THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Applied Machine Intelligence and Reinforcement Learning

Professor Hamza F. Alsarhan

SEAS 8505

Lecture 3

June 29, 2024

Welcome to SEAS Online at George Washington University

Class will begin shortly

Audio: To eliminate background noise, please be sure your audio is muted. To speak, please click the hand icon at the bottom of your screen (Raise Hand). When instructor calls on you, click microphone icon to unmute. When you've finished speaking, *be sure to mute yourself again*.

Chat: Please type your questions in Chat.

Recordings: As part of the educational support for students, we provide downloadable recordings of each class session to be used exclusively by registered students in that particular class for their own private use. Releasing these recordings is strictly prohibited.

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Agenda

- Instance-Based Learning
- Linear Discriminant Analysis
- Support Vector Machines (SVMs)
- Homework Overview

THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Instance-Based Learning & Non-Parametric Approaches

Non-Parametric Estimation

- Parametric (single global model, mean and std dev can describe the function), semiparametric (small number of local models)
- Nonparametric: Similar inputs have similar outputs
- Functions (pdf, discriminant, regression) change smoothly
- Keep the training data; “let the data speak for itself”
- Given x, find a small number of closest training instances and interpolate from these
- Aka lazy/memory-based/case-based/instance-based learning

Lazy vs. Eager Learning

Nonparametric learning is also called Instance Based Learning (IBL)

Lazy Learning Advantages

- Can provide good local approximations in the target function
- Does not require a prior assumptions about the parameters of the data
- Can simultaneously solve multiple problems and deal successfully with changes in the problem domain.

Disadvantages

- Large memory requirements
- Poor performance with noisy training data
- Slow at query time
- Easily fooled by irrelevant attributes

Eager Learning Advantages

- Target function is globally approximated during training
- Requires less memory than lazy learning
- Can handle noisy training data
- Also called “offline” or “batch” learning (e.g. after training is complete, test data does not have an effect on the learned model)
- Once model is learned training data set is no longer needed
- Fast at query time

Disadvantages

- Unable to provide good local approximations in the target function
- Slow training

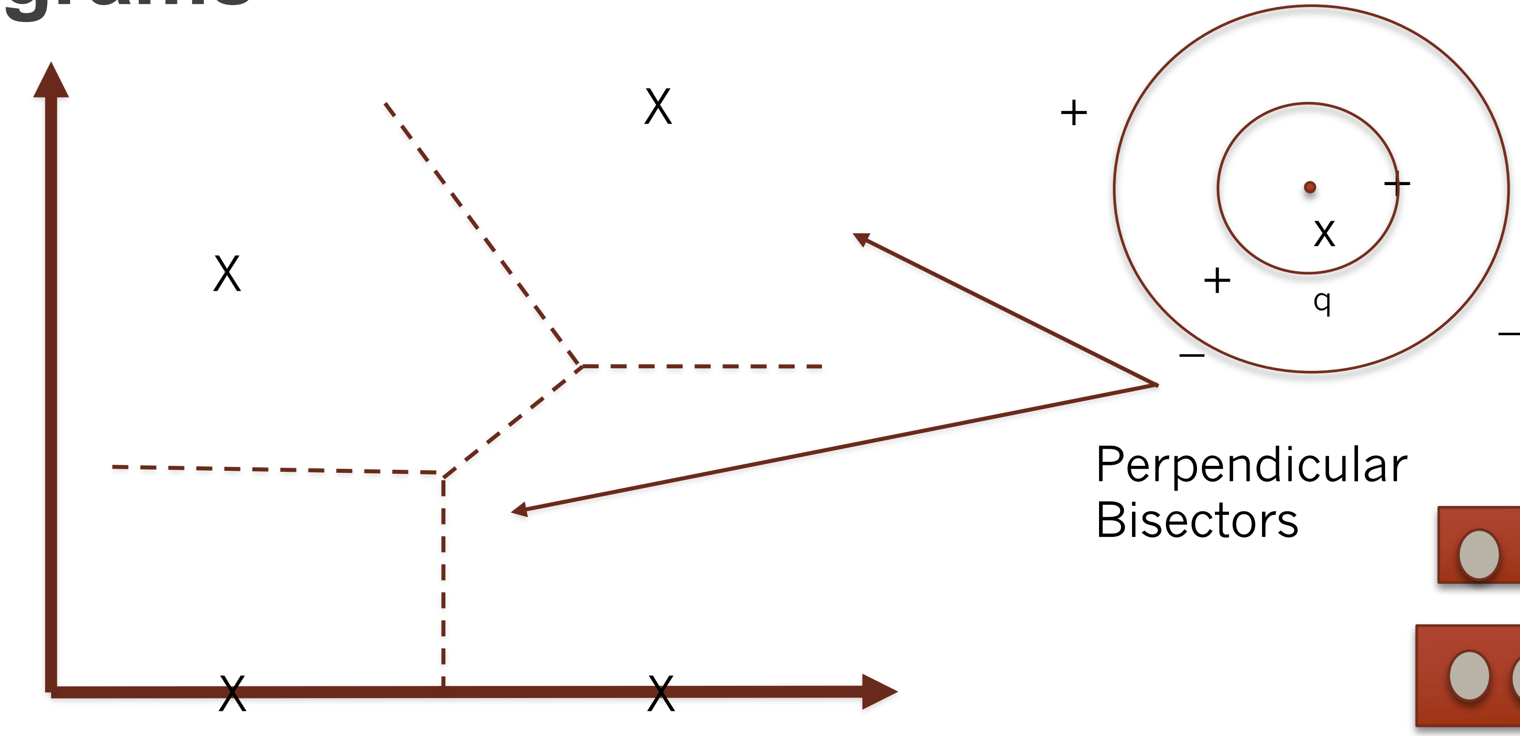
THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

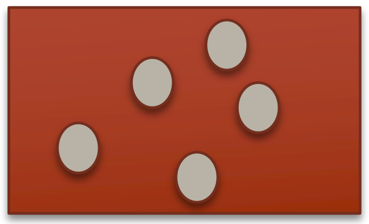
K-NN Overview

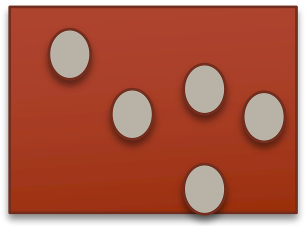
- K-NN has a hyperparameter, K. It is, however, non-parametric because we do not assume anything about the distribution of the data we are dealing with.
- Store all training examples $(x_i, f(x_i))$
- Nearest Neighbor:
  - Given a test instance x_n, locate the nearest training example x_i
  - Make an estimate such that $\hat{f}(x_n) = f(x_i)$
  - K-Nearest Neighbor:
  - Given x_n, take a vote among its k nearest neighbors if target function is discrete-valued
  - Take the mean of f values of k nearest neighbors target function

