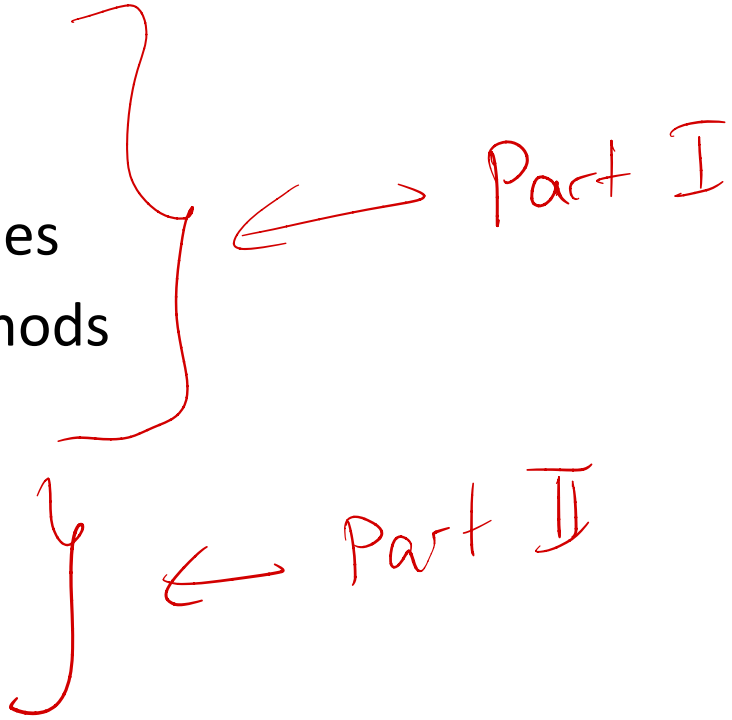
CS 4375
Midterm Review: Part I

Rishabh Iyer

University of Texas at Dallas

Topics for the Midterm Exam

- Linear Regression
- Perceptron
- Support Vector Machines
- Nearest Neighbor Methods
- Decision Trees
- Bayesian Methods
- Naïve Bayes
- Logistic Regression



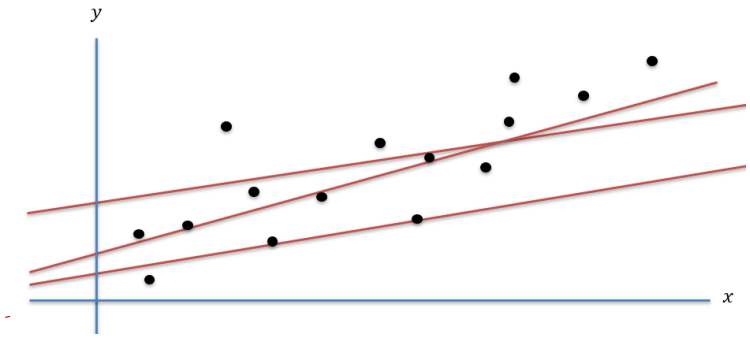
Topics for the Midterm Exam

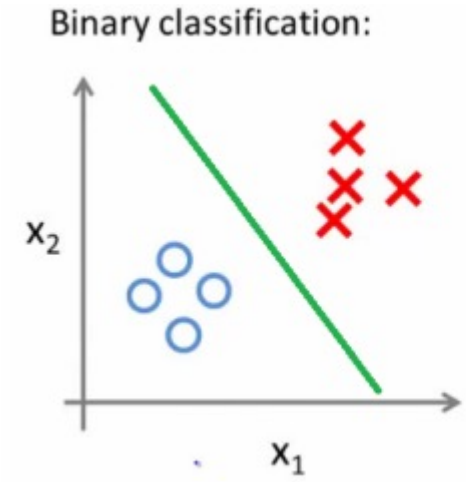
- **Linear Regression**
- Perceptron
- Support Vector Machines
- Nearest Neighbor Methods
- Decision Trees
- Bayesian Methods
- Naïve Bayes
- Logistic Regression

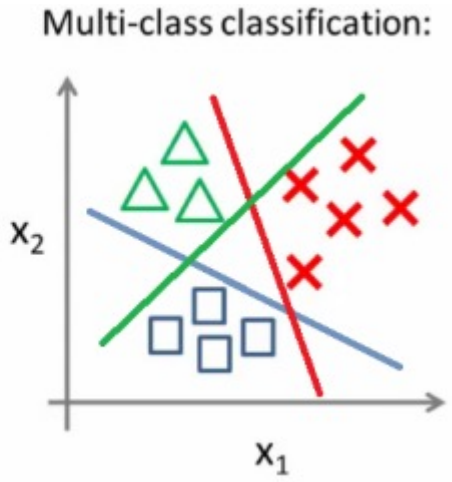
Classification vs Regression

Classification vs Regression

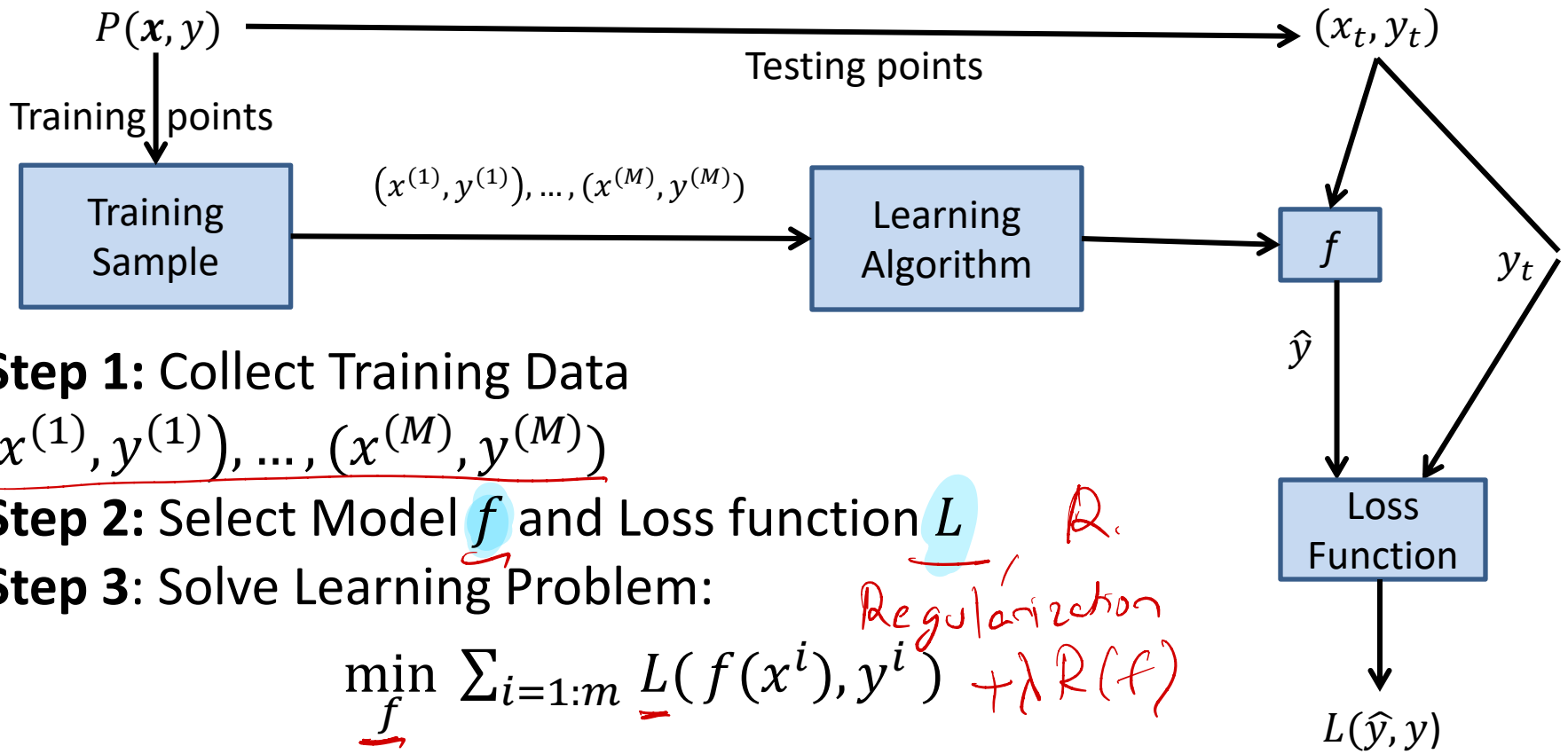
- Input: pairs of points $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\}$ with $x^{(m)} \in \mathbb{R}^n$ ← Feature vector
- Regression case: $y^{(m)} \in \mathbb{R}$ Labels
- Classification case: $y^{(m)} \in [0, k-1]$ [k-class classification]
- If k = 2, we get Binary classification







