Outline:

- Machine learning models
- The use of ML in neural engineering
- Homework

Recap: supervised learning

- Training Data: $S = \{(x_i, y_i)\}_{i=1}^{N}$ $\quad x \in R^D$ $\quad y \in \{-1, +1\}$
- Model Class: $f(x \mid w, b) = w^T x - b$ **Linear Models**
- Loss Function: $L(a, b) = (a - b)^2$ **Squared Loss**
- Learning Objective:

$$\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{N} L\big(y_i, f(x_i \mid w, b)\big)$$

Optimization Problem

Recap: Basic Recipe

$$S = \{(x_i, y_i)\}_{i=1}^{N}$$

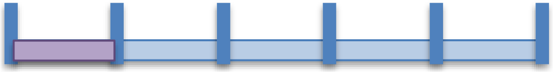
Training Data

$$f(x \mid w, b) = w^T x - b$$

Model Class(es)

$$L(a, b) = (a - b)^2$$

Loss Function

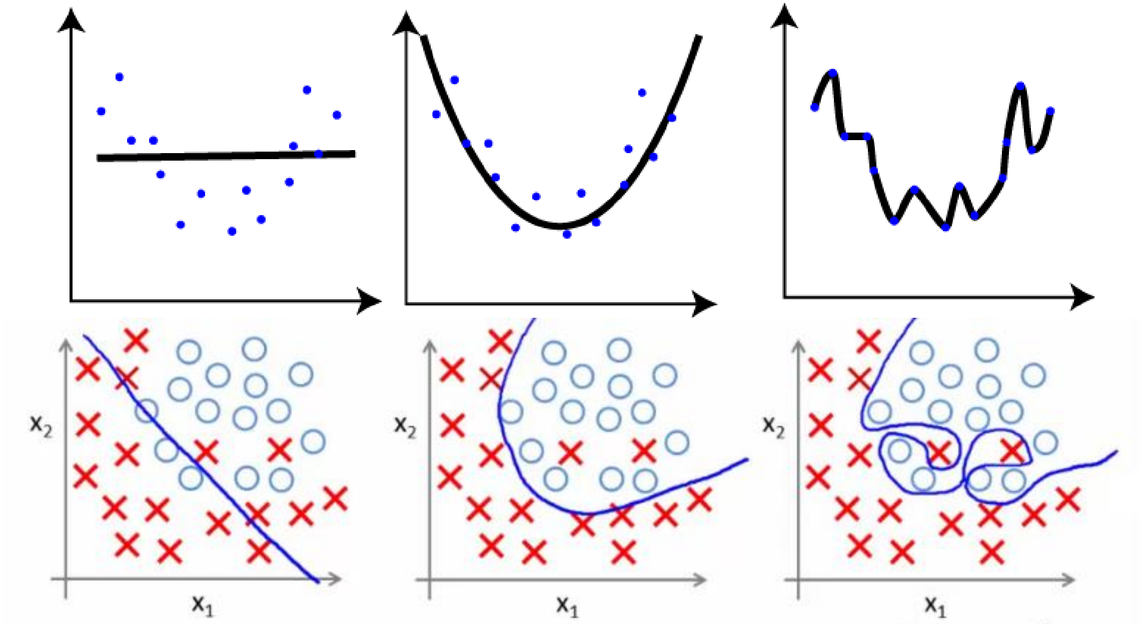


$$\operatorname*{argmin}_{w,b} \sum_{i=1}^{N} L\big(y_i, f(x_i \mid w, b)\big)$$

Cross Validation & Model Selection

Optimization Problem

Overfitting v. Underfitting



$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(g = sigmoid function)

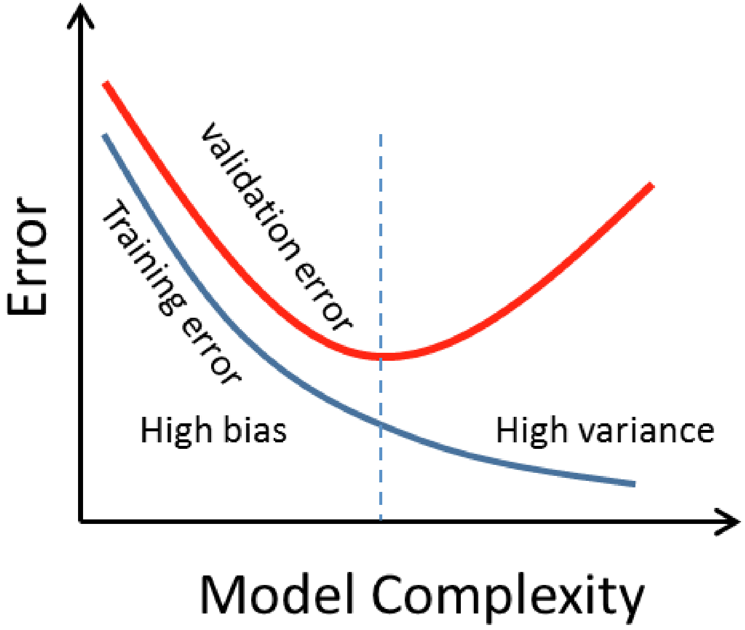
UNDERFITTING
(high bias)

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2$
$+\theta_3 x_1^2 + \theta_4 x_2^2$
$+\theta_5 x_1 x_2)$

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2$
$+\theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2$
$+\theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \ldots$

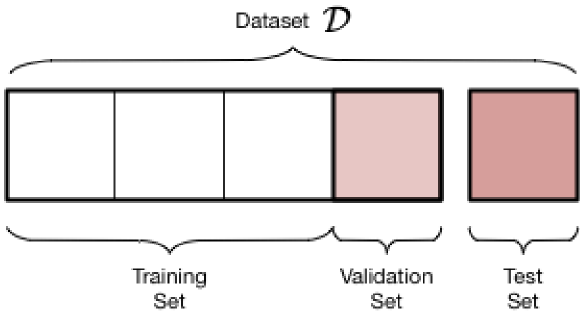
OVERFITTING
(high variance)

How not to overfit



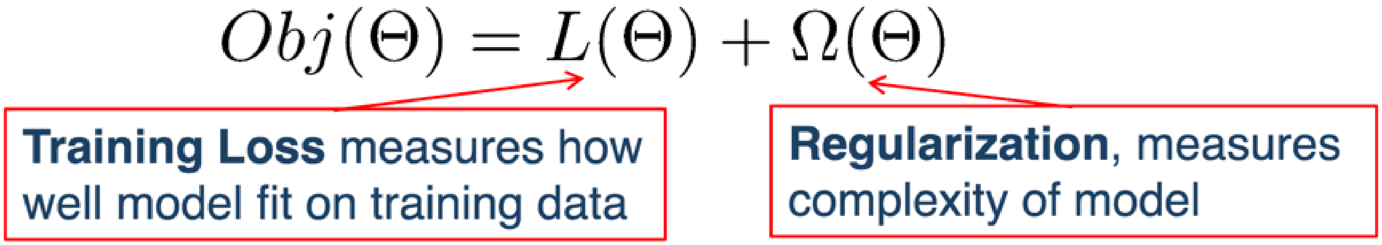
Two cures:

- Regularization: putting brakes
- Validation: checking the bottom line



Recap: Model training

Objective function



$$Obj(\Theta) = L(\Theta) + \Omega(\Theta)$$

Training Loss measures how well model fit on training data

Regularization, measures complexity of model

Loss on training data: $L = \sum_{i=1}^{n} l(y_i, \hat{y}_i)$

Square loss: $l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$

Logistic loss: $l(y_i, \hat{y}_i) = y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})$

Regularization: how complicated the model is?

L2 norm: $\Omega(w) = \lambda \|w\|^2$

L1 norm (lasso): $\Omega(w) = \lambda \|w\|_1$

Logistic Regression

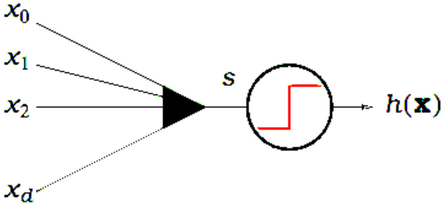
aka “Log-Linear”