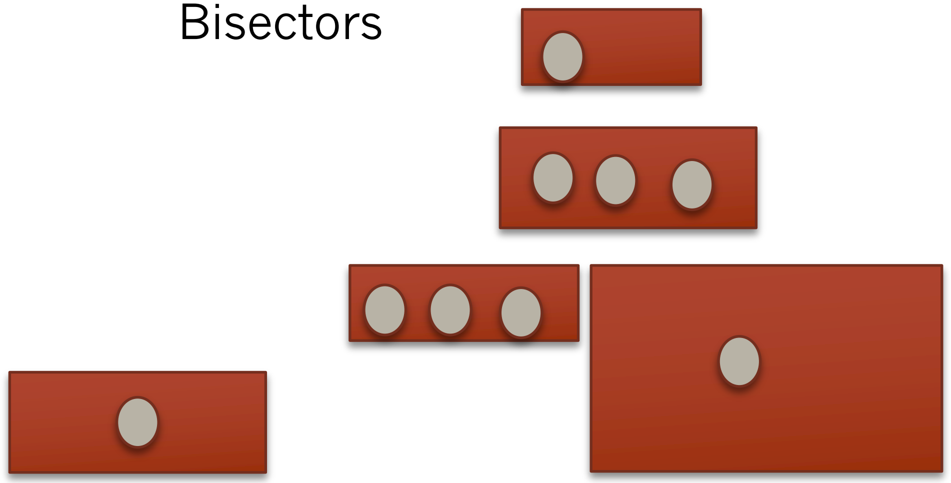
$$\hat{f}(x_n) = \frac{1}{k}\sum_{i=1}^{k} f(x_i)$$

Voronoi Diagrams

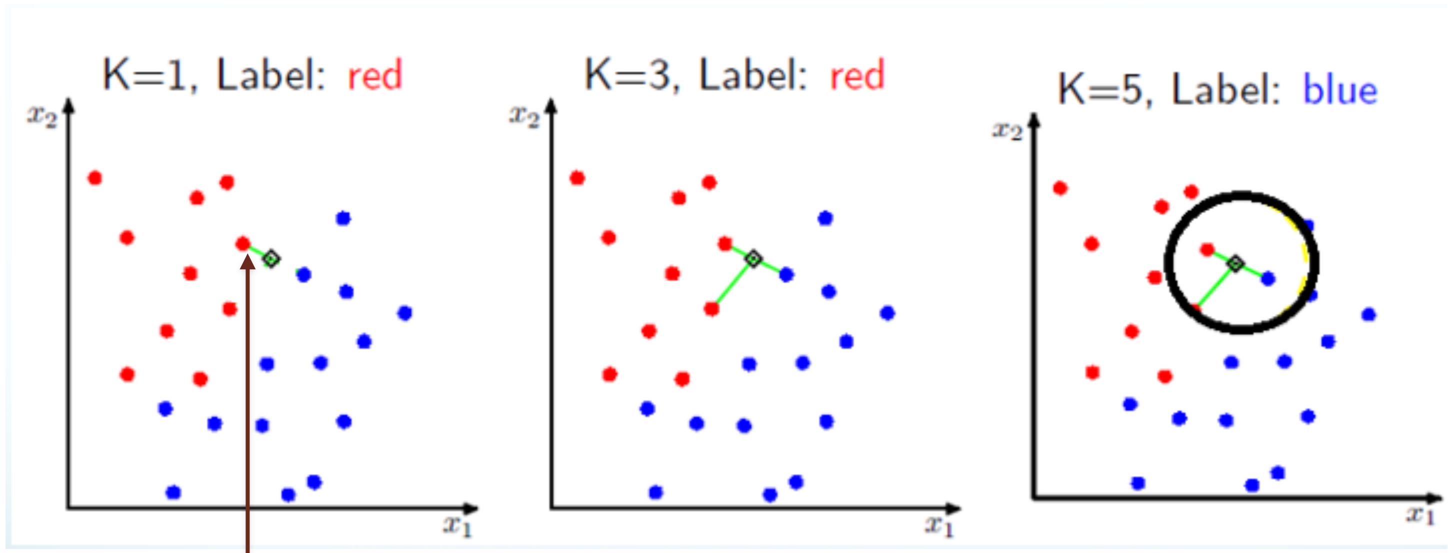








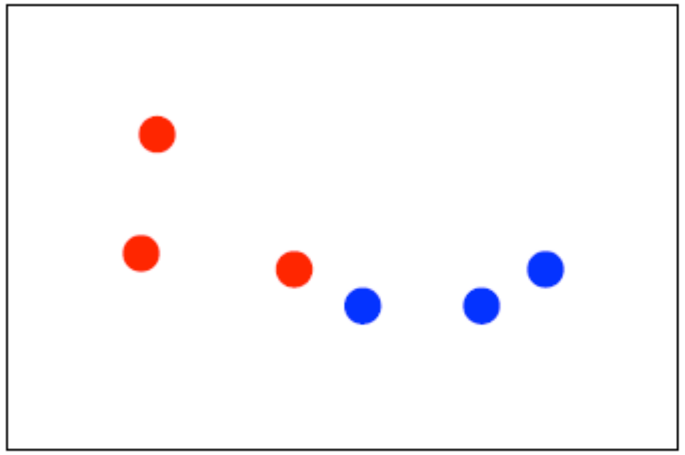
K-NN Examples



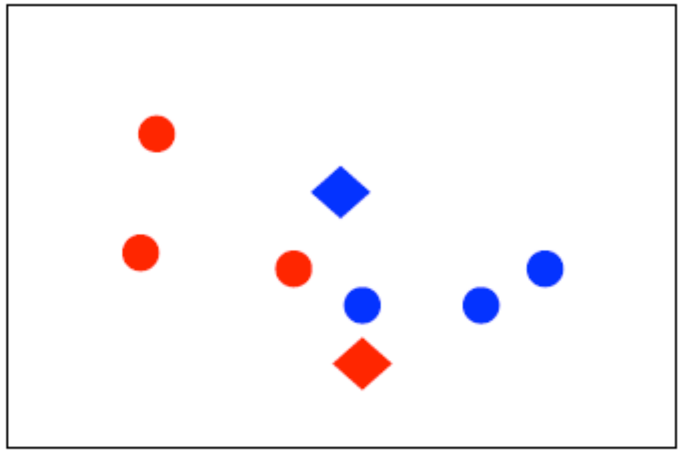
Closest point to test point

1-NN Examples

Training data



Test data



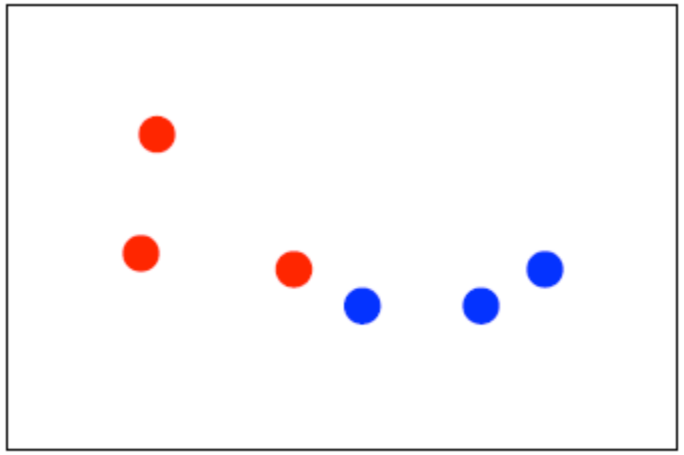
What are A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