Recap: Supervised Learning Workflow

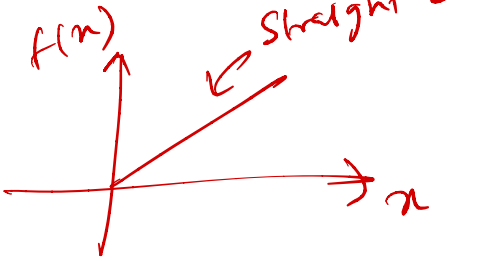


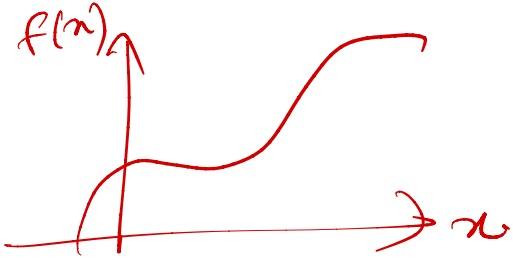
- **Step 1:** Collect Training Data
  $(x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})$
- **Step 2:** Select Model f and Loss function L
- **Step 3**: Solve Learning Problem:

$$\min_{f} \sum_{i=1:m} L(f(x^i), y^i)$$

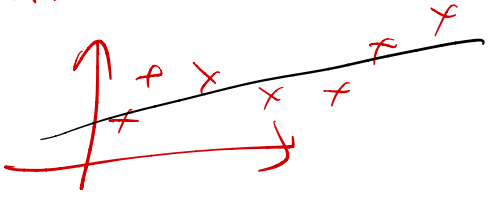
- **Step 4:** Obtain Predictions $\hat{y}_t = f(x_t)$ on all **Test Data**
- **Step 5:** Evaluation -- Measure the error $Err(\hat{y}_t, y_t)$

(1) Model f.





Goal: We want "f" to represent the data.



(2) Loss Fn L:

Goal: Fit "f" to the train data to best extent possible.

Linear Regression: Loss Function

- For any data point, x, the learning algorithm predicts $f(x)$
- In typical regression applications, measure the fit using a squared loss function

$$L(f) = \frac{1}{M}\sum_{m}\left(f\left(x^{(m)}\right) - y^{(m)}\right)^2$$

m ← Training Data.

- Want to minimize the average loss on the training data
- The optimal linear hypothesis is then given by

$$\min_{a,b}\frac{1}{M}\sum_{m}\left(ax^{(m)} + b - y^{(m)}\right)^2$$

$f(x) = \vec{a}x + b$

(1) Model $f(x) = \underline{a}^T x + \underline{b}$

(2) Loss Fn: $L(f) = (f(x) - y)^2$

$$\sum_m \left[f(x^{(m)}) - y^{(m)} \right]^2$$

Linear Regression Optimization: Gradient Descent

Iterative method to minimize a (convex) differentiable function L

- Pick an initial point w_0 ← Random
- Iterate until convergence

$$w_{t+1} = w_t - \gamma_t \nabla L(w_t)$$

  where γ_t is the t^{th} step size (sometimes called learning rate)
- w in this case is (a, b)

$$a_{t+1} = a_t - \gamma_t \nabla L(a_t)$$
$$b_{t+1} = b_t - \gamma_t \nabla L(b_t)$$

Gradients for Linear Regression

- The Loss Function for Linear Regression is:

$$L(a,b) = \frac{1}{M}\sum_{m}\left(a^{T}x^{(m)} + b - y^{(m)}\right)^2$$

- The gradients with respect to a and b are:

$$\nabla L_a(a,b) = \frac{1}{M}\sum_m 2(a^{T}x^{(m)} + b - y^{m})\, x^{(m)}$$

$$\nabla L_b(a,b) = \frac{1}{M}\sum_m 2(a^{T}x^{(m)} + b - y^{m})$$

- The gradients can be obtained by using the chain rule

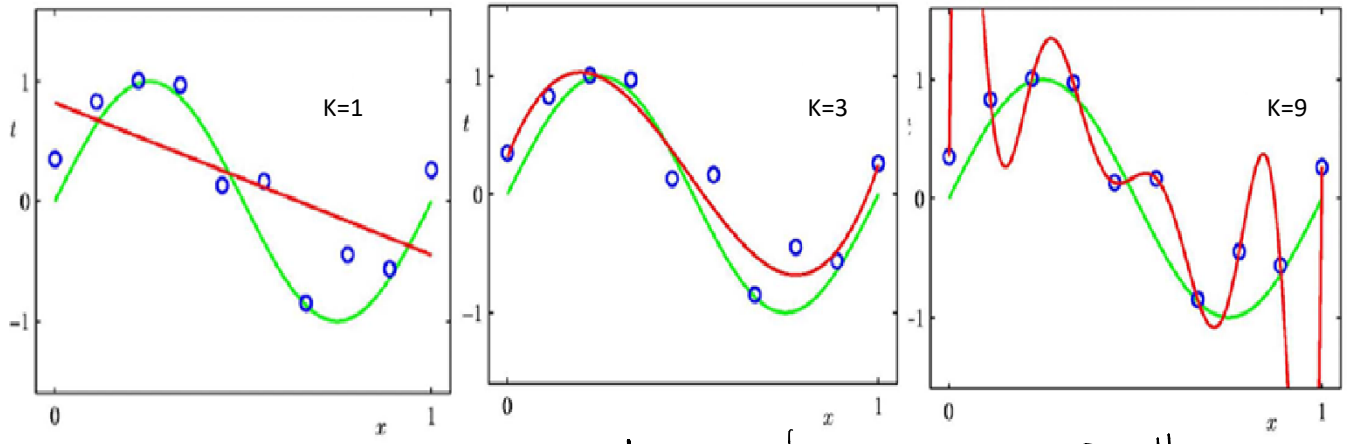
$x \in \mathbb{R}^n$, $\nabla_a L \in \mathbb{R}^n$, $\nabla_b L \in \mathbb{R}$

$a \in \mathbb{R}^n$

$b \in \mathbb{R}$

Polynomial Regression

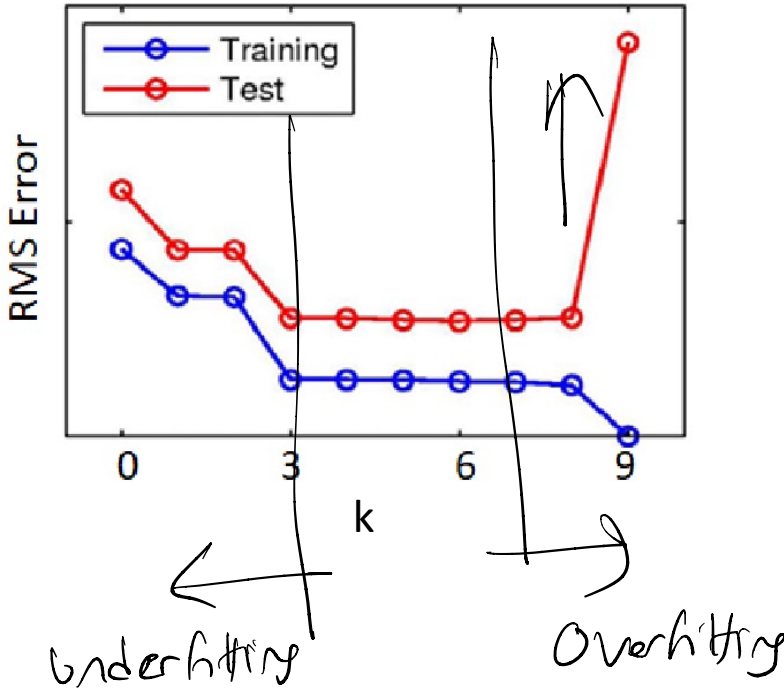
- What if we enlarge the hypothesis class?
  - Quadratic functions: $ax^2 + bx + c = f(n)$
  - k-degree polynomials: $a_k x^k + a_{k-1} x^{k-1} + \cdots + a_1 x + a_0 = f(n)$
- Can we always learn "better" with a larger hypothesis class?



Better Fit on Train Data ≠ Learn Better

Polynomial Regression: Overfitting

- As *the degree of the polynomial (k)* increases, training error decreases monotonically
- As *k* increases test error can increase
- Test error can decrease at first, but increases
- Overfitting can occur
  - When the model is too complex and trivially fits the data (i.e., too many parameters)
  - When the data is not enough to estimate the parameters
  - Model captures the noise (or the chance)



Use $k = 3$

Topics for the Midterm Exam

- Linear Regression
- **Perceptron**
- Support Vector Machines
- Nearest Neighbor Methods
- Decision Trees
- Bayesian Methods
- Naïve Bayes
- Logistic Regression