Linear models

$$s = \sum_{i=0}^{d} w_i x_i$$

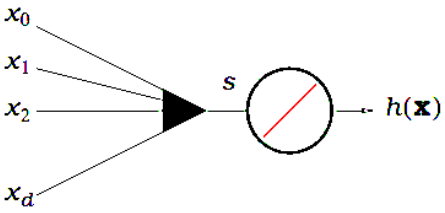
linear classification

$$h(\mathbf{x}) = \text{sign}(s)$$



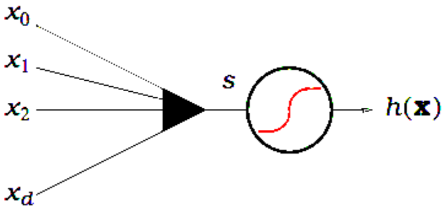
linear regression

$$h(\mathbf{x}) = s$$

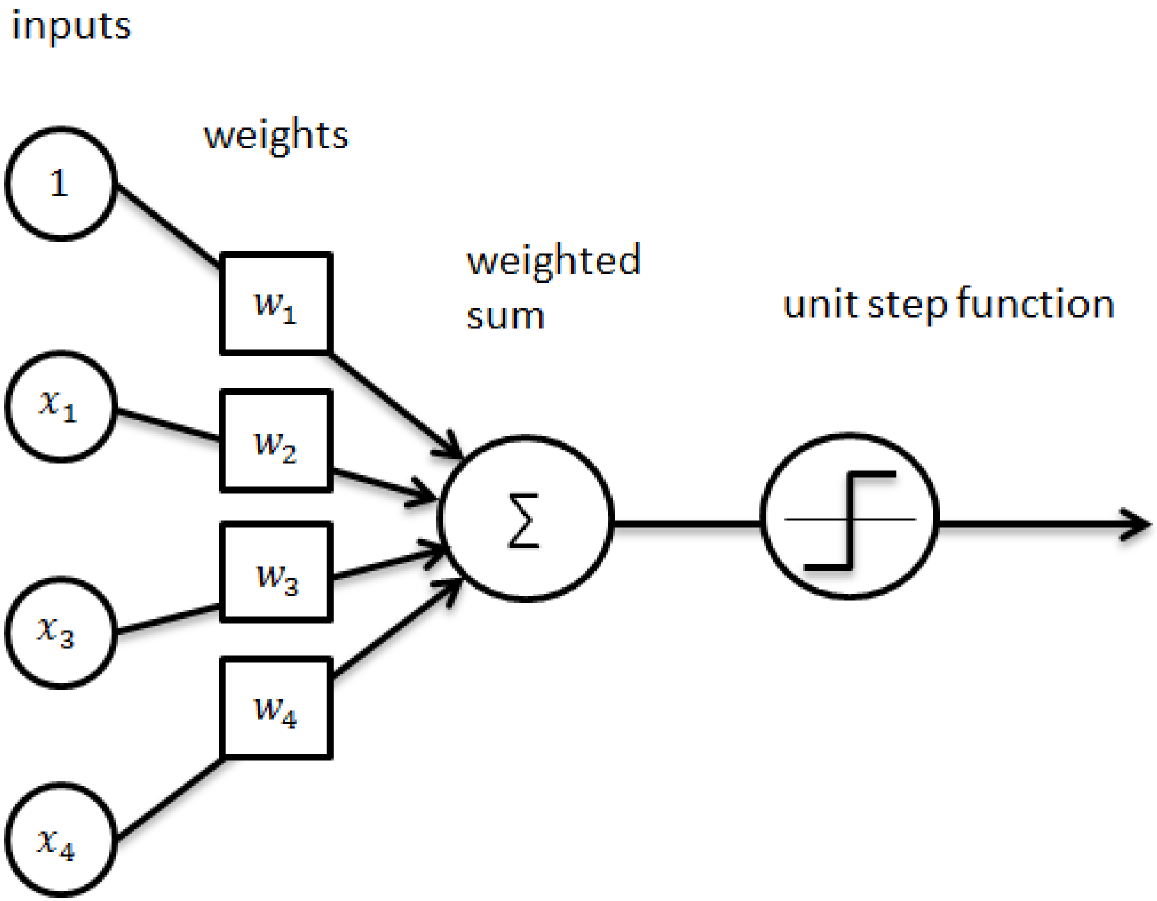


logistic regression

$$h(\mathbf{x}) = \theta(s)$$



Linear Model--Perceptron



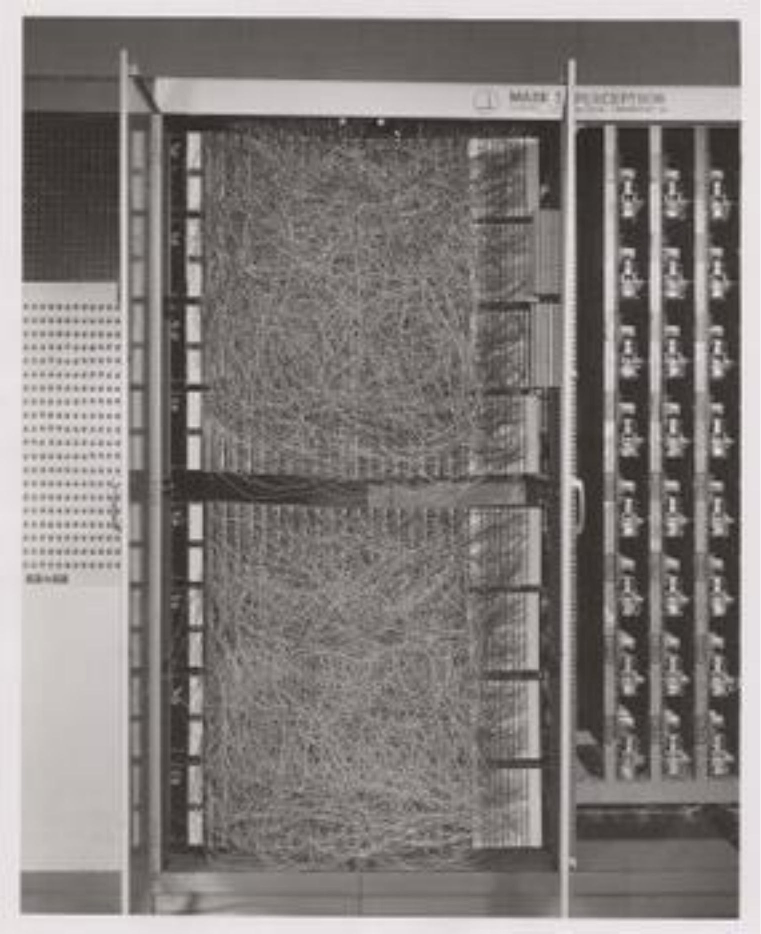
inputs

weights

weighted sum

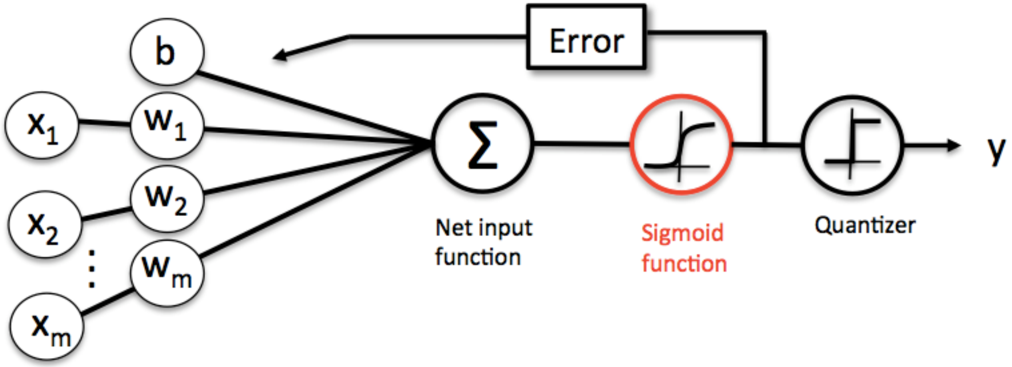
unit step function

First ML Hardware!

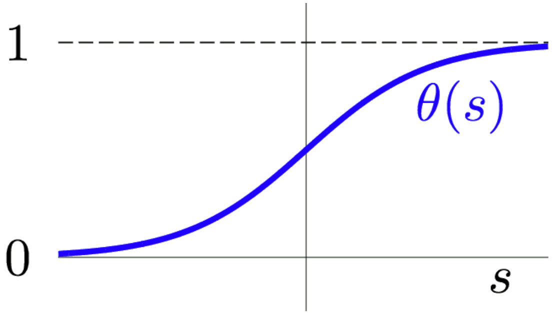


Frank Rosenblatt, 1957
Mark I Perceptron at the Cornell Aeronautical Laboratory, hardware implementation of the first Perceptron

Logistic Regression



$$\theta(s) = \frac{e^s}{1 + e^s}$$



- sigmoid: soft threshold (uncertainty)
- h(x) is interpreted as probability

Maximum Likelihood Training

- Training set: $S = \{(x_i, y_i)\}_{i=1}^{N}$ $\quad x \text{ Î } R^D$ $\quad y \text{ Î } \{-1, +1\}$

- Maximum Likelihood:

$$\underset{w,b}{\text{argmax}} \underset{i}{\text{Õ}} P(y_i \mid x_i, w, b)$$

- Each (x,y) in S sampled independently!

Log Loss

$$P(y \mid x, w, b) = \frac{e^{\frac{1}{2}y\left(w^T x - b\right)}}{e^{\frac{1}{2}y\left(w^T x - b\right)} + e^{-\frac{1}{2}y\left(w^T x - b\right)}} = \frac{e^{\frac{1}{2}yf(x|w,b)}}{e^{\frac{1}{2}yf(x|w,b)} + e^{-\frac{1}{2}yf(x|w,b)}}$$

$$\underset{w,b}{\operatorname{argmax}} \prod_i P(y_i \mid x_i, w, b) = \underset{w,b}{\operatorname{argmin}} \sum_i \underbrace{-\ln P(y_i \mid x_i, w, b)}_{\textbf{Log Loss}}$$

$$L(y, f(x)) = -\ln\left(\frac{e^{\frac{1}{2}yf(x)}}{e^{\frac{1}{2}yf(x)} + e^{-\frac{1}{2}yf(x)}}\right)$$