What are A^{TEST} and $\varepsilon^{\text{TEST}}$?

$$A^{\text{TRAIN}} = 100\%, \quad \varepsilon^{\text{TRAIN}} = 0\%$$

1-NN Examples

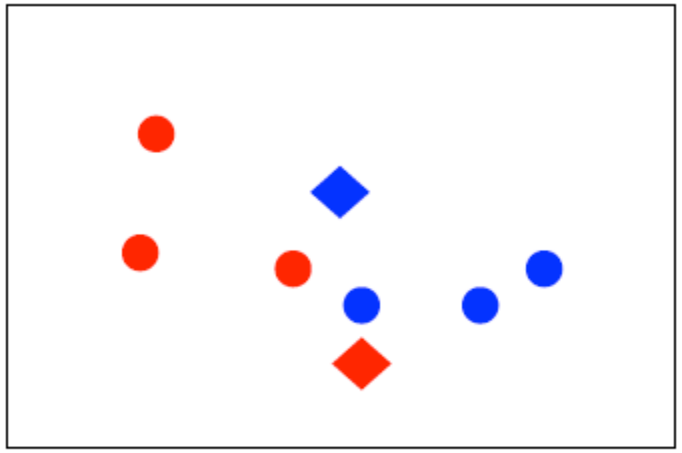
Training data



What are A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

$$A^{\text{TRAIN}} = 100\%, \quad \varepsilon^{\text{TRAIN}} = 0\%$$

Test data



What are A^{TEST} and $\varepsilon^{\text{TEST}}$?

$$A^{\text{TEST}} = 0\%, \quad \varepsilon^{\text{TEST}} = 100\%$$

Ameet Talwalkar

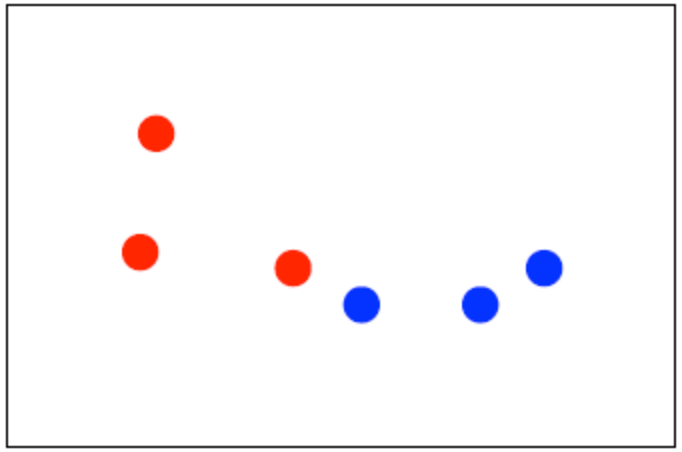
Leave One Out (LOO)

Idea

- For each training instance x_n, take it out of the training set and then label it.
- For NNC, x_n's nearest neighbor will not be itself. So the error rate would not become 0 necessarily.

NNC: Nearest Neighbor Classifier

Training data



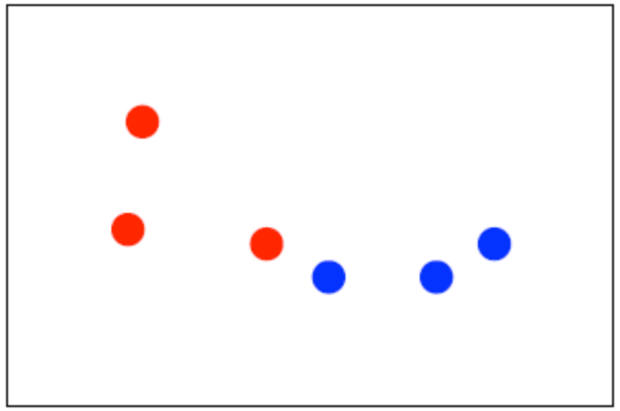
What are the LOO-version of A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

Leave One Out (LOO)

Idea

- For each training instance x_n, take it out of the training set and then label it.
- For NNC, x_n's nearest neighbor will not be itself. So the error rate would not become 0 necessarily.

Training data



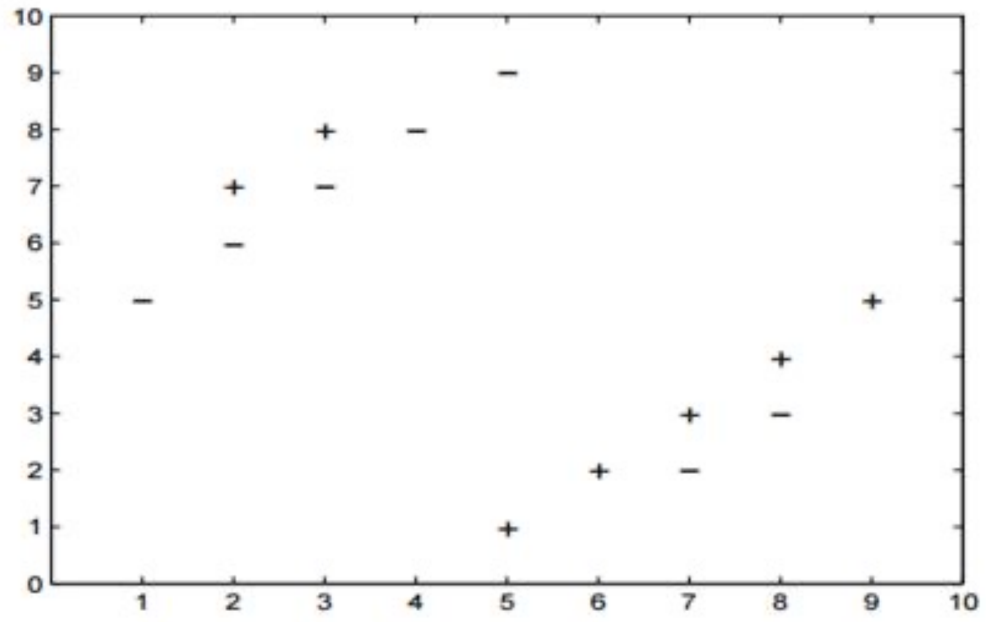
What are the LOO-version of A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

$$A^{\text{TRAIN}} = 66.67\% (\text{i.e.}, 4/6)$$
$$\varepsilon^{\text{TRAIN}} = 33.33\% (\text{i.e.}, 2/6)$$