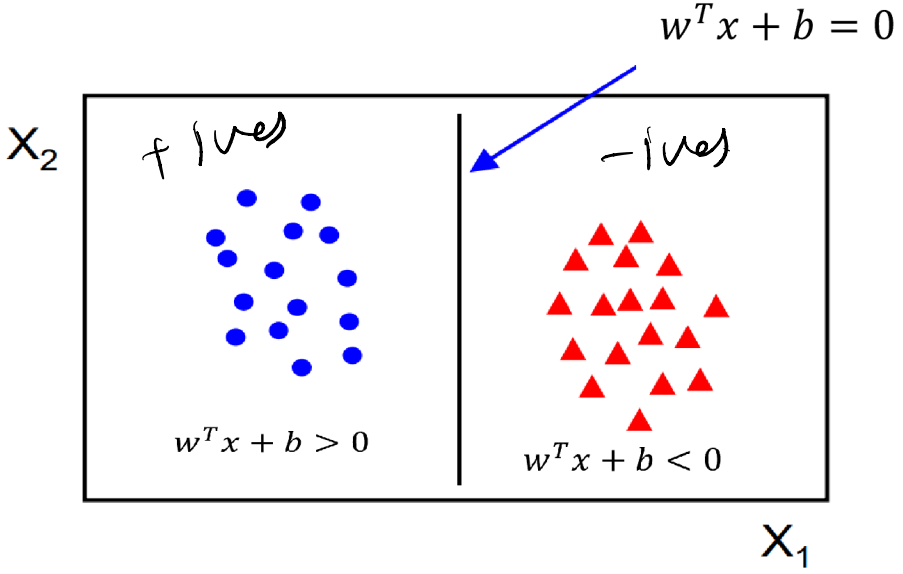
Linear Separators and Hyperplanes

- In n dimensions, a hyperplane is a solution to the equation

$$w^T x + b = 0$$

with $w \in \mathbb{R}^n, b \in \mathbb{R}$

- Hyperplanes divide $\mathbb{R}^n$ into two distinct sets of points (called open halfspaces)
  - Half Space 1: $w^T x + b > 0$
  - Half Space 2: $w^T x + b < 0$



The 0/1 Loss (Seperable Case)

- Input $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(M)}, y^{(M)}\right)$ with $x^{(m)} \in \mathbb{R}^n$ and $y^{(m)} \in \{-1, +1\}$
- Hypothesis space: separating hyperplanes

$$f(x) = sign\left(w^T x + b\right)$$

- How should we choose the loss function?
  - Count the number of misclassifications

$$zero/one\ loss = \frac{1}{2}\sum_m \left|y^{(m)} - sign(w^T x^{(m)} + b)\right|$$

  - Tough to optimize, gradient contains no information

The Perceptron Loss (Seperable Case)

- Input $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(M)}, y^{(M)}\right)$ with $x^{(m)} \in \mathbb{R}^n$ and $y^{(m)} \in \{-1, +1\}$
- Hypothesis space: separating hyperplanes

$$f(x) = sign\,(w^T x + b)$$

- How should we choose the loss function?
  - Penalize misclassification linearly by the size of the violation

$$perceptron\ loss = \sum_m \max\{0, -y^{(m)}(w^T x^{(m)} + b)\}$$

  - Modified hinge loss (this loss is convex, but not differentiable)

↳ Cont, Non Diff.

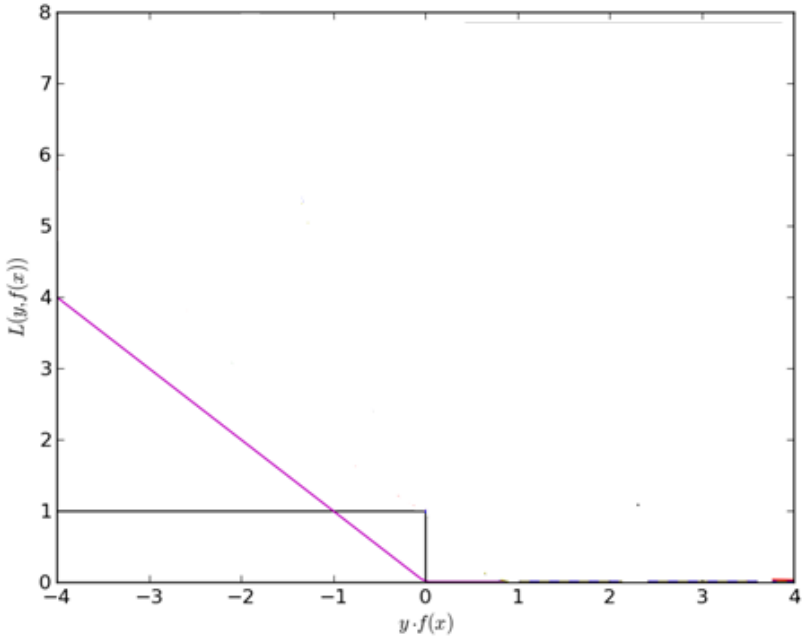
0/1 Loss Vs Perceptron Loss

- Zero/One Loss which counts the number of mis-classifications:

$$zero/one\ loss = \frac{1}{2}\sum_{m}\left|y^{(m)} - sign(w^T x^{(m)} + b)\right|$$

- Perceptron Loss:

$$perceptron\ loss = \sum_{m}\max\{0, -y^{(m)}(w^T x^{(m)} + b)\}$$



The Perceptron Algorithm

- Try to minimize the perceptron loss using (sub)gradient descent

$$\nabla_w(perceptron\ loss) = -\sum_{m=1}^{M}\left(y^{(m)}x^{(m)} \cdot 1_{-y^{(m)}f_{w,b}(x^{(m)})\geq 0}\right)$$

$$\nabla_b(perceptron\ loss) = -\sum_{m=1}^{M}\left(y^{(m)} \cdot 1_{-y^{(m)}f_{w,b}(x^{(m)})\geq 0}\right)$$

Is equal to zero if the m^{th} data point is correctly classified and one otherwise

The Perceptron Algorithm

- Try to minimize the perceptron loss using (sub)gradient descent

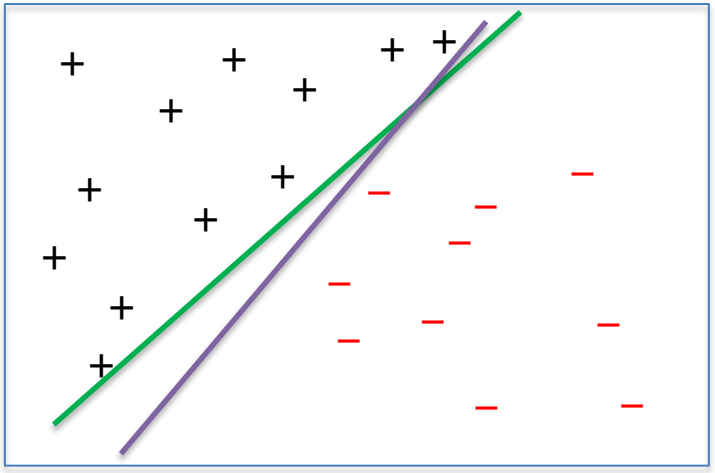
$$w^{(t+1)} = w^{(t)} + \gamma_t \sum_{m=1}^{M} \left(y^{(m)} x^{(m)} \cdot 1_{-y^{(m)} f_{w,b}(x^{(m)}) \geq 0} \right)$$

$$b^{(t+1)} = b^{(t)} + \gamma_t \sum_{m=1}^{M} \left(y^{(m)} \cdot 1_{-y^{(m)} f_{w,b}(x^{(m)}) \geq 0} \right)$$

- With step size γ_t (also called the learning rate)
- Note that, for convergence of subgradient methods, a diminishing step size, e.g., $\gamma_t = \frac{1}{1+t}$ is required

Perceptron Drawbacks

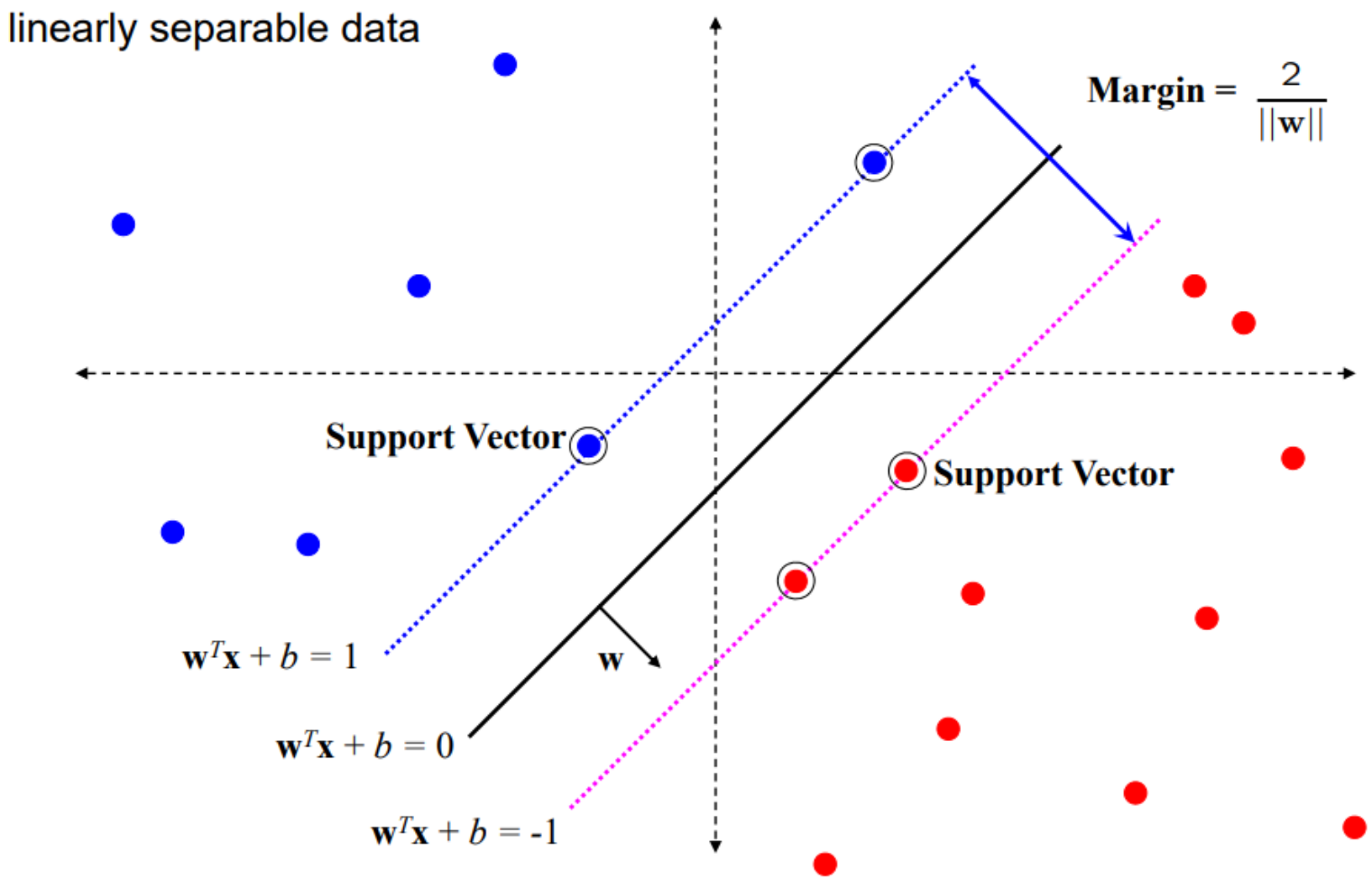
- No convergence guarantees if the observations are not linearly separable
- Can overfit
  - There can be a number of perfect classifiers, but the perceptron algorithm doesn't have any mechanism for choosing between them