Solve using
Gradient Descent

Support Vector Machines

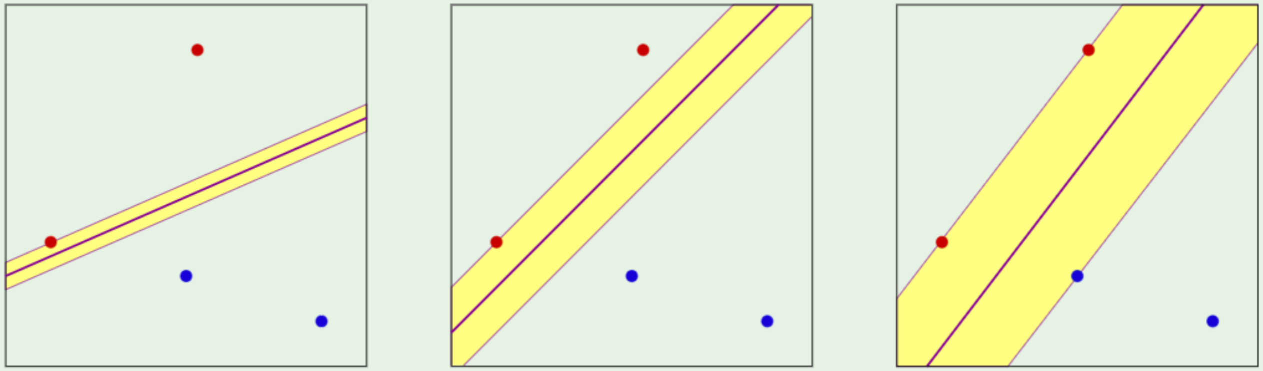
aka Max-Margin Classifiers

Better linear separation

Linearly separable data

Different separating lines

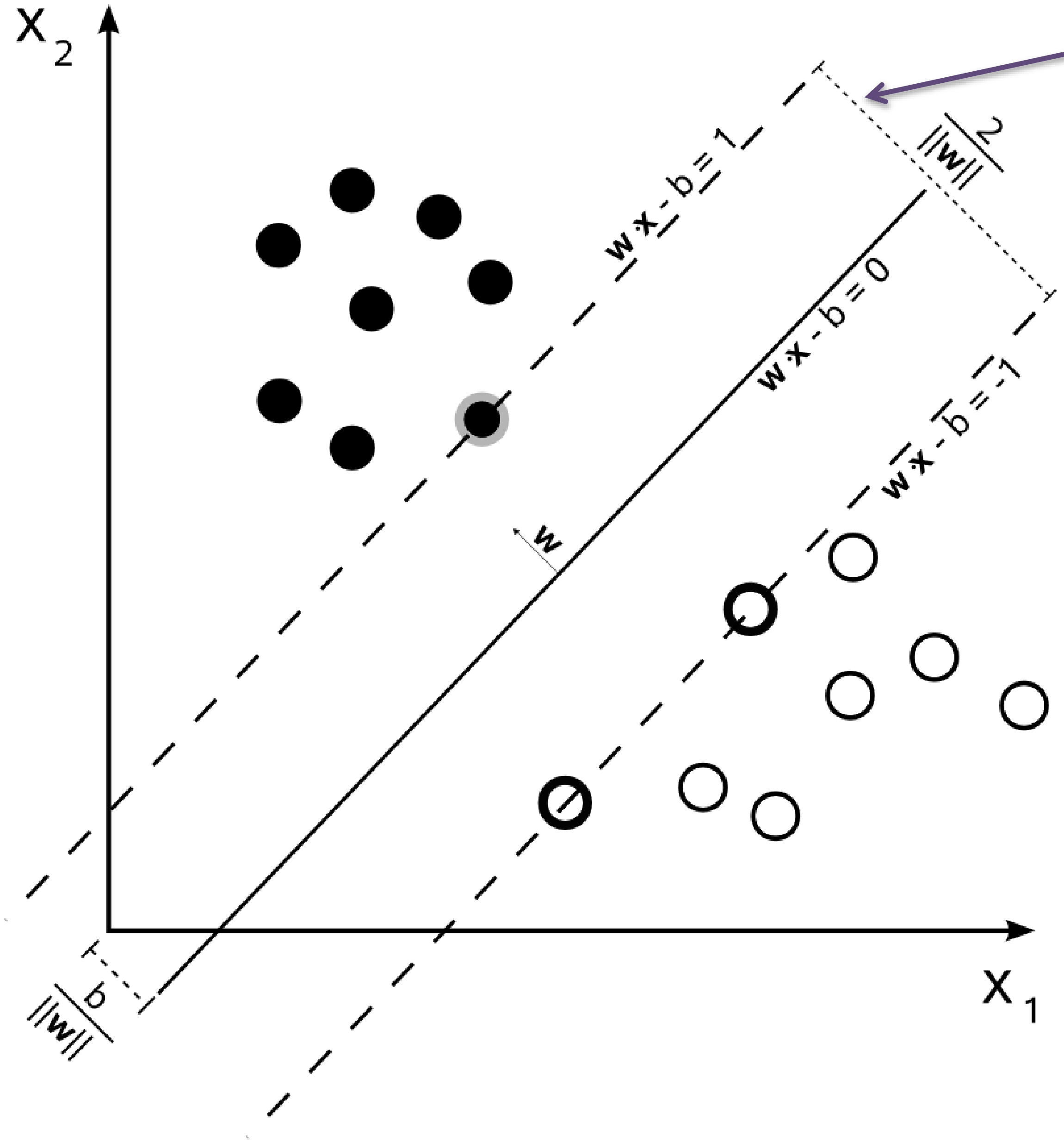
Which is best?



Two questions:

1. Why is bigger margin better?
2. Which $\mathbf{w}$ maximizes the margin?

Max Margin Classifier (Support Vector Machine)



“Margin”

$$\operatorname*{argmin}_{w,b} \frac{1}{2} w^T w \text{ o } \frac{1}{2}\|w\|^2$$

$$\text{"}\ i : y_i\left(w^T x_i - b\right) {}^{3}\ 1$$

Better generalization
to unseen test examples
(beyond scope of course*)

Linearly Separable

The optimization problem

Maximize $\frac{1}{\|\mathbf{w}\|}$

subject to $\min_{n=1,2,\ldots,N} |\mathbf{w}^{\intercal}\mathbf{x}_n + b| = 1$

Notice: $|\mathbf{w}^{\intercal}\mathbf{x}_n + b| = y_n(\mathbf{w}^{\intercal}\mathbf{x}_n + b)$

Minimize $\frac{1}{2}\mathbf{w}^{\intercal}\mathbf{w}$

subject to $y_n(\mathbf{w}^{\intercal}\mathbf{x}_n + b) \geq 1$ for $n = 1, 2, \ldots, N$

Support vectors

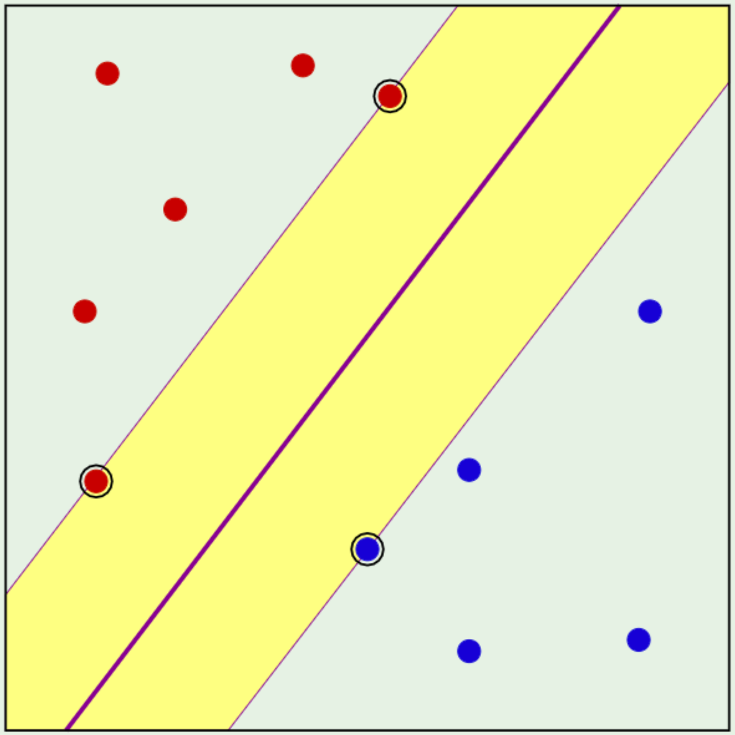
Closest $\mathbf{x}_n$'s to the plane: achieve the margin

$$\Longrightarrow \quad y_n\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b\right) = 1$$

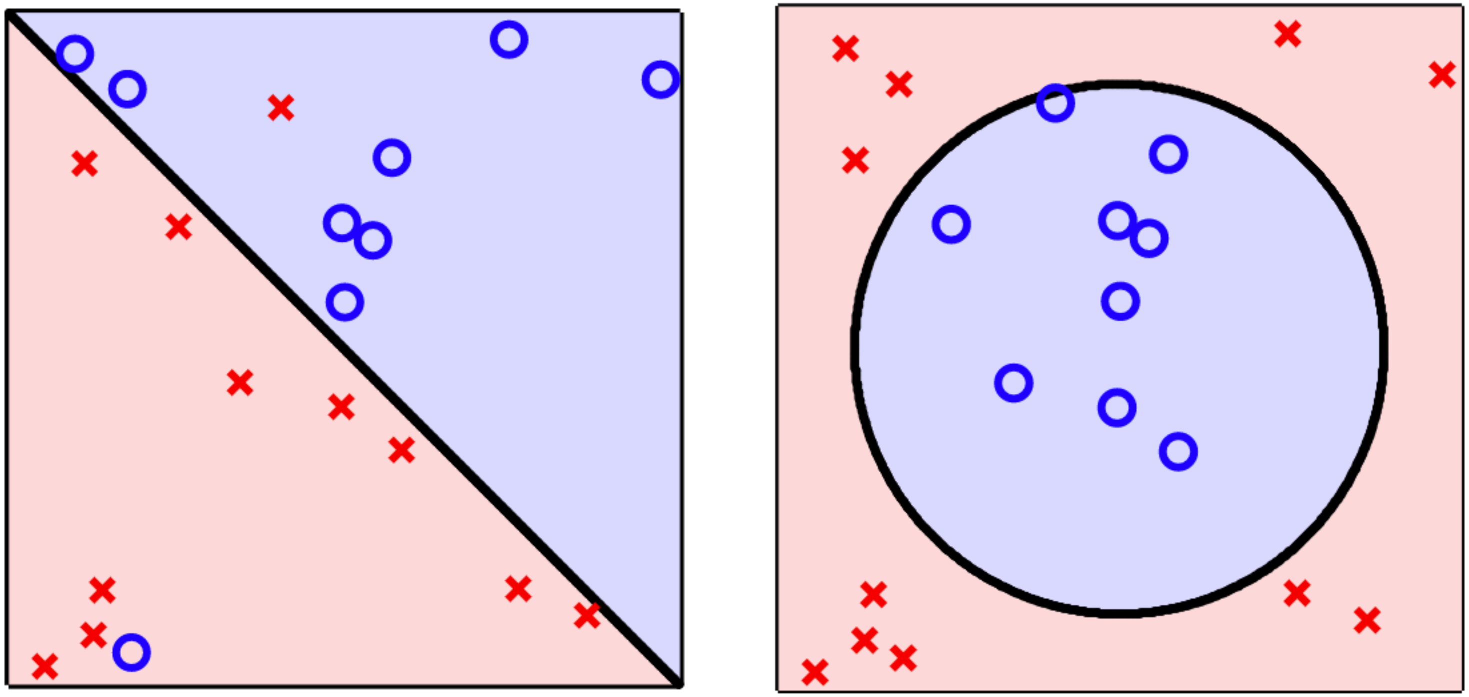
$$\mathbf{w} = \sum_{\mathbf{x}_n \text{ is SV}} \alpha_n y_n \mathbf{x}_n$$

Solve for b using any SV:

$$y_n\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b\right) = 1$$



Linearly non-separable cases?