A Bit More Challenging Example

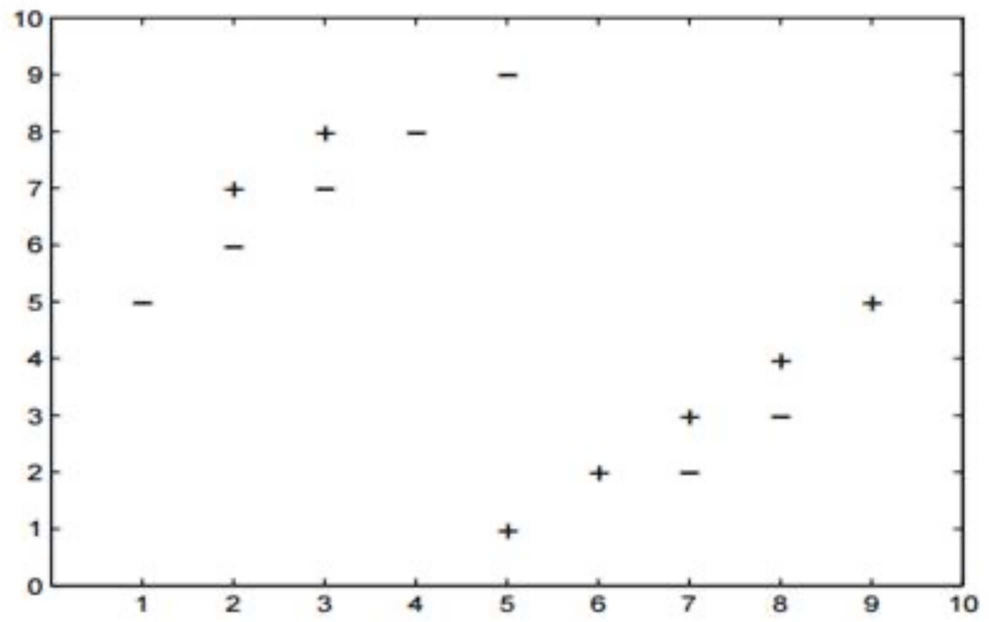
- Suppose you have given the following 2-class data where “+” represents a positive class and “−” represents a negative class.

Why might using too large values k be bad in this dataset? Why might too small values of k also be bad?
What’s the impact of k on bias and variance?



A Bit More Challenging Example

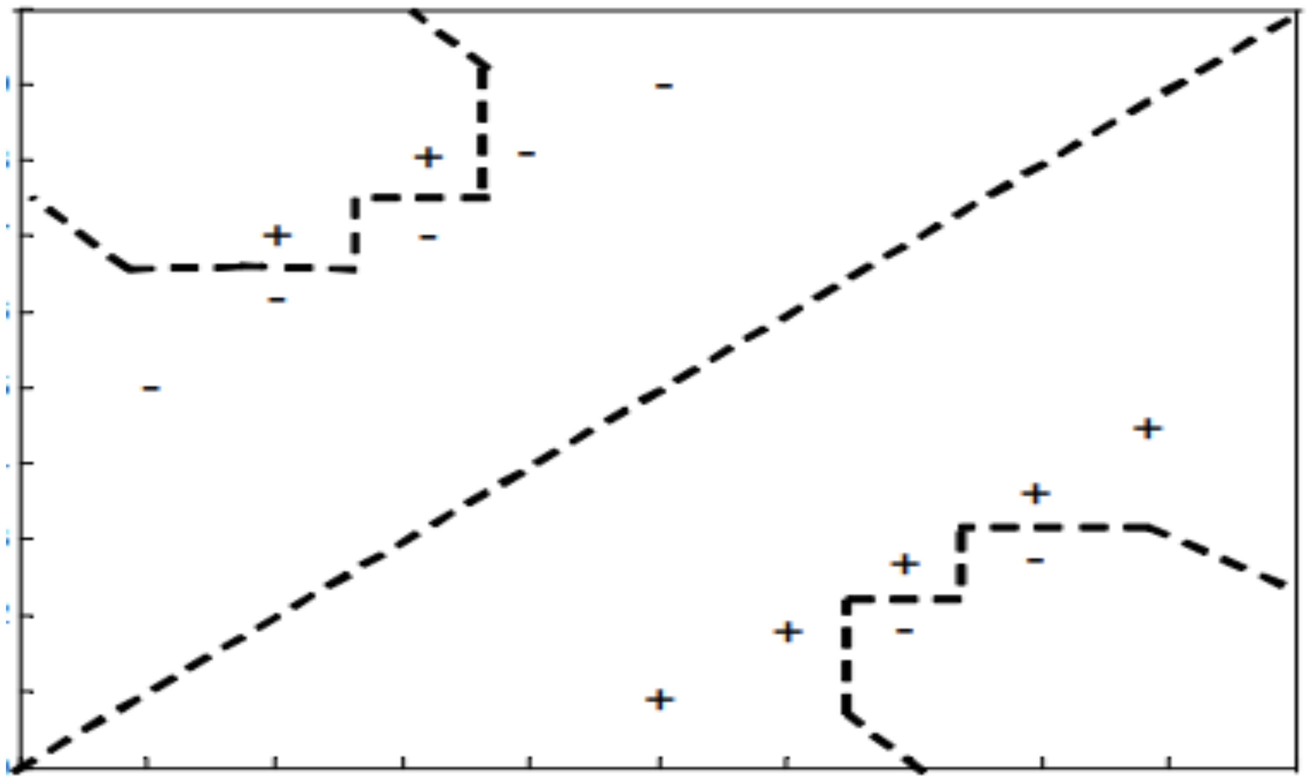
- Suppose you have given the following 2-class data where “+” represents a positive class and “−” represents a negative class.



- k is too small, high variance, sensitive to each data point changing (overfitting)
- k is too large, high bias, smooth average of the outcomes of many neighbors (underfitting)

K-NN In Action

What is k?



What is being depicted in this diagram?

Notes to Ponder

- k = 5 or k = 7 minimizes the leave-one-out cross-validation error. The error is 4/14.
- k = 13 misclassifies every datapoint (using leave one out cross validation).
- Too small k leads to overfitting.

How to Choose *k*?

- When k is small, single instances matter; bias is small, variance is large: High complexity.

- As k increases, we average over more instances and variance decreases but bias increases: Low complexity.