Topics for the Midterm Exam

- Linear Regression
- Perceptron
- **Support Vector Machines**
- Nearest Neighbor Methods
- Decision Trees
- Bayesian Methods
- Naïve Bayes
- Logistic Regression

Support Vector Machines



Support Vector Machines Formulation

- This analysis yields the following optimization problem

$$\max_{w,b} \frac{1}{\|w\|}$$

such that

$$y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1, \text{for all } i$$

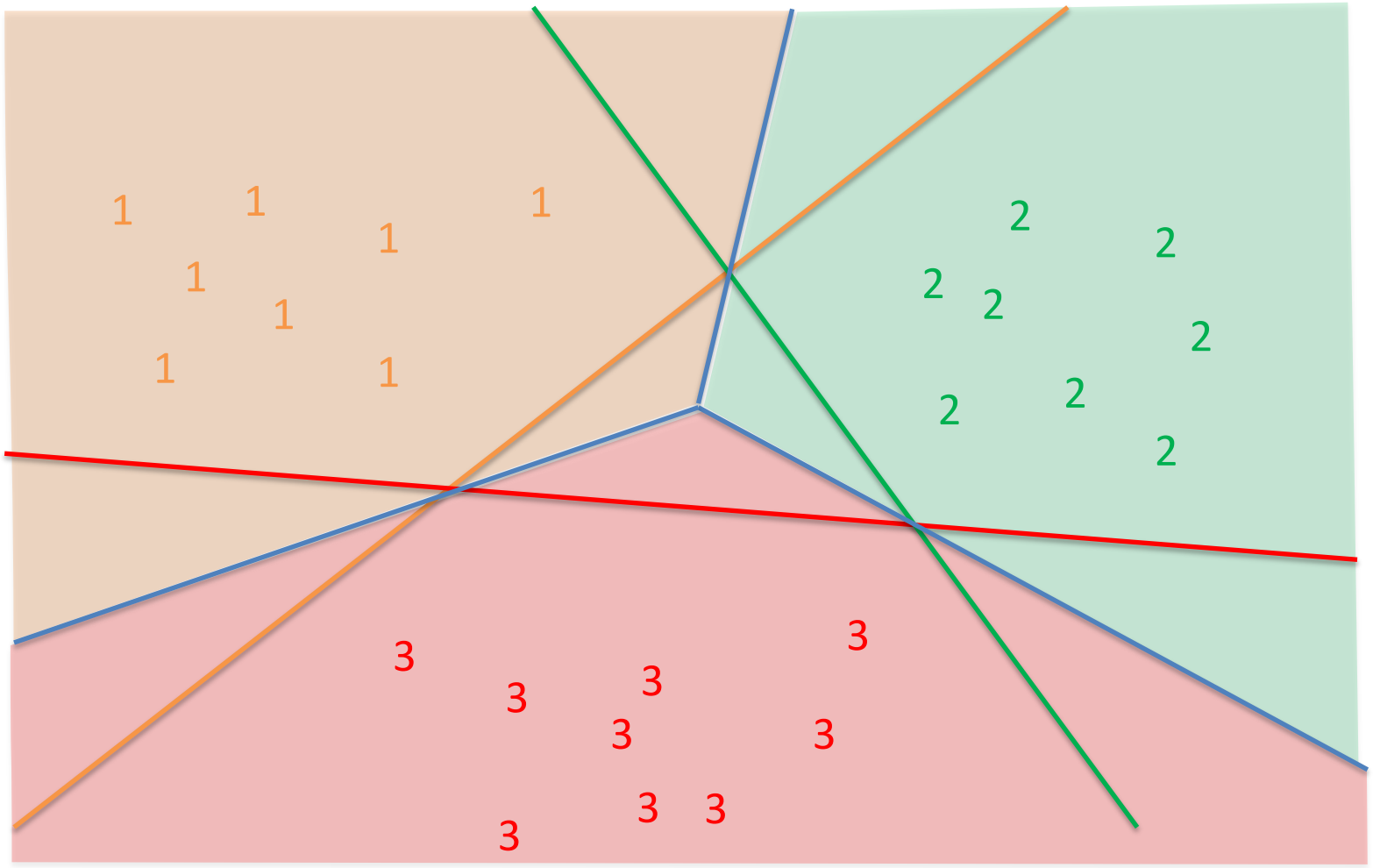
- Or, equivalently,

$$\min_{w,b} \|w\|^2$$

such that

$$y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1, \text{for all } i$$

One-Versus-All SVMs



Regions in which points are classified by highest value of $w^T x + b$

SVM and Slack Variables

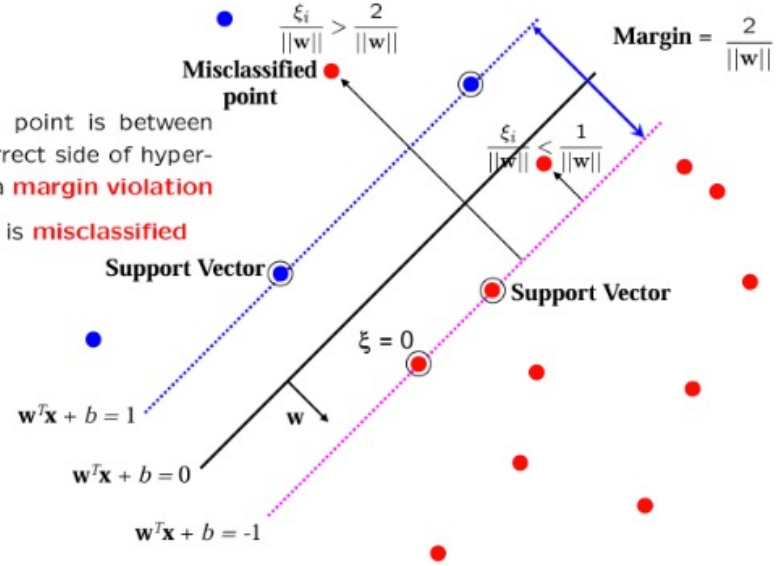
$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{M}\xi_i$$

subject to $y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i.