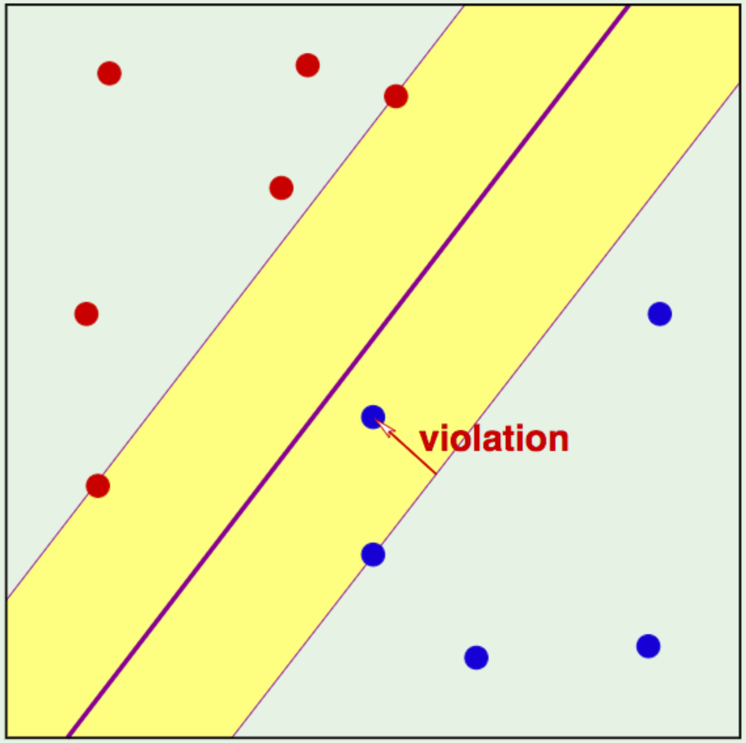


Case 1

Margin violation: $y_n\left(\mathbf{w}^\intercal\mathbf{x}_n + b\right) \ge 1$ fails

Quantify: $y_n\left(\mathbf{w}^\intercal\mathbf{x}_n + b\right) \ge 1 - \xi_n \qquad \xi_n \ge 0$

$$\text{Total violation } = \sum_{n=1}^{N} \xi_n$$



The new optimization

Minimize $\frac{1}{2}\,\mathbf{w}^{\mathsf{T}}\mathbf{w} + C\sum_{n=1}^{N}\xi_n$

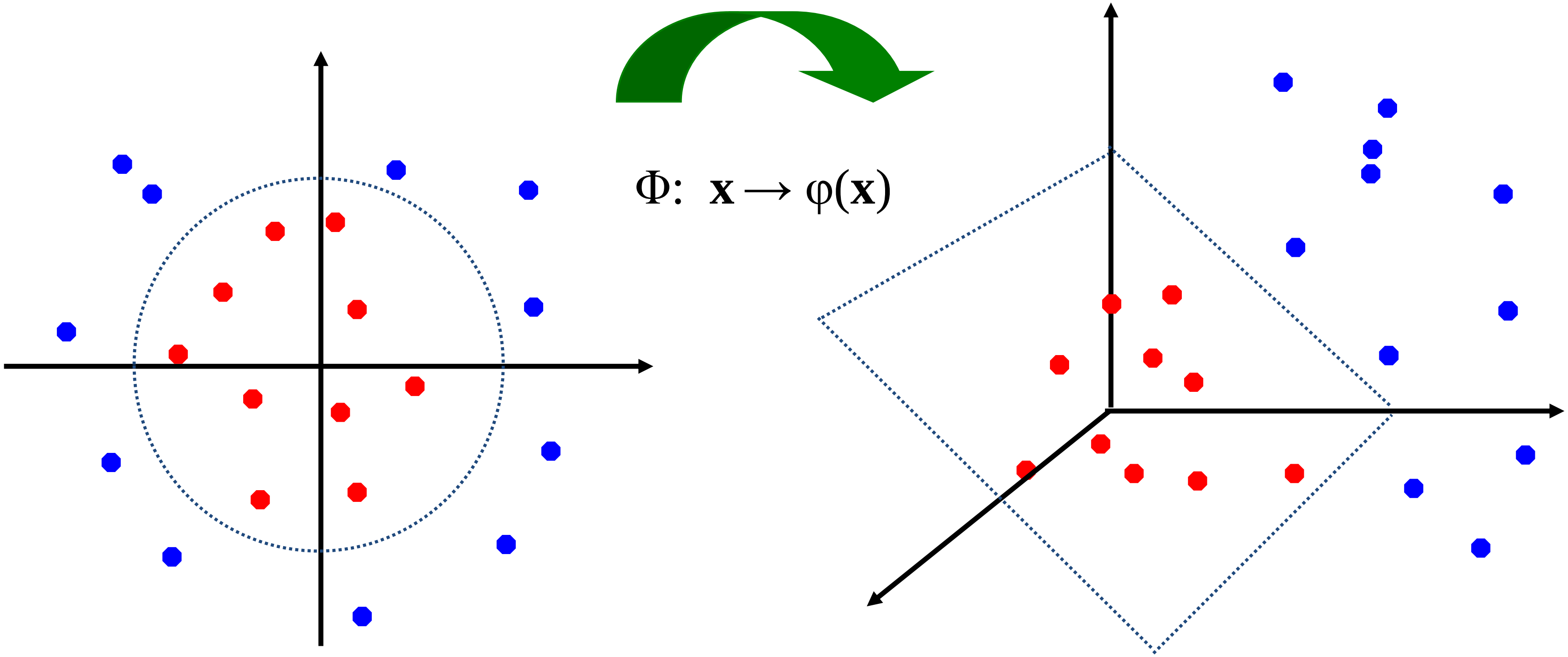
subject to $y_n\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b\right) \geq 1 - \xi_n$ for $n = 1, \ldots, N$

and $\xi_n \geq 0$ for $n = 1, \ldots, N$

$\mathbf{w} \in \mathbb{R}^d \;,\; b \in \mathbb{R} \;,\; \boldsymbol{\xi} \in \mathbb{R}^N$

Nonlinear SVMs

- General idea: the original input space can be mapped to some higher-dimensional feature space where the training set is separable:



$\Phi\colon\ \mathbf{x} \rightarrow \varphi(\mathbf{x})$

Kernel: $K(\mathbf{x}_i\,, \mathbf{x}_j) = \boldsymbol{\varphi}(\mathbf{x}_i\,) \cdot \boldsymbol{\varphi}(\mathbf{x}_j)$

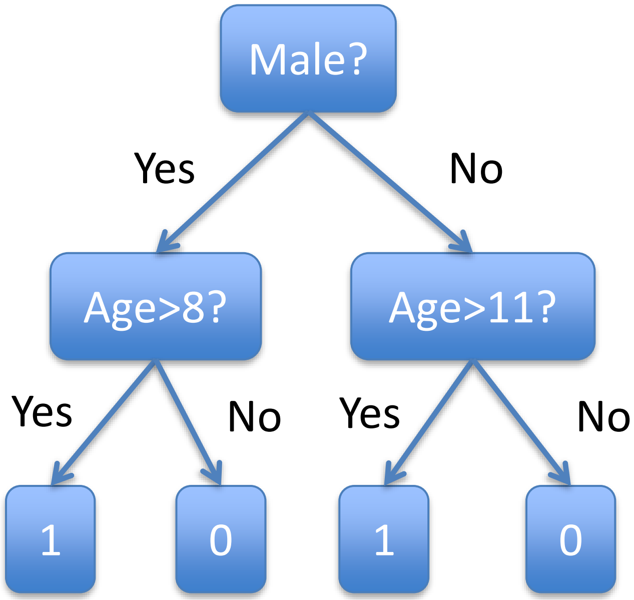
SVMs: Pros and cons

- Pros
  - Many publicly available SVM packages
  - Kernel-based framework is very powerful, flexible
  - SVMs work very well in practice, even with very small training sample sizes

- Cons
  - Computation, memory
    - During training time, must compute matrix of kernel values for every pair of examples
    - Learning can take a very long time for large-scale problems
  - Linear kernel SVMs are similar to linear perceptrons (just with added regularization) if trained with SGD