- Cross-validation is used to fine tune k.

Distance Measures

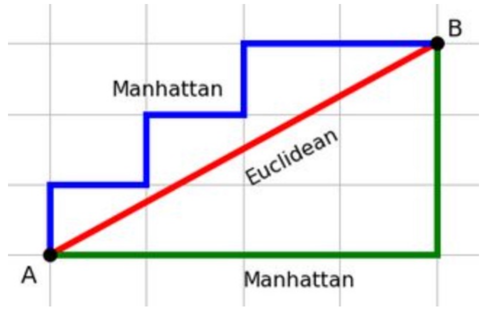
- Numerical: Manhattan

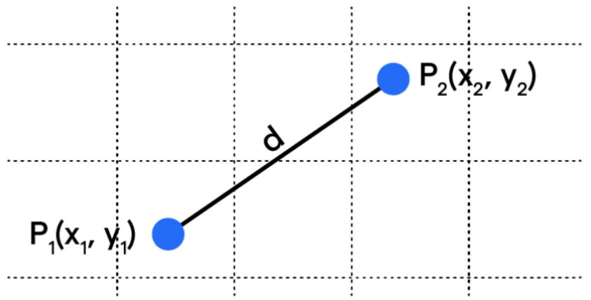
Manhattan $d(x_1, x_2) = \sum_{i=1}^{m} | x_i - y_i|$

For two points: |x1-x2| + |y1-y2|

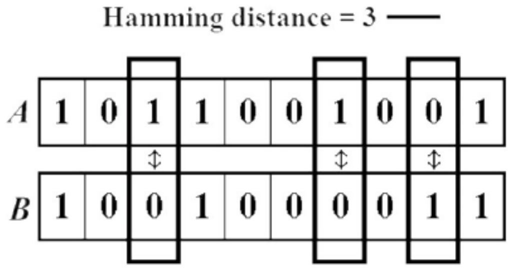
- Numerical: Euclidean
- Symbolic Features/Categorical
  - Hamming distance (Overlap)
  - Value Difference Measure:

$$\delta(val_i, val_j) = \sum_{h=1}^{\#\,classes} |P(c_h|val_i) - P(c_h|val_j)|^n$$





Euclidean Distance (d) $= \sqrt{(x_2 - y_1)^2 + (y_2 - y_1)^2}$



Measuring the distance between two binary strings – just counting the differences

Overfitting Avoidance

- Set k by cross validation
- Form Prototypes – cluster data into groups and select a prototype from each cluster. When presented with a new data point, measure its distance to the prototype from each cluster to determine which class it belongs to. Lower overfitting risk because it is less sensitive to noise and focuses on major patterns in the dataset
- Remove Noisy Instances/Samples
  - e.g. Remove **x** if all of **x**'s k nearest neighbors belong to a different class

Hyperparameter Search Tip

- A common approach to finding the right hyperparameter values is to use grid search (Chapter 2).

- It is often faster to first do a very coarse grid search, then a finer grid search around the best values found.

- Having a good sense of what each hyperparameter actually does can also help you search in the right part of the hyperparameter space.

Outlier Detection

- Find outlier/novelty points. K-distance: measuring the distance for each point to its kth neighbor and identifying outliers based on significantly larger k-distances.
- Not a two-class problem because outliers are very few, of many types, and seldom labeled.
- Instead, one-class classification problem: Find instances that have low probability of belonging to the one class (normal class).
- In nonparametric cases like K-NN, which do not assume a specific distribution of the data: Find instances far away from other instances (distance-based method).

K-NN Summary

- Advantages
  - Fast Training
  - Learns complex target functions easily
  - Retains information (e.g. training set)
- Disadvantages
  - Slow at testing and inference time
  - Needs a lot of storage
  - Easily fooled by irrelevant attributes & noise

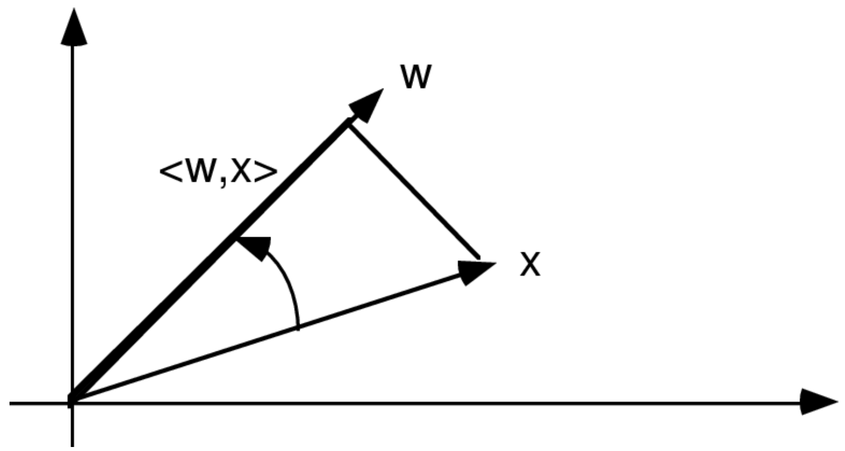
THE GEORGE WASHINGTON UNIVERSITY

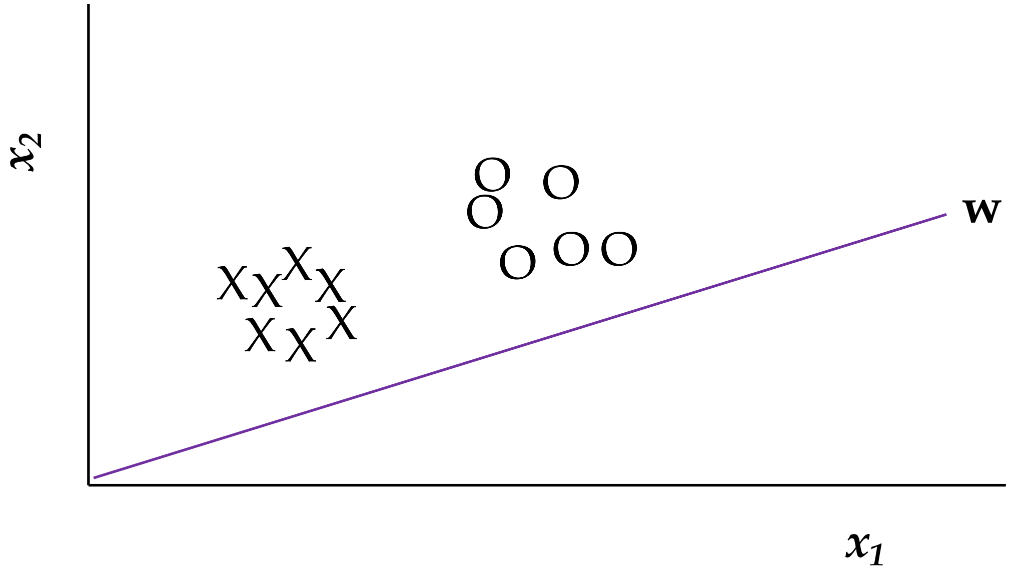
WASHINGTON, DC

Linear Discriminant Analysis

Problem Setting

- $z = \boldsymbol{w}^T\boldsymbol{x}$ is the projection of $\boldsymbol{x}$ onto $\boldsymbol{w}$ and thus is a dimensionality reduction from d to 1.
- We want $z = \boldsymbol{w}^T\boldsymbol{x}$ such that the "between class variance" is maximized relative to "within class variance"





We want to find the direction, as defined by a vector w, such that when the data are projected onto w, the examples from the two classes are as well separated as possible.