$\xi_i \geq 0$

for $0 < \xi \leq 1$ point is between margin and correct side of hyper-plane. This is a **margin violation**

for $\xi > 1$ point is **misclassified**



Hinge Loss Formulation

- Obtain a new objective by substituting in for ξ

$$\min_{w,b} \frac{1}{2}\|w\|^2 + c\sum_i \max\{0, 1 - y_i(w^T x^{(i)} + b)\}$$

Penalty to prevent overfitting

Hinge loss

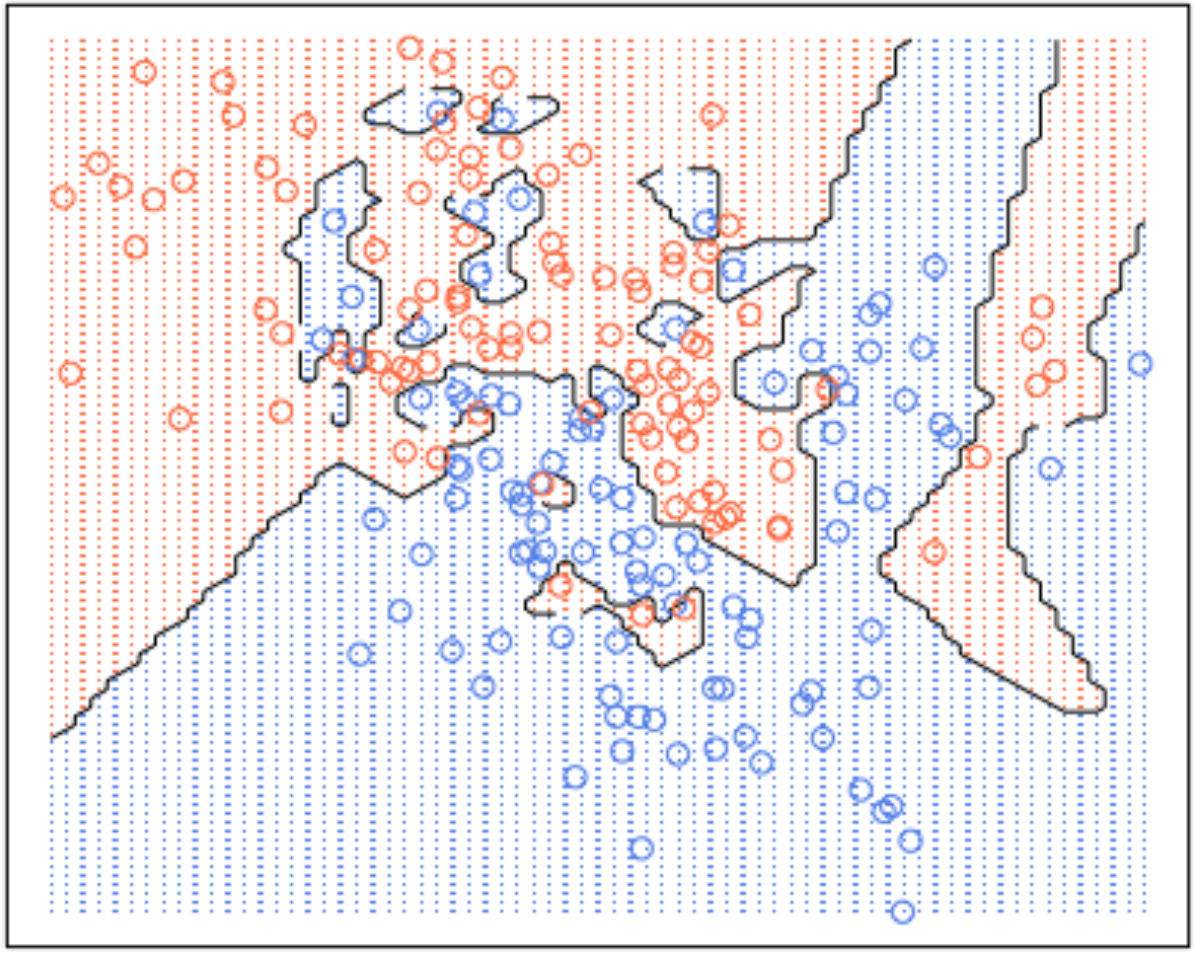
Topics for the Midterm Exam

- Linear Regression
- Perceptron
- Support Vector Machines
- **Nearest Neighbor Methods**
- Decision Trees
- Bayesian Methods
- Naïve Bayes
- Logistic Regression

Nearest Neighbor Methods

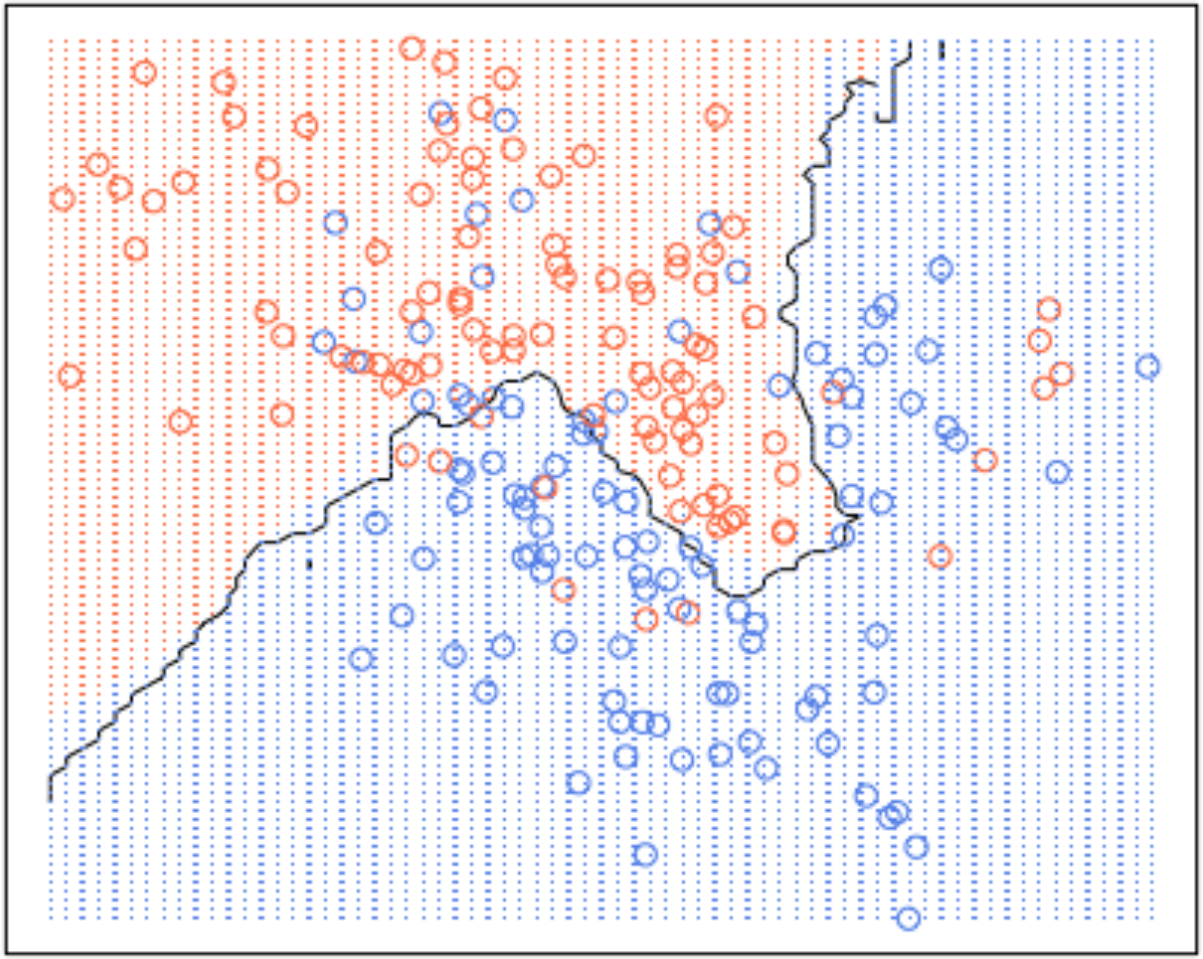
- Learning
  - Store all training examples
- Classifying a new point x'
  - Find the training example $(x^{(i)}, y^{(i)})$ such that $x^{(i)}$is closest (for some notion of close) to x'
  - Classify x' with the label $y^{(i)}$

1-NN Example



[Kevin Zakka]

20-NN Example



[Kevin Zakka]

Nearest Neighbor Methods

- Applies to data sets with points in $\mathbb{R}^d$
  - Best for large data sets with only a few (< 20) attributes
- Advantages
  - Learning is easy
  - Can learn complicated decision boundaries
- Disadvantages
  - Classification is slow (need to keep the entire training set around)
  - Easily fooled by irrelevant attributes

Secret to Getting Nearest Neighbors to Work

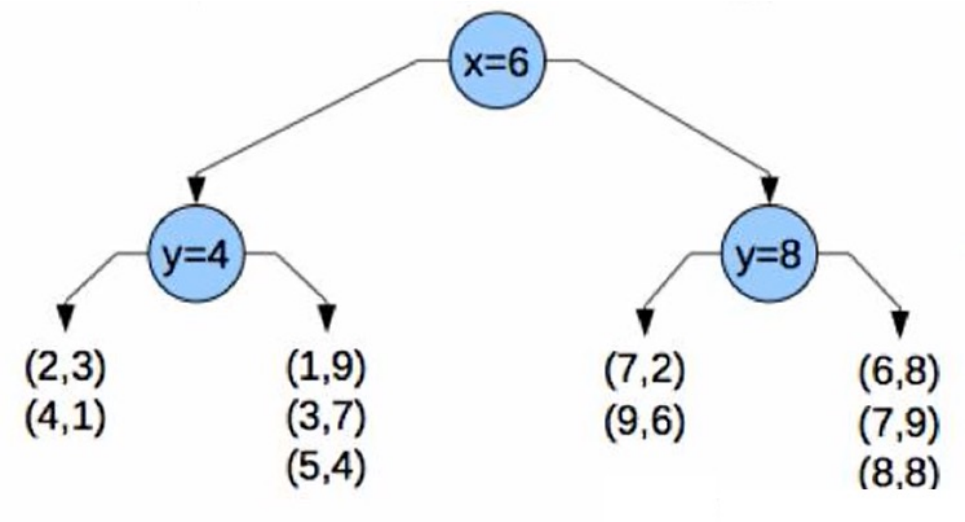
- Make sure there are no Irrelevant Attributes
  - Clean up features and remove features with low correlation with labels
- Make sure all features are normalized!
  - Feature normalization is key in KNN
- Make sure to tune k
- For large datasets use approximate nearest neighbor methods like kD trees