Decision Trees

(Binary) Decision Tree



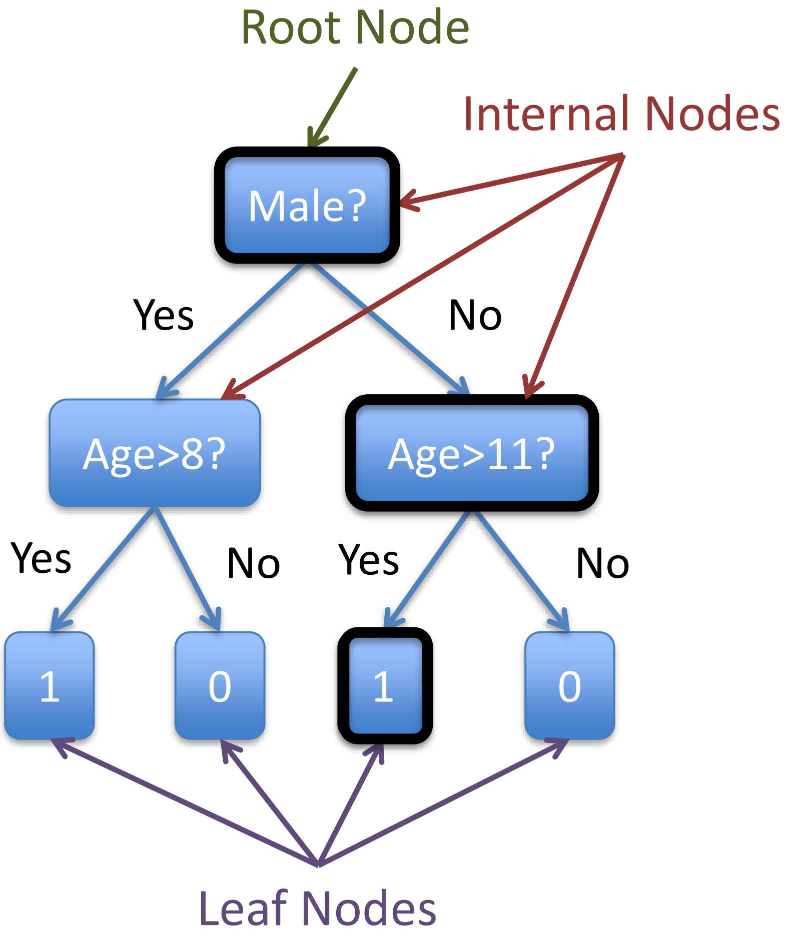
Don't overthink this, it is literally what it looks like.

Person	Age	Male?	Height > 55"
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	10	0	0

x

y

(Binary) Decision Tree



Input:

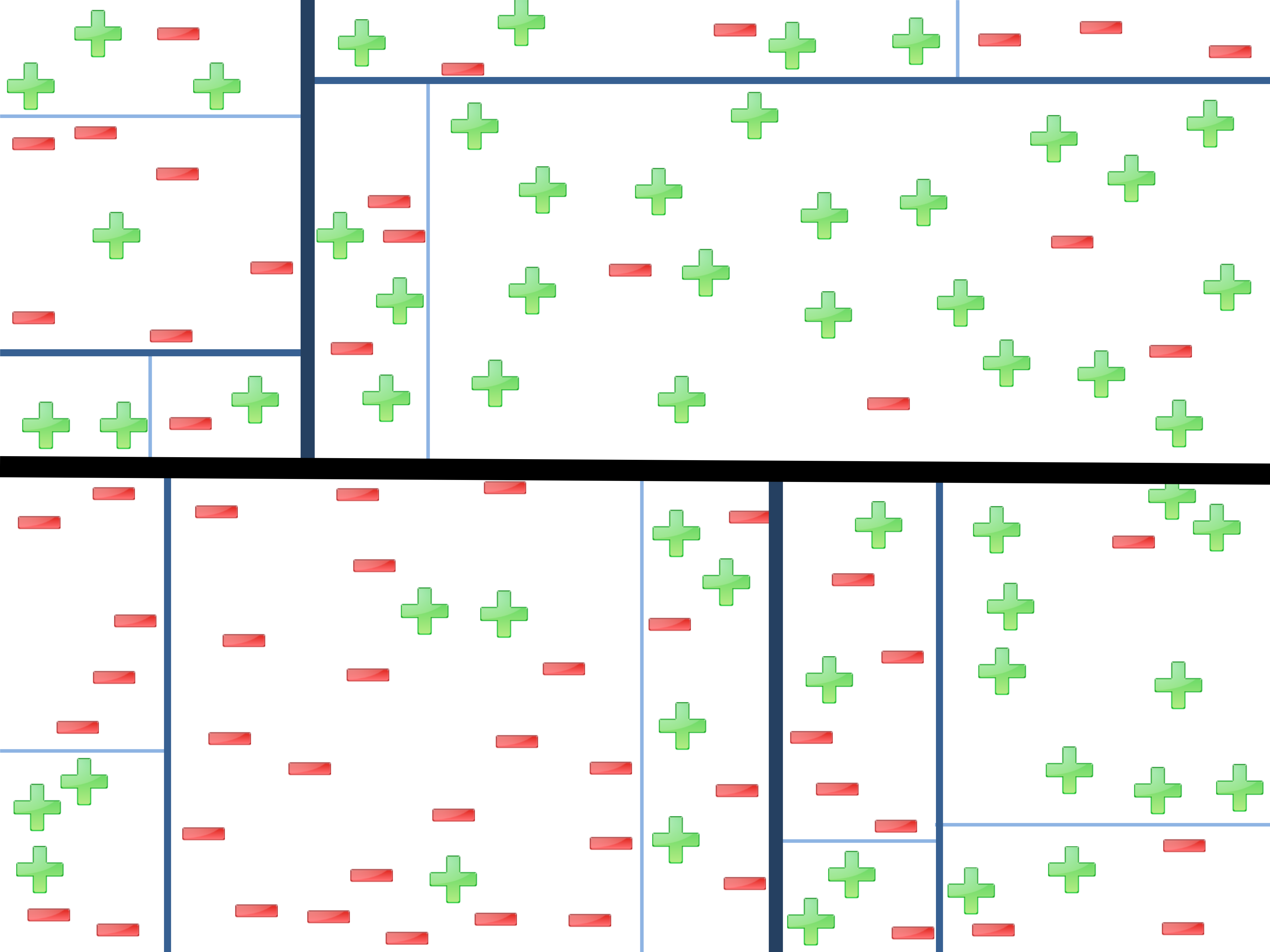
Alice
Gender: Female
Age: 14

Prediction: Height > 55”

Every **internal node** has a **binary** function q(x).

Every **leaf node** has a prediction, e.g., 0 or 1.

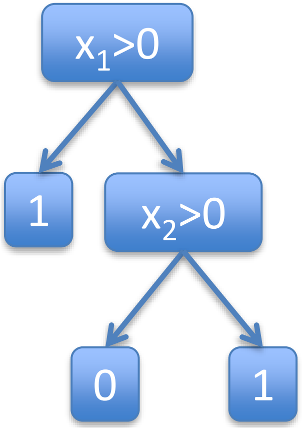
Prediction starts at **root node**.
Recursively calls query function.
Positive response → Left Child.
Negative response → Right Child.
Repeat until Leaf Node.



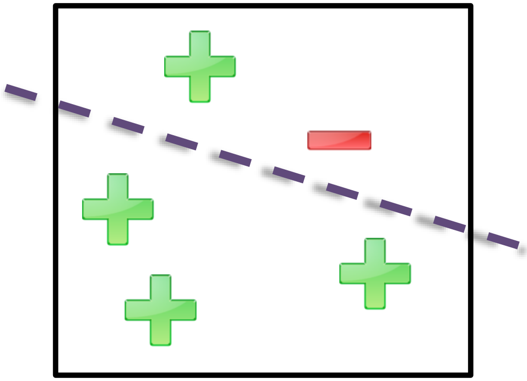
Decision Trees vs Linear Models

- Decision Trees are NON-LINEAR Models!

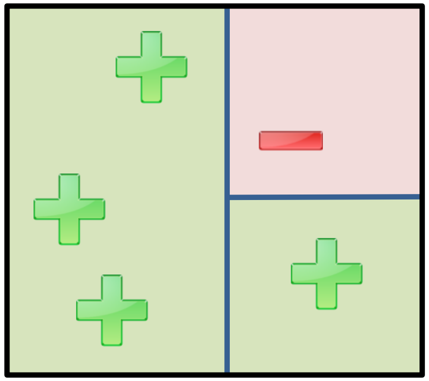
- Example:



No Linear Model
Can Achieve 0 Error



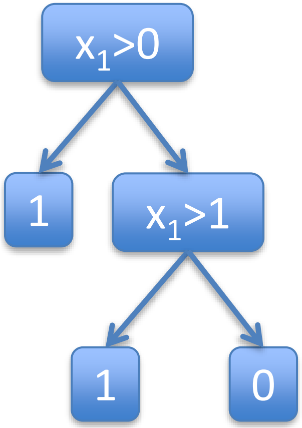
Simple Decision Tree
Can Achieve 0 Error



Decision Trees v. Linear Models

- Decision Trees are NON-LINEAR Models!