Problem Components

Find a low-dimensional space such that when ***x*** is projected, classes are well-separated.

$\bar{\boldsymbol{m}}_{\boldsymbol{1}}$ & $\bar{\boldsymbol{m}}_{\boldsymbol{2}}$ are the means of Class 1 & 2

m_1 & m_2 are the means after projection

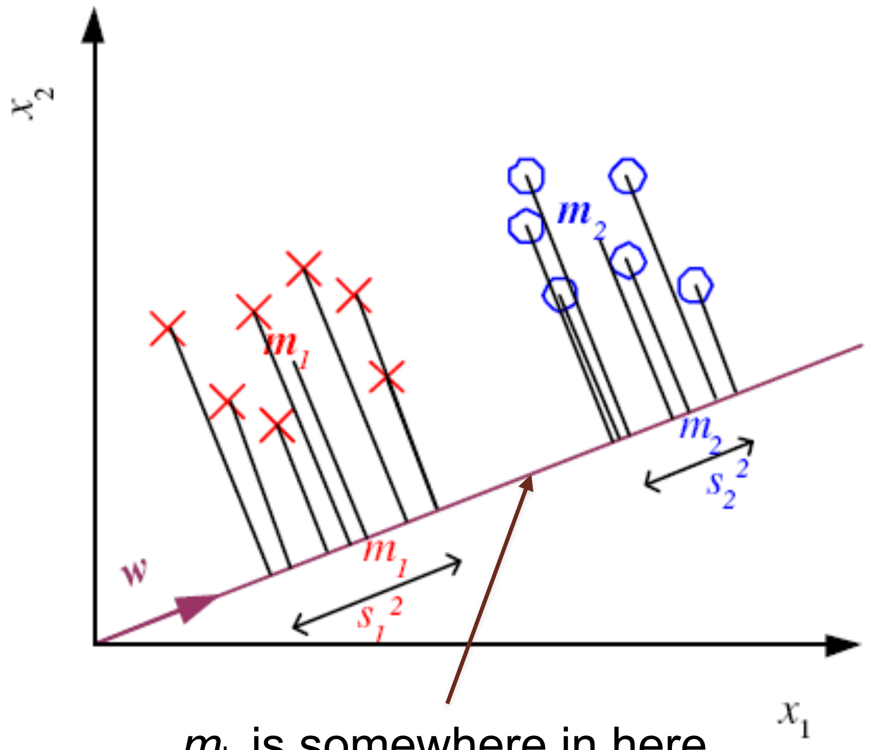
$m_k = \boldsymbol{w}^{\boldsymbol{T}} \bar{\boldsymbol{m}}_{\boldsymbol{k}}$ is the projected mean

Our goal is to maximize:

$\boldsymbol{w}^{\boldsymbol{T}}(\bar{\boldsymbol{m}}_{\boldsymbol{1}} - \bar{\boldsymbol{m}}_{\boldsymbol{2}})$ or alternatively:

argmax $\boldsymbol{w}^{\boldsymbol{T}}(\bar{\boldsymbol{m}}_{\boldsymbol{1}} - \bar{\boldsymbol{m}}_{\boldsymbol{2}})$ such that $\sum_i w_i^2 = 1$

We want their means to be as far apart as possible!



m_k is somewhere in here

m_k is the projected mean

Scatter

We are given a sample $\mathcal{X} = \{\boldsymbol{x}^t, r^t\}$ such that $r^t = 1$ if $\boldsymbol{x}^t \in C_1$ and $r^t = 0$ if $\boldsymbol{x}^t \in C_2$.

$$m_1 = \frac{\sum_t \mathbf{w}^T \mathbf{x}^t r^t}{\sum_t r^t} = \mathbf{w}^T \boldsymbol{m}_1$$

$$m_2 = \frac{\sum_t \mathbf{w}^T \mathbf{x}^t (1-r^t)}{\sum_t (1-r^t)} = \mathbf{w}^T \boldsymbol{m}_2$$

We define the ***scatter*** of samples after the projection as:

$$\text{Variance} \left\{ \begin{aligned} s_1^2 &= \sum_t (\mathbf{w}^T \mathbf{x}^t - m_1)^2 r^t \\ s_2^2 &= \sum_t (\mathbf{w}^T \mathbf{x}^t - m_2)^2 (1 - r^t) \end{aligned} \right.$$

NOTE: Remember that $\mathrm{Var}(X) = \underbrace{E[\,(X - m)^2\,]}_{\text{Expected value}}$ where m is the expected value E(X) ~ the mean of X

Linear Discriminant Analysis

- Find ***w*** that maximizes:

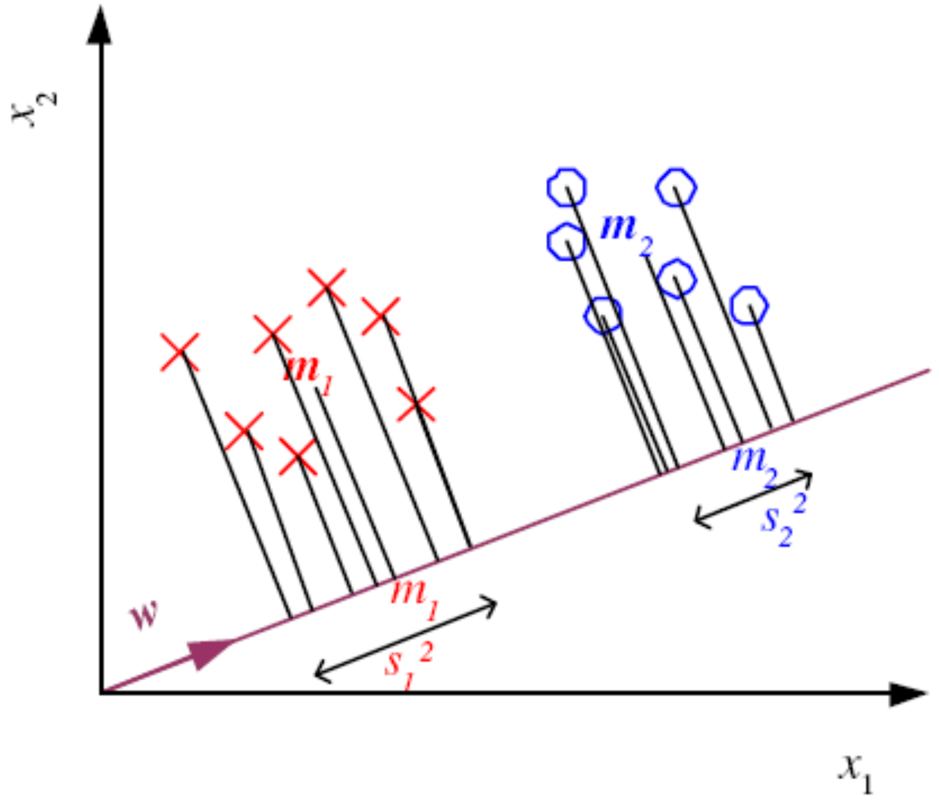
$$J(\mathbf{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

