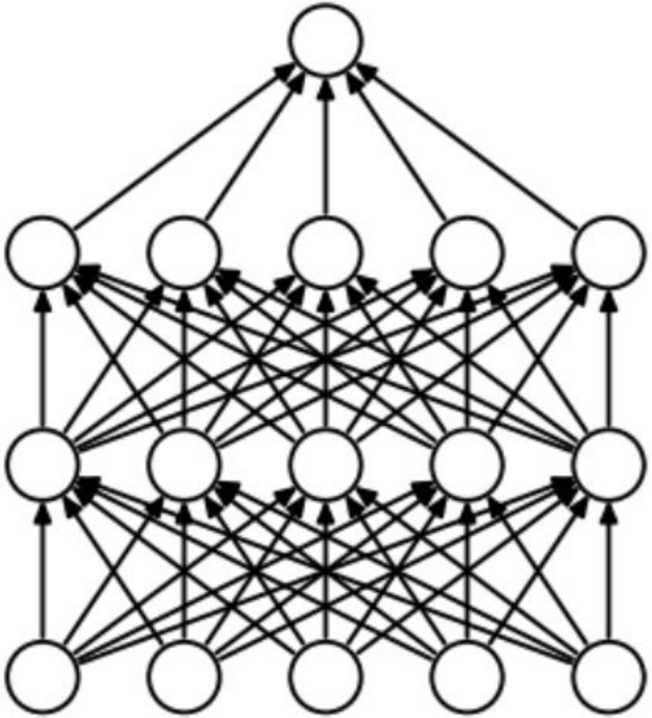
$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Neural network:

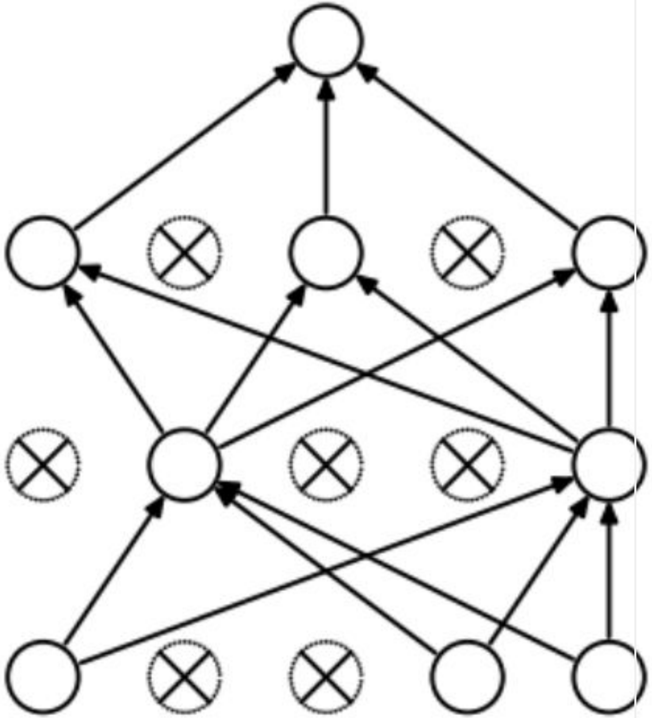
$$h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}$$

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)}\log(h_\Theta(x^{(i)}))_k + (1-y_k^{(i)})\log(1-(h_\Theta(x^{(i)}))_k)\right]$$
$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

Neural Networks Dropout



(a) Standard Neural Net



(b) After applying dropout.

Neural Networks Momentum

- **Mini-batch Gradient Descent**

- **Gradient Descent optimization**
  - **with Momentum**
  - **RMSprop**
  - **Adam**

Resources (shown in class)

- **Multiclass classification (graphs):** https://jermwatt.github.io/mlrefined/blog_posts/Linear_Supervised_Learning/Part_5_One_versus_all.html
- **SVMs on fraud detection:** https://www.kaggle.com/pierra/credit-card-dataset-svm-classification
- **SVM kernels:** https://www.youtube.com/watch?v=3liCbRZPrZA
- **Random Forests on Titanic:** https://www.kaggle.com/zlatankr/titanic-random-forest-82-78/notebook
- **AdaBoost:** (slides 13 to 23) http://media.nips.cc/Conferences/2007/Tutorials/Slides/schapire-NIPS-07-tutorial.pdf
- **Computation graphs video:** (minute 6:00) https://www.youtube.com/watch?v=u2OeYrlAx_A&t=836s
- **CNNs vizualization:** (someone asked in class about this, the one below is one I found, not the one I had in mind, but it should be similar) https://github.com/InFoCusp/tf_cnnvis
- **Pydata :)** https://pydata.org/cordoba2018/