K-Dimensional Trees

- k-d trees provide a data structure that can help simplify the classification task by constructing a tree that partitions the search space
  - Starting with the entire training set, choose some dimension, i
  - Select an element of the training data whose i^{th} dimension has the median value among all elements of the training set
  - Divide the training set into two pieces: depending on whether their i^{th} attribute is smaller or larger than the median
  - Repeat this partitioning process on each of the two new pieces separately

K-Dimensional Trees

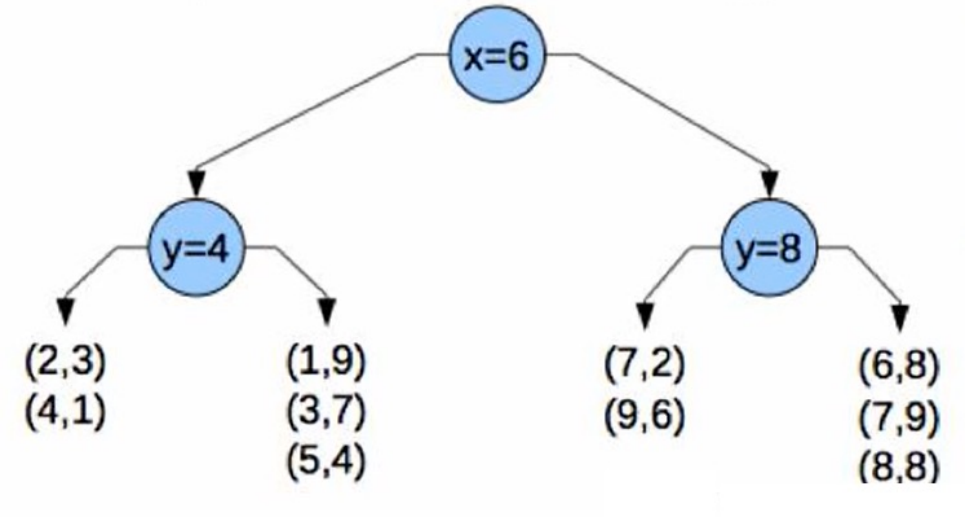
- Building a K-D tree from training data:
  - {(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)}
  - pick random dimension, find median, split data, repeat

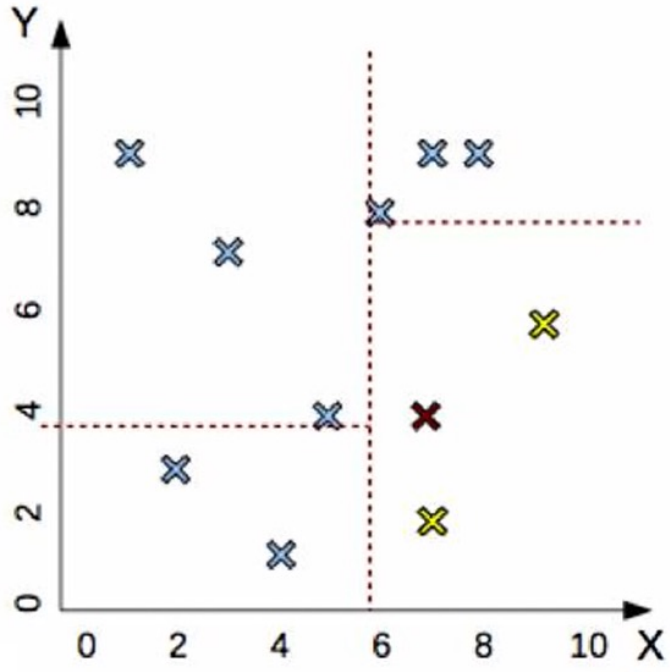


Adapted from Victor Lavrenko

K-Dimensional Trees

- Building a K-D tree from training data:
  - {(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)}
  - pick random dimension, find median, split data, repeat

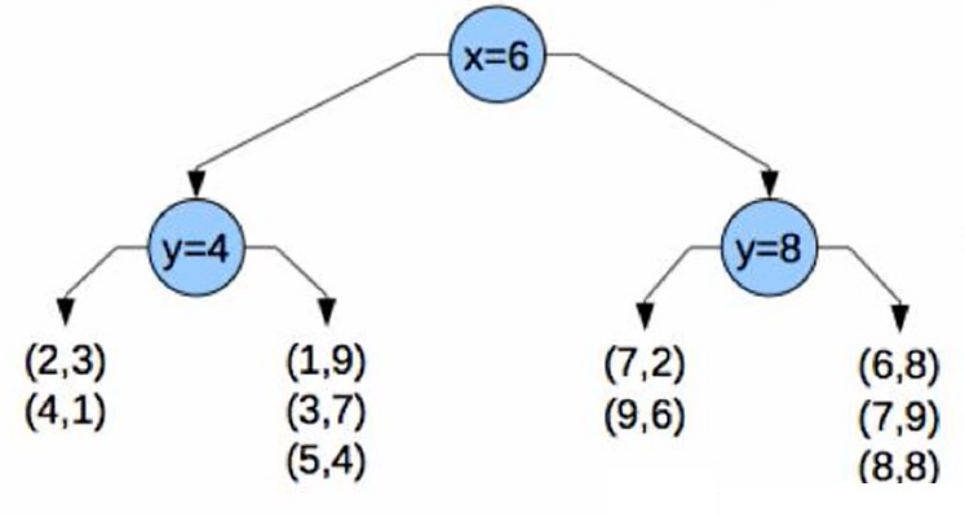


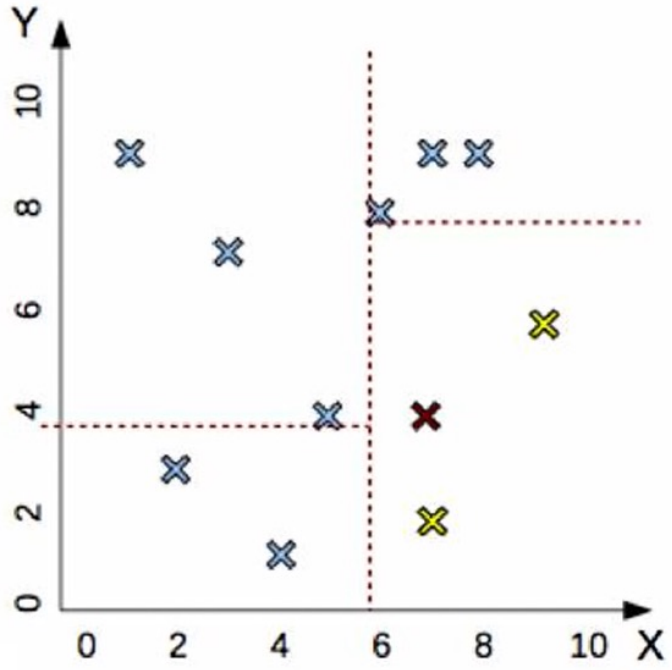


Adapted from Victor Lavrenko

K-Dimensional Trees: Inference

- Find NNs for new point (7,4)
  - find region containing (7,4)
  - compare to all points in region



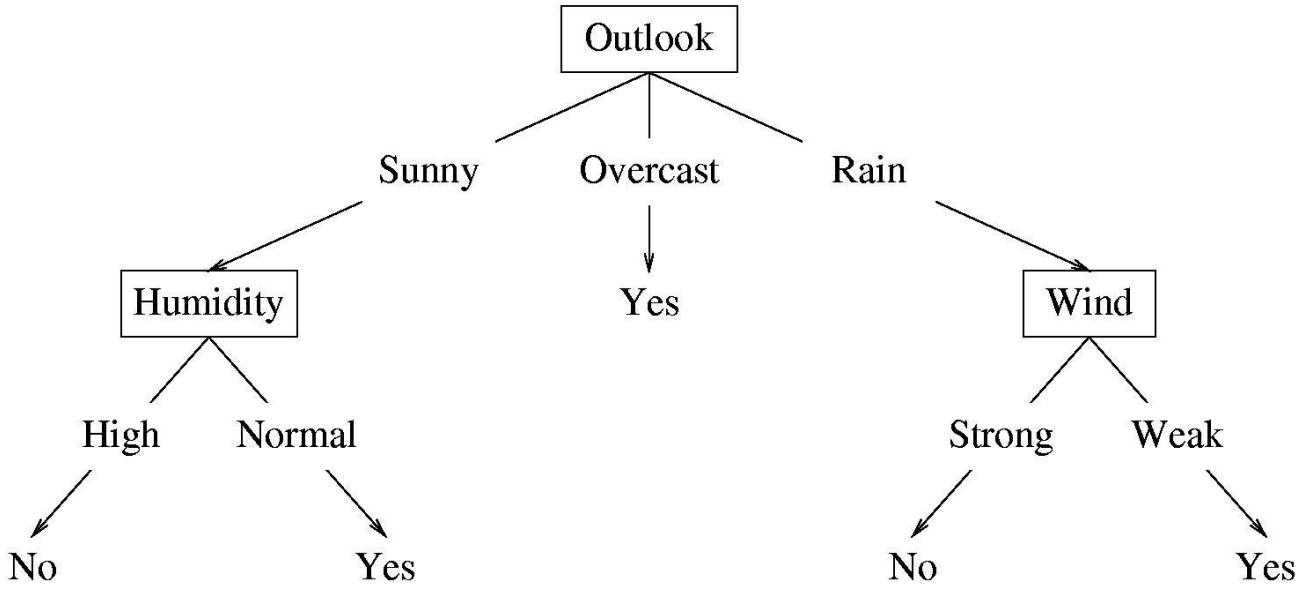


Adapted from Victor Lavrenko

Topics for the Midterm Exam

- Linear Regression
- Perceptron
- Support Vector Machines
- Nearest Neighbor Methods
- **Decision Trees**
- Bayesian Methods
- Naïve Bayes
- Logistic Regression