$$m_1 = \frac{\sum_t \mathbf{w}^T \mathbf{x}^t r^t}{\sum_t r^t}$$

$$s_1^2 = \sum_t (\mathbf{w}^T \mathbf{x}^t - m_1)^2 r^t$$

- The ***w*** that maximizes $J(\mathbf{w})$ *is called*:

$$Fisher's\ Linear\ Discriminant$$



Linear Discriminant Analysis

Between-class scatter (rewriting the numerator):

$$\begin{aligned}(m_1 - m_2)^2 &= (\mathbf{w}^T\mathbf{m}_1 - \mathbf{w}^T\mathbf{m}_2)^2 \\ &= \mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T\mathbf{w} \\ &= \mathbf{w}^T \mathrm{S}_B \mathbf{w} \ \text{ where } \ \mathrm{S}_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T\end{aligned}$$

Within-class scatter:

$$\begin{aligned}s_1^2 &= \sum_t (\mathbf{w}^T\mathbf{x}^t - m_1)^2 r^t \\ &= \sum_t \mathbf{w}^T(\mathbf{x}^t - \mathbf{m}_1)(\mathbf{x}^t - \mathbf{m}_1)^T\mathbf{w}r^t = \mathbf{w}^T\mathrm{S}_1\mathbf{w}\end{aligned}$$

$$\text{where } \ \mathrm{S}_1 = \sum_t (\mathbf{x}^t - \mathbf{m}_1)(\mathbf{x}^t - \mathbf{m}_1)^T r^t$$

$s_1^2 + s_2^2 = \mathbf{w}^T\mathrm{S}_W\mathbf{w}$ where $\mathrm{S}_W = \mathrm{S}_1 + \mathrm{S}_2$ **is the total within class scatter**

Fisher's Linear Discriminant

Find w that maximizes: $J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}} = \frac{|\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)|^2}{\mathbf{w}^T S_W \mathbf{w}}$

LDA solution: $\mathbf{w} = c \cdot S_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$

Parametric solution:

$$\mathbf{w} = \Sigma^{-1}(\mu_1 - \mu_2)$$

$$\text{when } p(\mathbf{x}|C_i) \sim N(\mu_i, \Sigma)$$

Key Ideas:

Fisher's linear discriminant is optimal if the classes are normally distributed.
Fisher's linear discriminant can be used even when the classes are not normal.
We have projected the samples from d dimensions to 1.

THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Support Vector Machines

Overview

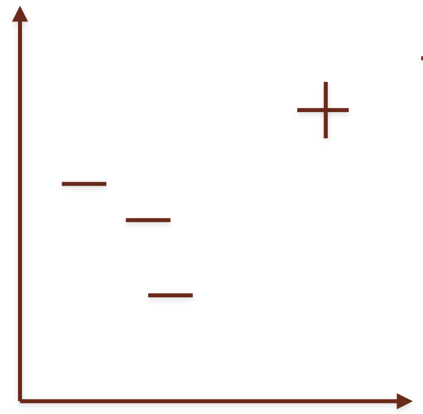
- A Support Vector Machine (SVM) is a powerful and versatile Machine Learning model, capable of performing linear or nonlinear classification, regression, and even outlier detection.
- It is one of the most popular models in Machine Learning, and anyone interested in Machine Learning should have it in their toolbox.
- SVMs are particularly well suited for classification of complex small- or medium- sized datasets.

Practical Considerations for SVMs

- SVMS have been used for a wide variety of applications including:
- Image classification
- Text classification
- Fraud detection
- Recommender systems

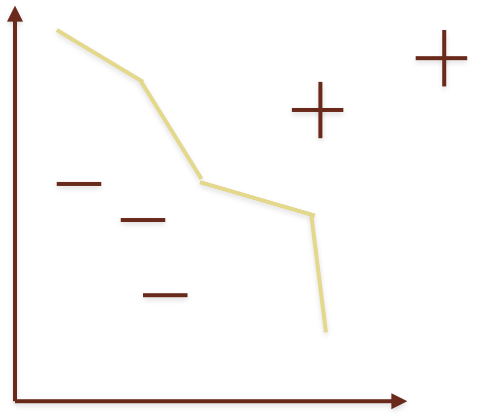
https://medium.com/@mun.articles/svms-in-practice-applications-and-use-cases-for-machine-learnings-most-effective-model-2ae25f4207ef

How do we best separate these samples?



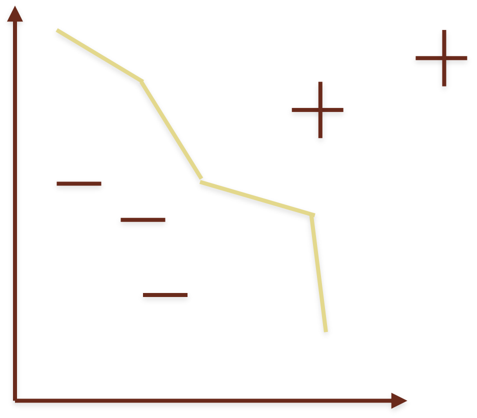
It’s all about decision boundaries!