- Example:



$x_1>0$

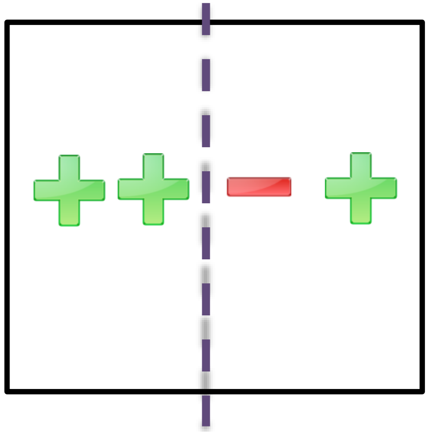
1

$x_1>1$

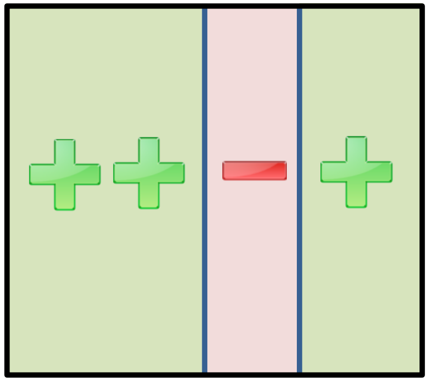
1

0

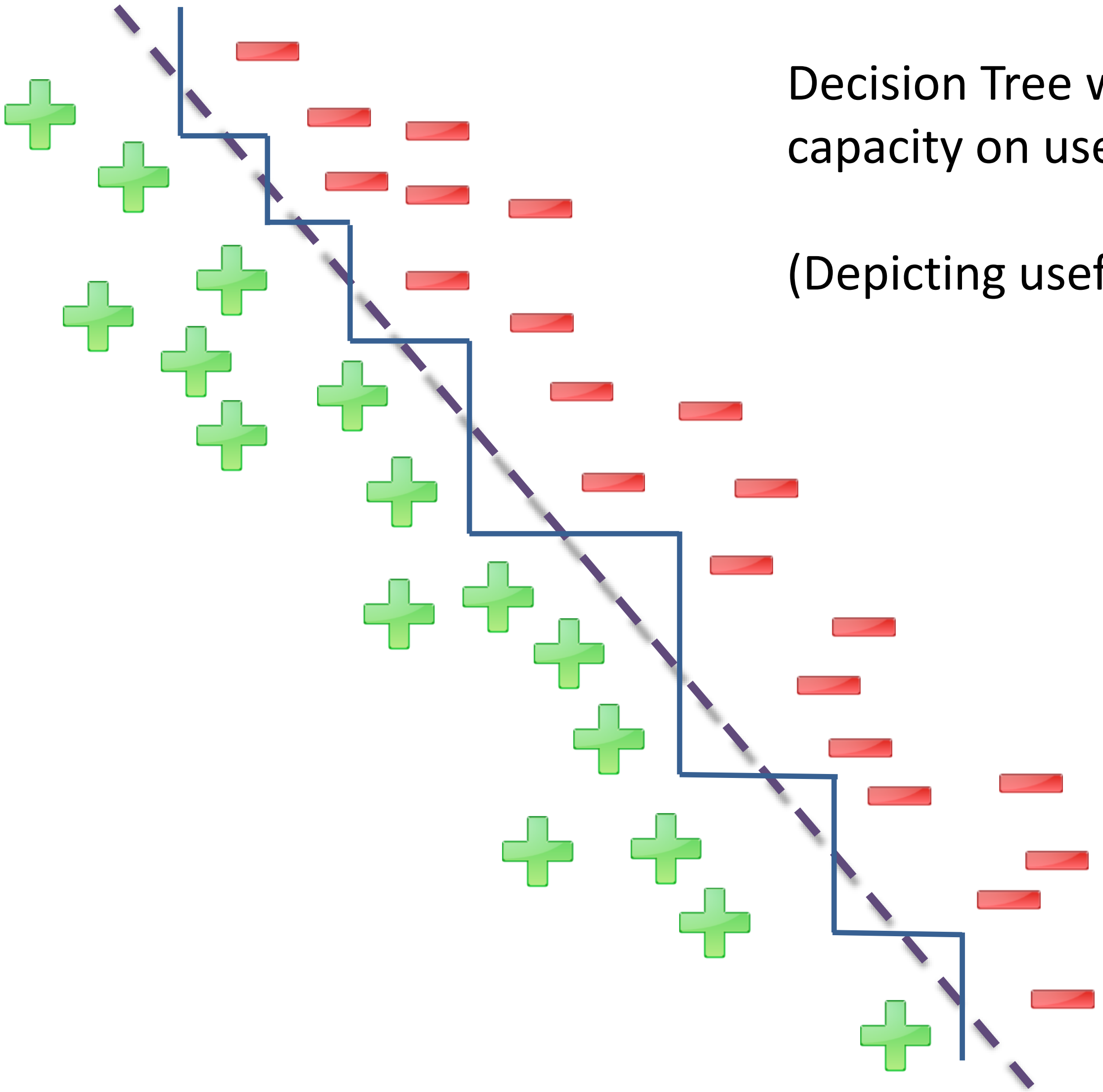
No Linear Model
Can Achieve 0 Error



Simple Decision Tree
Can Achieve 0 Error



More Extreme Example



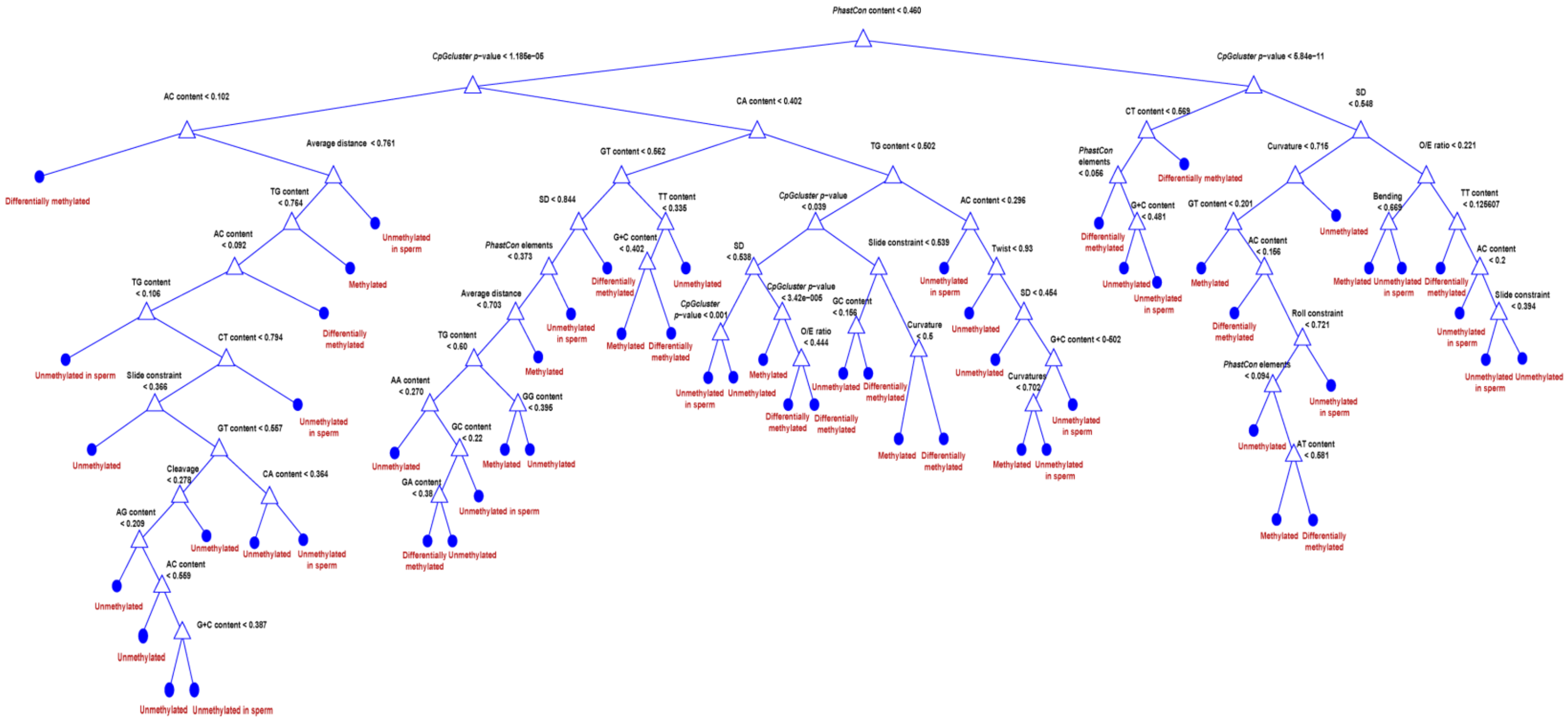
Decision Tree wastes most of model capacity on useless boundaries.

(Depicting useful boundaries)

Decision Trees v. Linear Models

- Decision Trees are often more accurate!
- Non-linearity is often more important
  - Just use many axis-aligned boundaries to approximate diagonal boundaries
- **Catch:** individual trees easily overfit
  - requires sufficient training data
  - Ensemble methods can fix this.

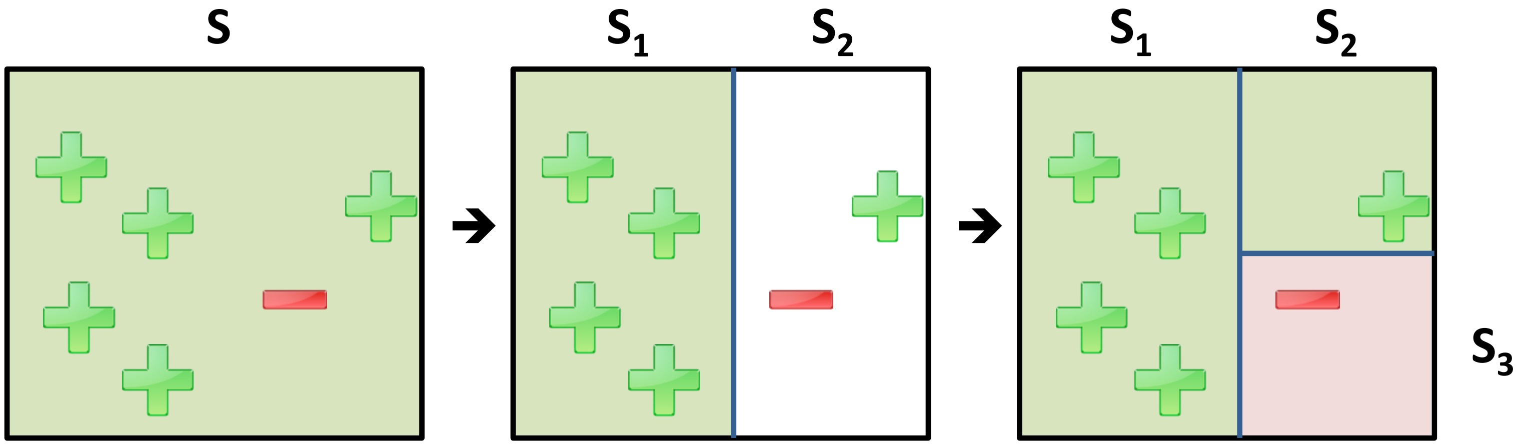
Decision Trees



Can get much larger!