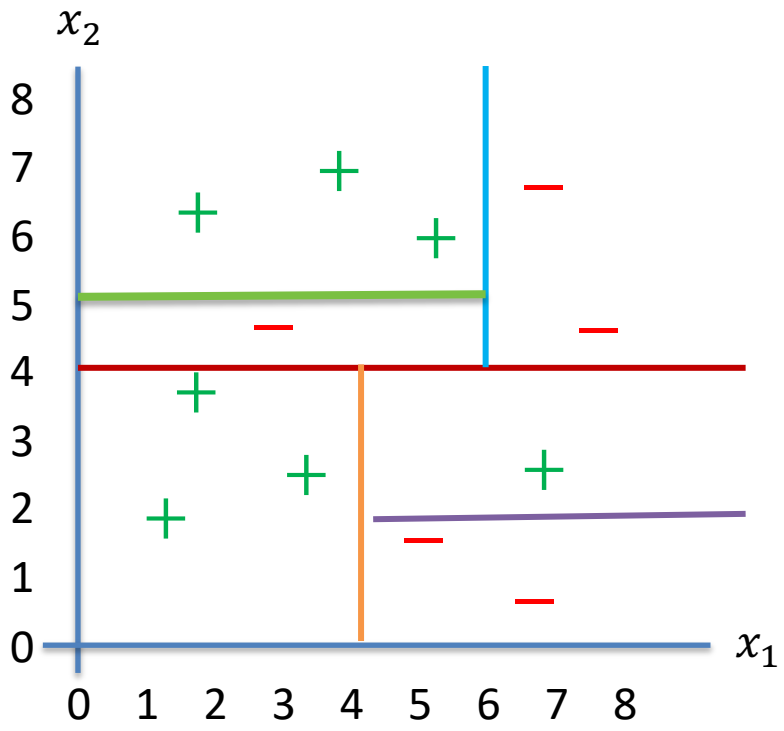
Decision Trees

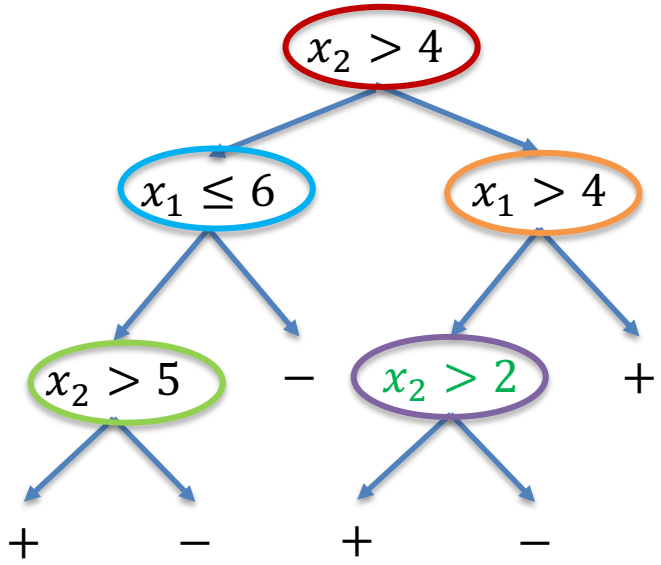


- A tree in which each internal (non-leaf) node tests the value of a particular feature
- Each leaf node specifies a class label (in this case whether or not you should play tennis)

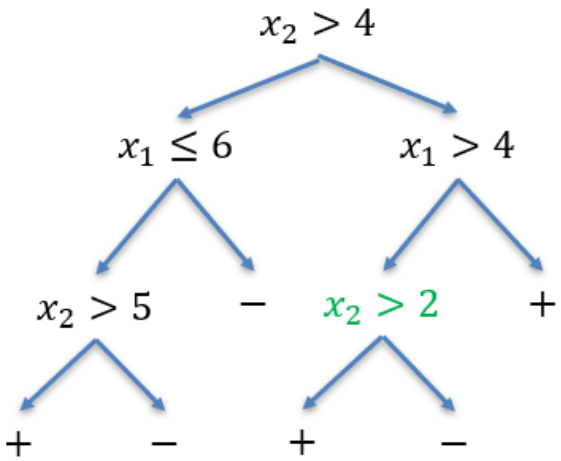
Decision Trees

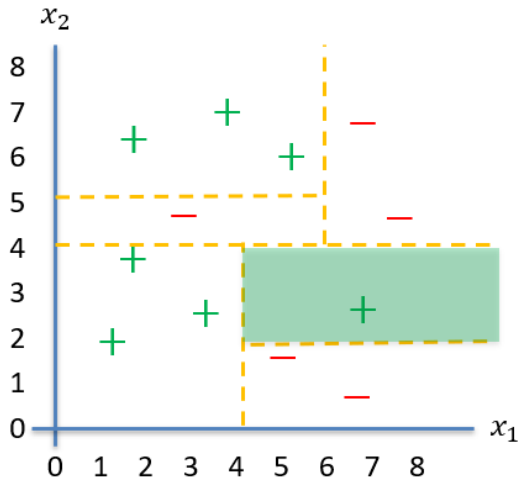
- Decision trees divide the feature space into axis parallel rectangles





Decision Tree Learning



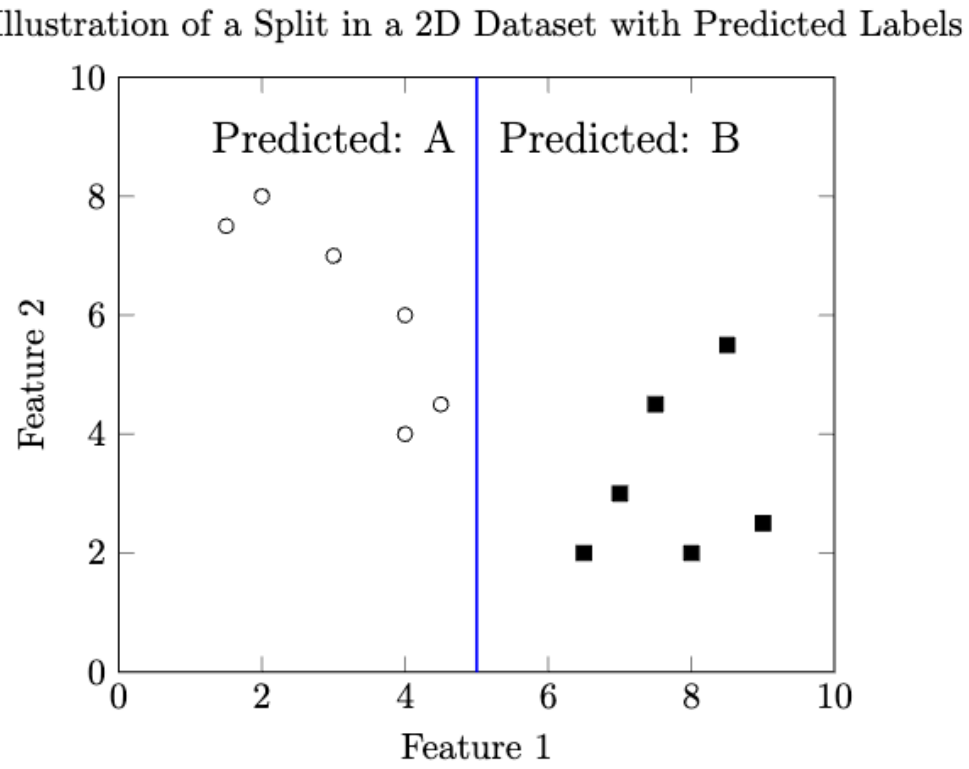


Basic decision tree building algorithm:

Pick some feature/attribute (how to pick the "best"?)	Partition the data based on the value of this attribute	Recurse over each new partition (when to stop?)

We'll focus on the discrete case first (i.e., each feature takes a value in some finite set)

Choosing the Best Attribute to Split



- Splitting on Feature 1 results in homogeneous datasets (i.e., the same label in the two child datasets after the split).
- No split on Feature 2 would achieve this!