Training Decision Trees (Top-Down)

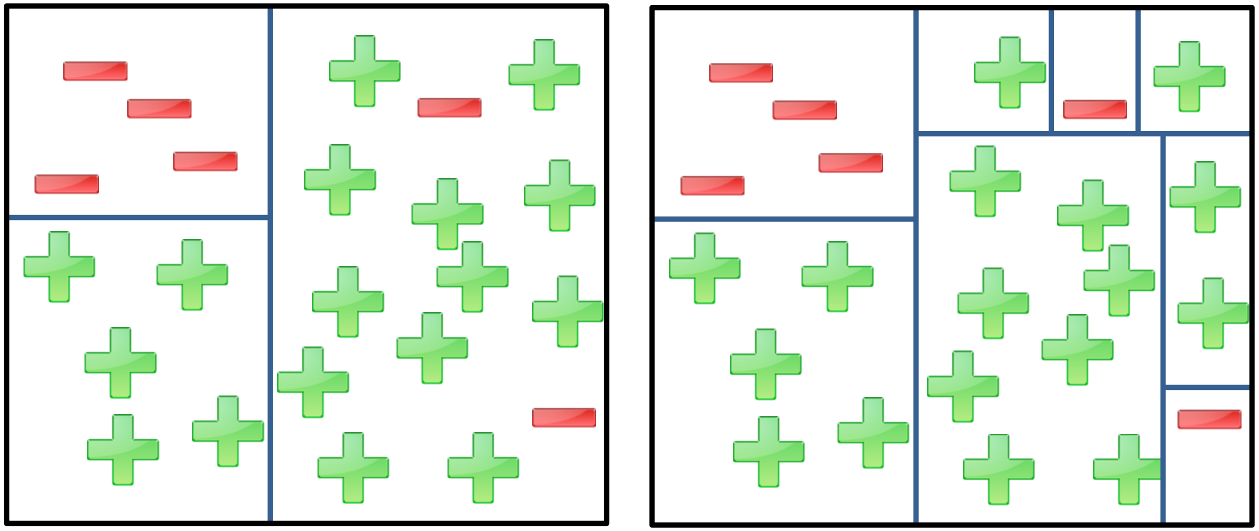
- Every intermediate step is a decision tree
  - You can stop any time and have a model
- Greedy algorithm
  - Doesn't backtrack
  - Cannot reconsider different higher-level splits.



When to Stop?

- In kept going, can learn tree with zero training error.
  - But such tree is probably overfitting to training set.
- How to stop training tree earlier?
  - I.e., how to regularize?

Which one has better test error?



Stopping Conditions (Regularizers)

- **Minimum Size:** do not split if resulting children are smaller than a minimum size.
  - **Most common stopping condition.**
- **Maximum Depth:** do not split if the resulting children are beyond some maximum depth of tree.
- **Maximum #Nodes:** do not split if tree already has maximum number of allowable nodes.
- **Minimum Reduction in Impurity:** do not split if resulting children do not reduce impurity by at least δ%.

…

Ensemble Methods

The idea

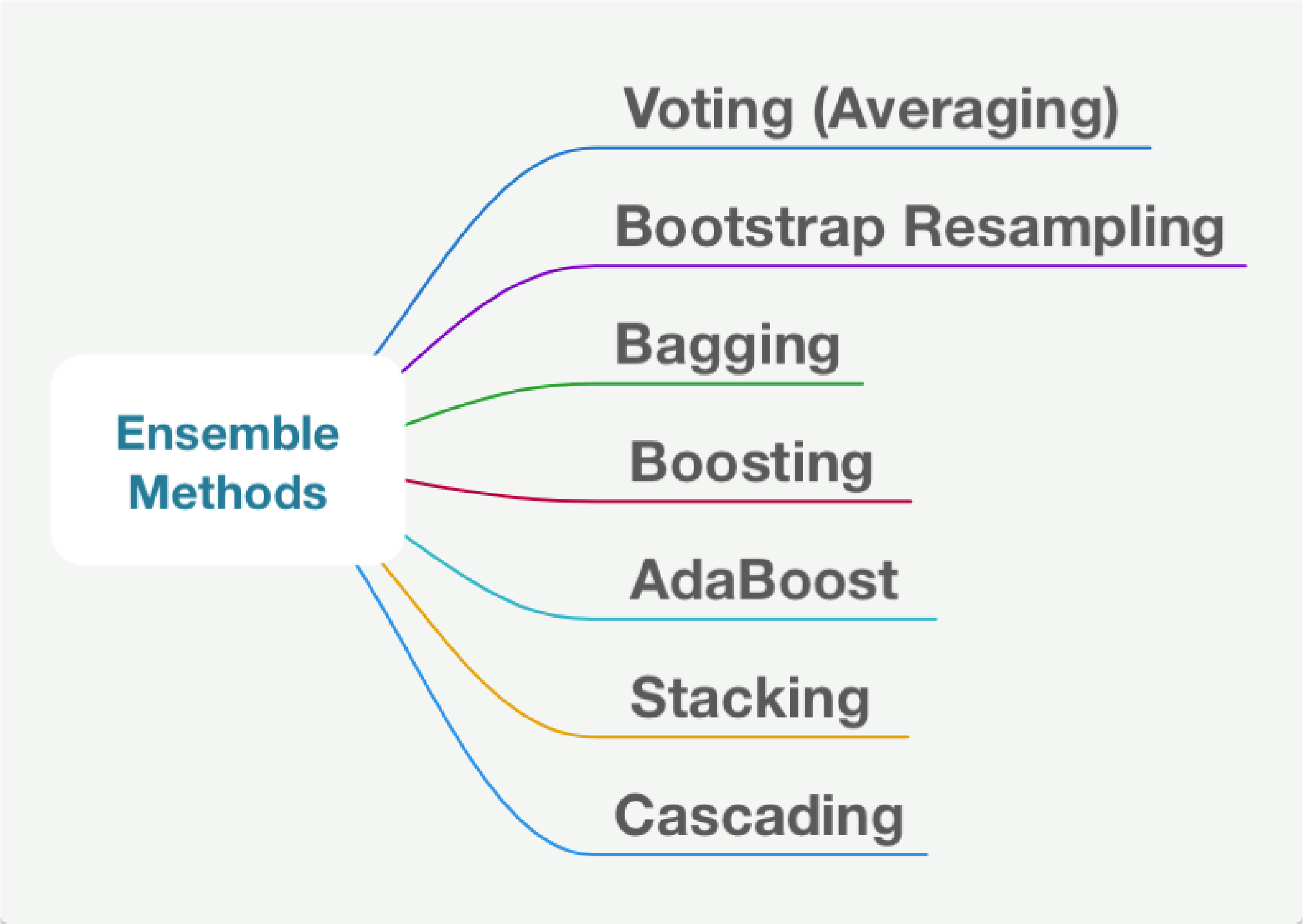
- It is often a good idea to combine several learning methods
- We want diverse classifiers, so their errors cancel out

- Base learner: Arbitrary learning algorithm which could be used on its own

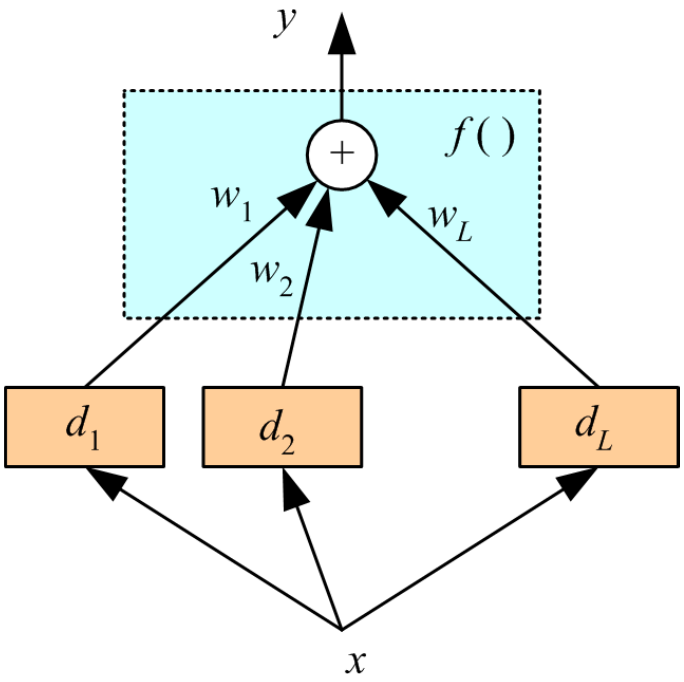
- Ensemble: A learning algorithm composed of a set of base learners.The

- base learners may be organized in some structure

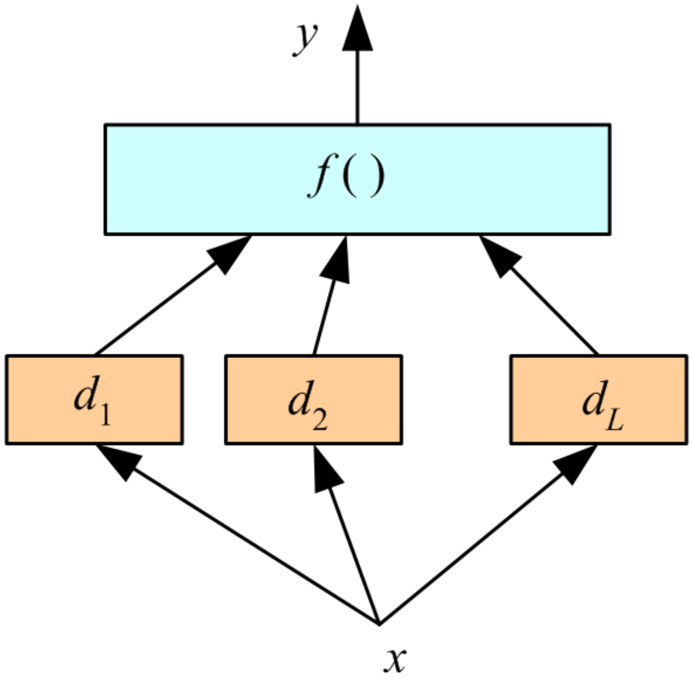
Constructing Ensembles



Averaging (Voting)



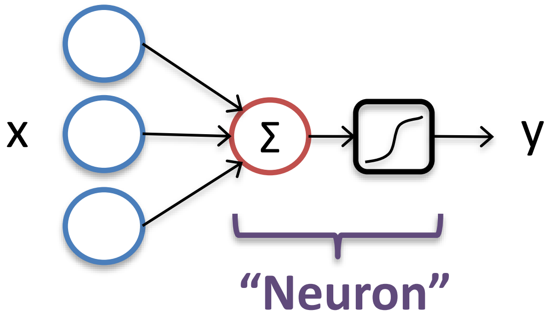
Stacking



Feed-Forward Neural Networks

1 Layer Neural Network

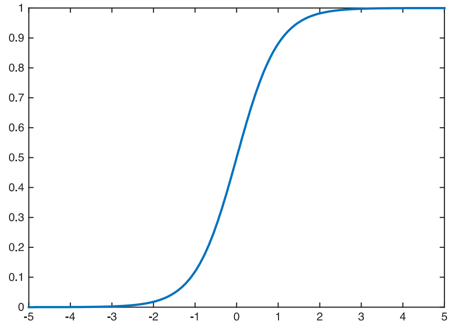
- 1 Neuron
  - Takes input x
  - Outputs y



$$f(x|w,b) = w^T x - b$$
$$= w_1 * x_1 + w_2 * x_2 + w_3 * x_3 - b$$

$$\longrightarrow y = \sigma(\ f(x)\ )$$

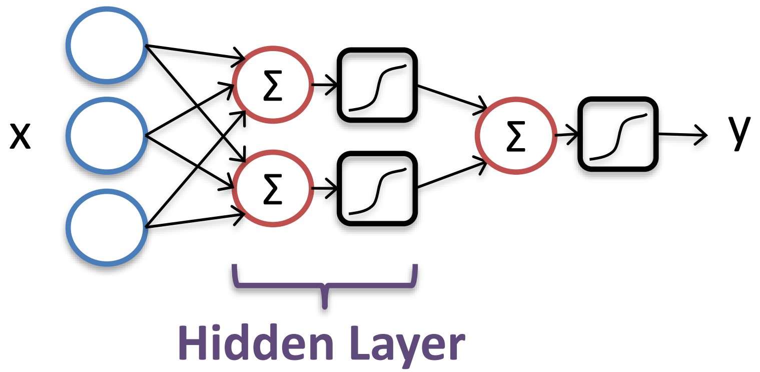
- **~Logistic Regression!**
  - Gradient Descent



sigmoid

tanh

2 Layer Neural Network

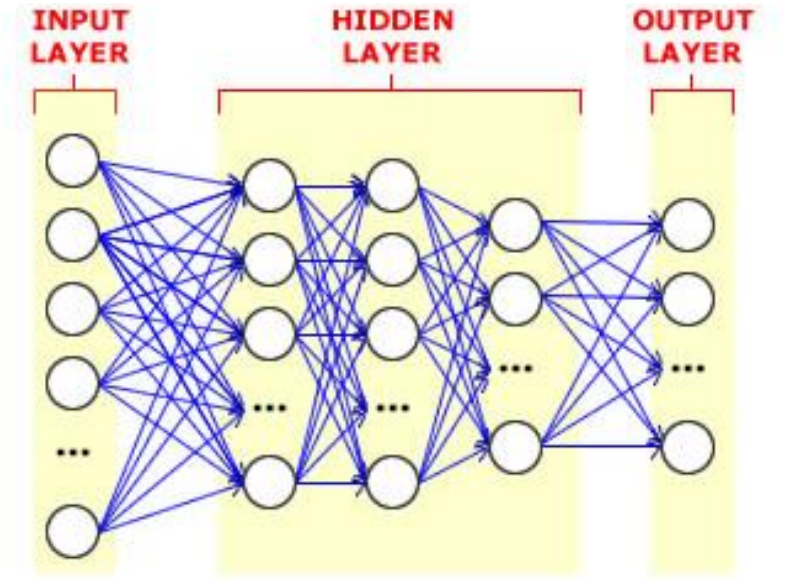


- 2 Layers of Neurons
  - 1st Layer takes input x
  - 2nd Layer takes output of 1st layer

Non-Linear!

- Can approximate arbitrary functions
  - Provided hidden layer is large enough
  - “fat” 2-Layer Network

Deep Neural Networks



Start here: *playground.tensorflow.org*

HW1

Overview

Intracranial EEG, multichannel

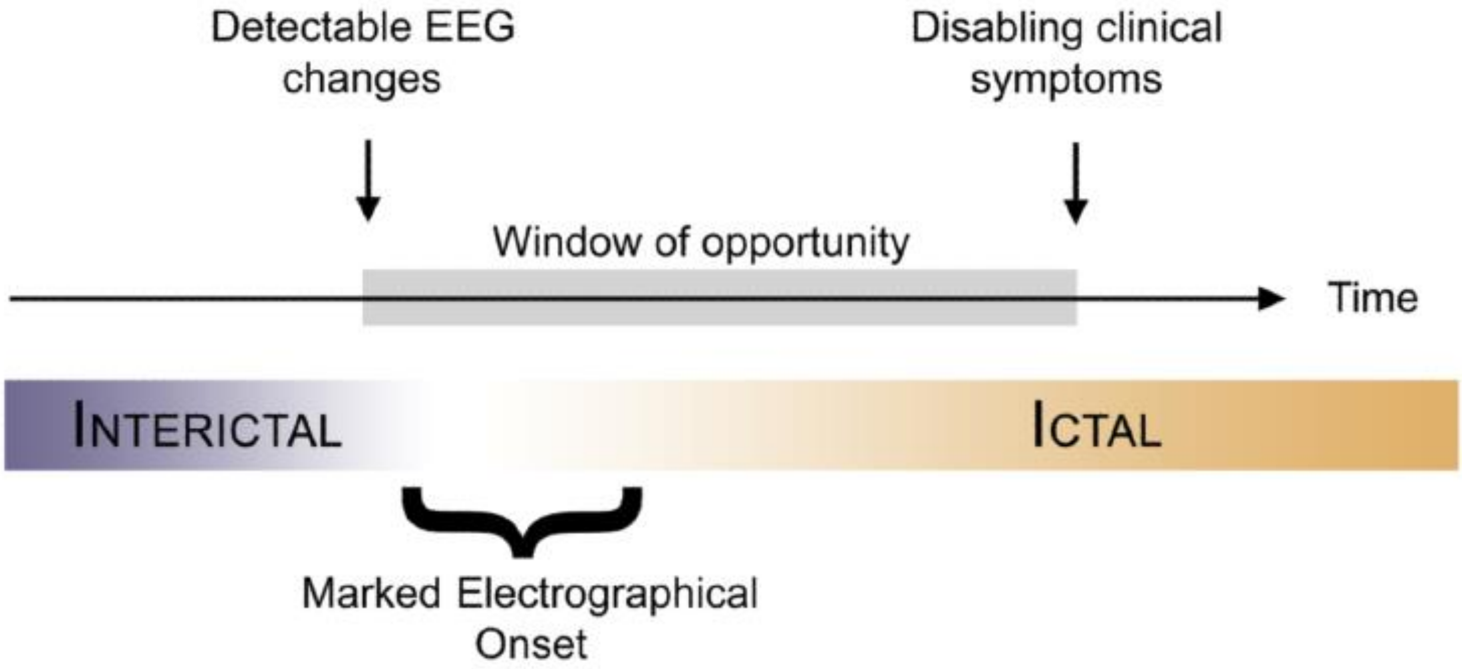
- varying numbers of electrodes
- sampled at 500 Hz or 5000 Hz

The temporal dynamics of brain activity can be classified into 4 states:

- Interictal (between seizures, or baseline)
- Preictal (prior to seizure)
- Ictal (seizure)
- Post-ictal (after seizures)

- The primary challenge in seizure forecasting is differentiating between the **preictal** and **interictal** states.

Seizure Detection Time Frame



- The time of the earliest detectable changes and the onset of disabling clinical symptoms: few seconds up to 30 seconds
- Closed-loop therapy must be delivered within that time frame to provide optimum benefit to the patient

Data preprocessing, feature extraction

- Different mathematical techniques can be applied to pre-process the data – Noisy data
- Magnitudes of different frequencies: a good source of features
- Frequency range chosen based on literature, and trial and error
- Time-domain features (biomarkers)
- Combinations of multiple features to be used in classification
- Features kept or discarded based on cross-validation performance
- The features from all channels concatenated, used for training

How to improve? try more complex features:

- Correlation coefficients
- …

Classification

Choose a model for classification:

- Each run gives a cross-validation score
- Find combinations of feature set and classifier giving higher scores
- scikit-learn python machine learning library
- Many different classifiers can be easily substituted in the code
- Many classifiers ranging from logistic regression to decision trees or support vector machines, …
- Optimize classifier parameters

Cross Validation

Cross-validation:

- Split the ictal training data based on whole seizures
    - For example for a ratio of 0.25 and 4 seizures, 1 entire seizure split out leaving the other 3 to train on
    - Or use k-fold cross-validation, takes much longer training time

Machine learning cycle:

- Train your model and check your cross-validation score

Ensemble!

Last but not least, ensemble:

- Individual models
- Other team member’s models