Recap: Information Gain

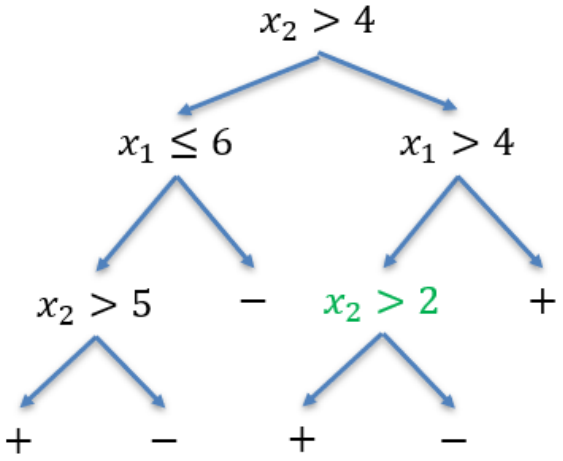
- Using entropy to measure uncertainty, we can greedily select an attribute that guarantees the largest expected decrease in entropy (with respect to the empirical partitions)

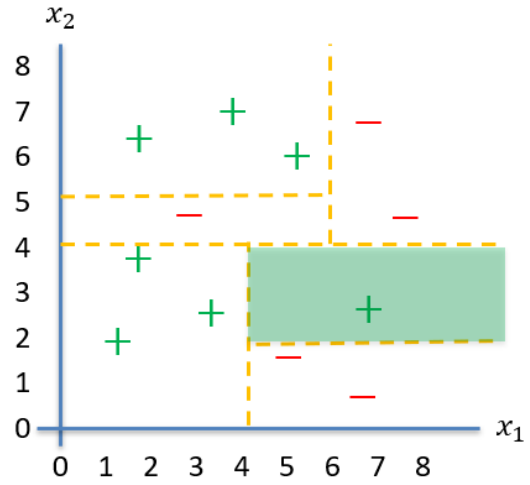
$$IG(X) = H(Y) - H(Y|X)$$

  - Called information gain
  - Larger information gain corresponds to less uncertainty about Y given X
    - Note that $H(Y|X) \leq H(Y)$

Decision Tree Learning

- Basic decision tree building algorithm:
  - Pick the feature/attribute with the highest information gain (or lowest conditional entropy)
  - Partition the data based on the value of this attribute
  - Recurse over each new partition





The Gini Coefficient

- The Gini coefficient is another popular measure used to evaluate splits, focusing on minimizing the probability of misclassification. It is defined for a set S as:

$$Gini(S) = 1 - \sum_{i=1:N} p_i^2$$

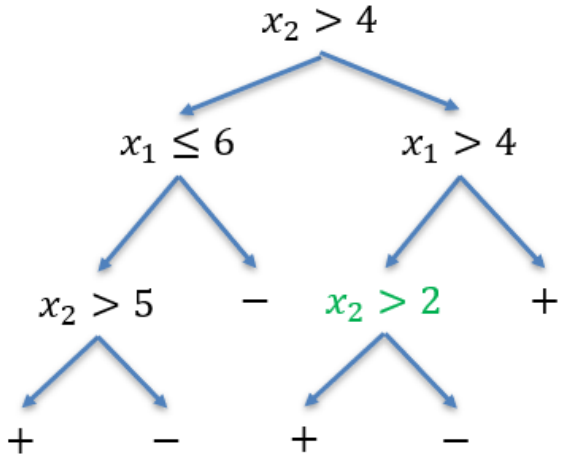
- Once a dataset is split into two sets S_1 and S_2, the Gini-split is defined as:

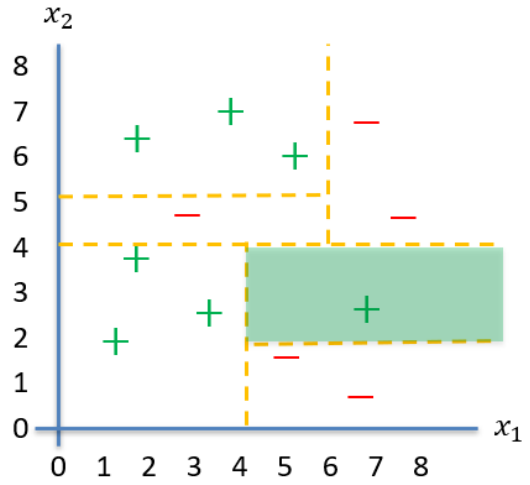
$$GiniSplit = \frac{|S_1|}{|S|} Gini(S_1) + \frac{|S_2|}{|S|} Gini(S_2)$$

- The goal is to find the split that **minimizes the Gini Split.**

Decision Tree Learning

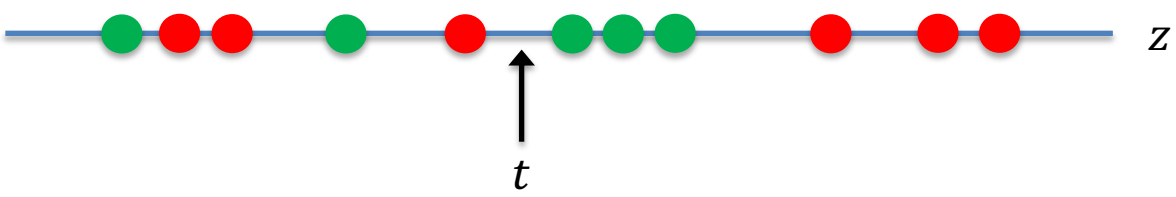
- Basic decision tree building algorithm:
  - Pick the feature/attribute with the lowest gini-split
  - Partition the data based on the value of this attribute
  - Recurse over each new partition





Handling Real-Valued Attributes

- Sort the data according to the k^{th} attribute: $z_1 > z_2 > \cdots > z_n$



- Only a finite number of thresholds make sense
  - Just split in between each consecutive pair of data points (e.g., splits of the form $t = \frac{z_i + z_{i+1}}{2}$)

Regression Decision Trees

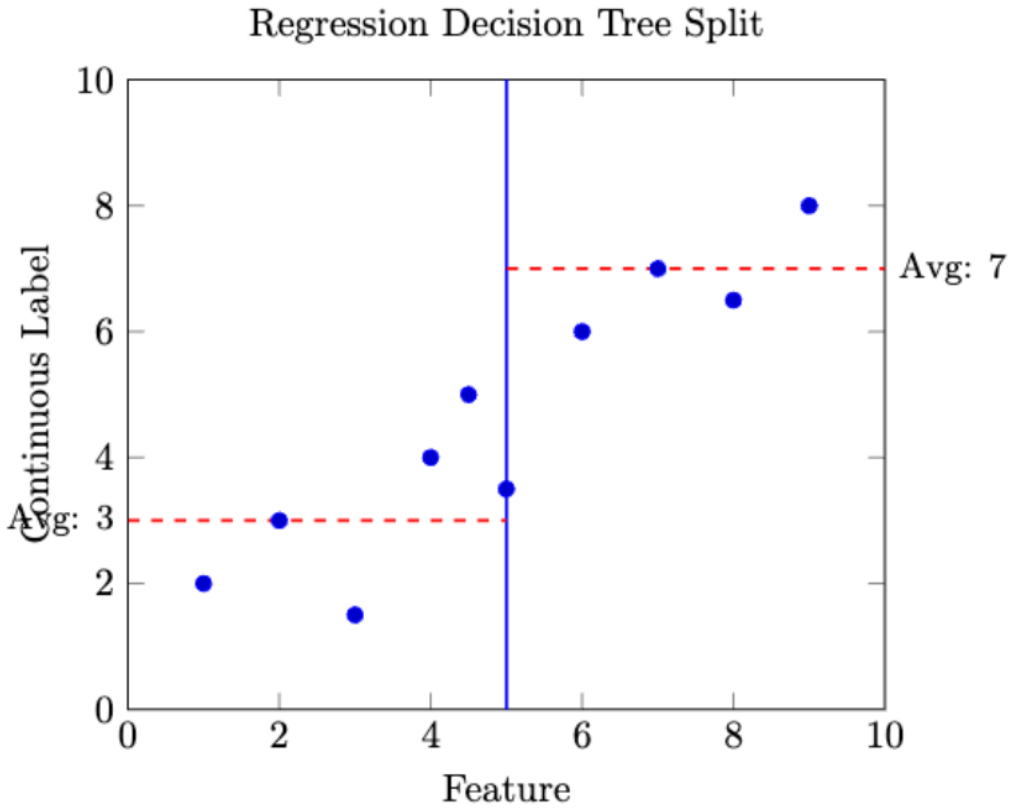


Figure 5: Illustration of a regression decision tree split with a single feature. The dataset is split at Feature = 5, with horizontal dashed lines representing the average label value for each partition.

Regression Splitting Criteria

- MSE Reduction: Calculate the Mean Squared Error of each dataset (i.e. parent dataset and the two children dataset)
- Given a dataset S, the predicted label y_S is the mean of the labels in that set.
- The MSE is then defined as:

$$MSE(S) = \frac{1}{|S|}\sum_{i \in S}(y_i - y_S)^2$$

- We can then define the MSE Reduction as:

$$MSERed = MSE(S) - \frac{|S_1|}{|S|}MSE(S_1) - \frac{|S_2|}{|S|}MSE(S_2)$$

- The goal is to find the split that maximize the MSE Reduction.