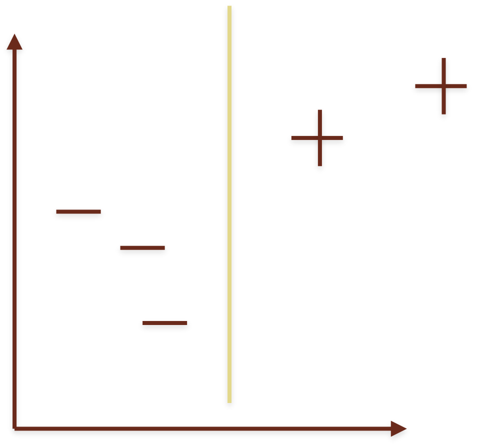
We could use K-NN



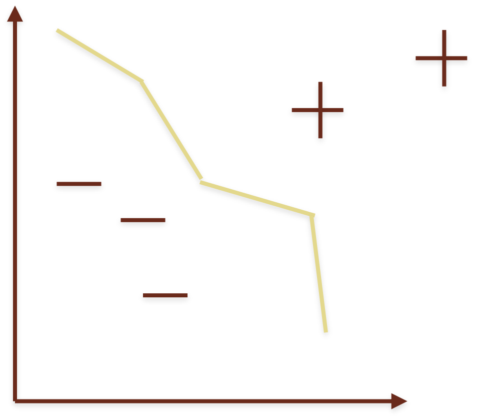
K-NN

We could use a Decision Tree

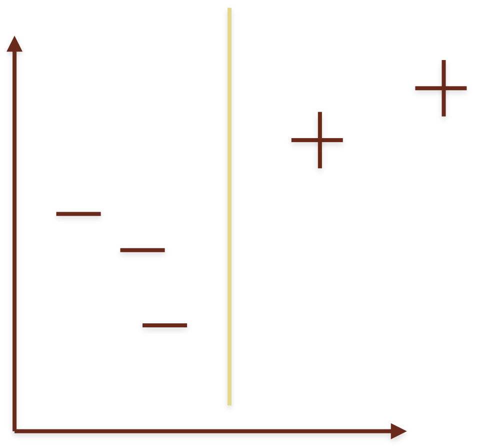

K-NN


Decision Tree

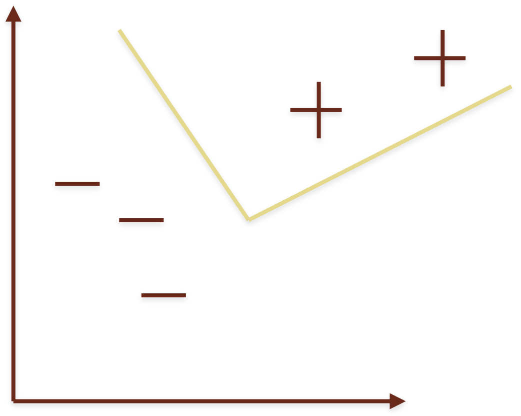
We could use a Neural Network



K-NN



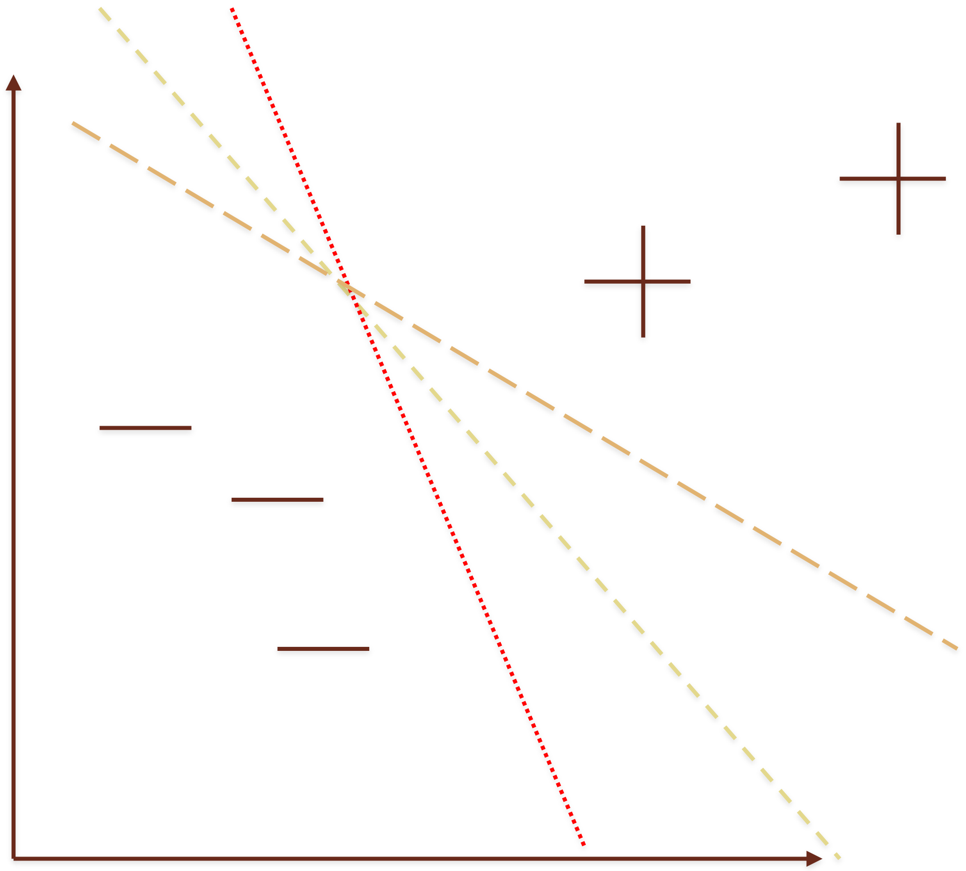
Decision Tree

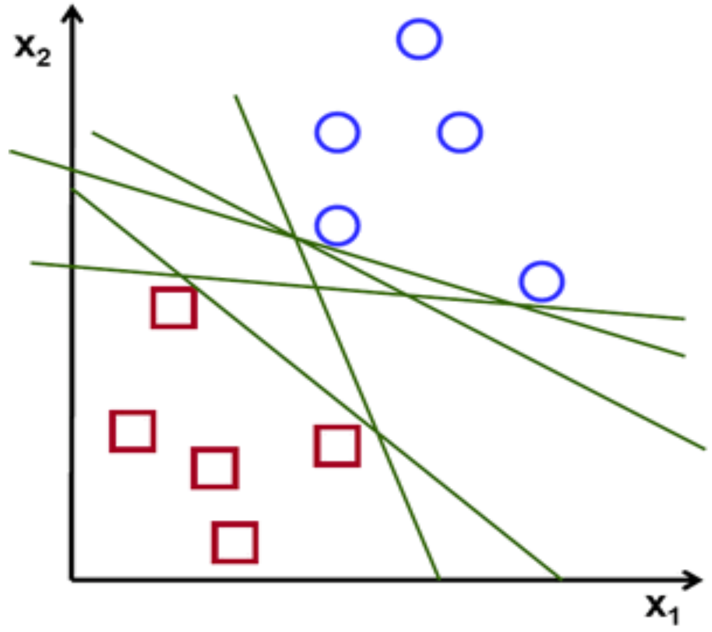


Neural Network

Based on Alpaydin Introduction to Machine Learning

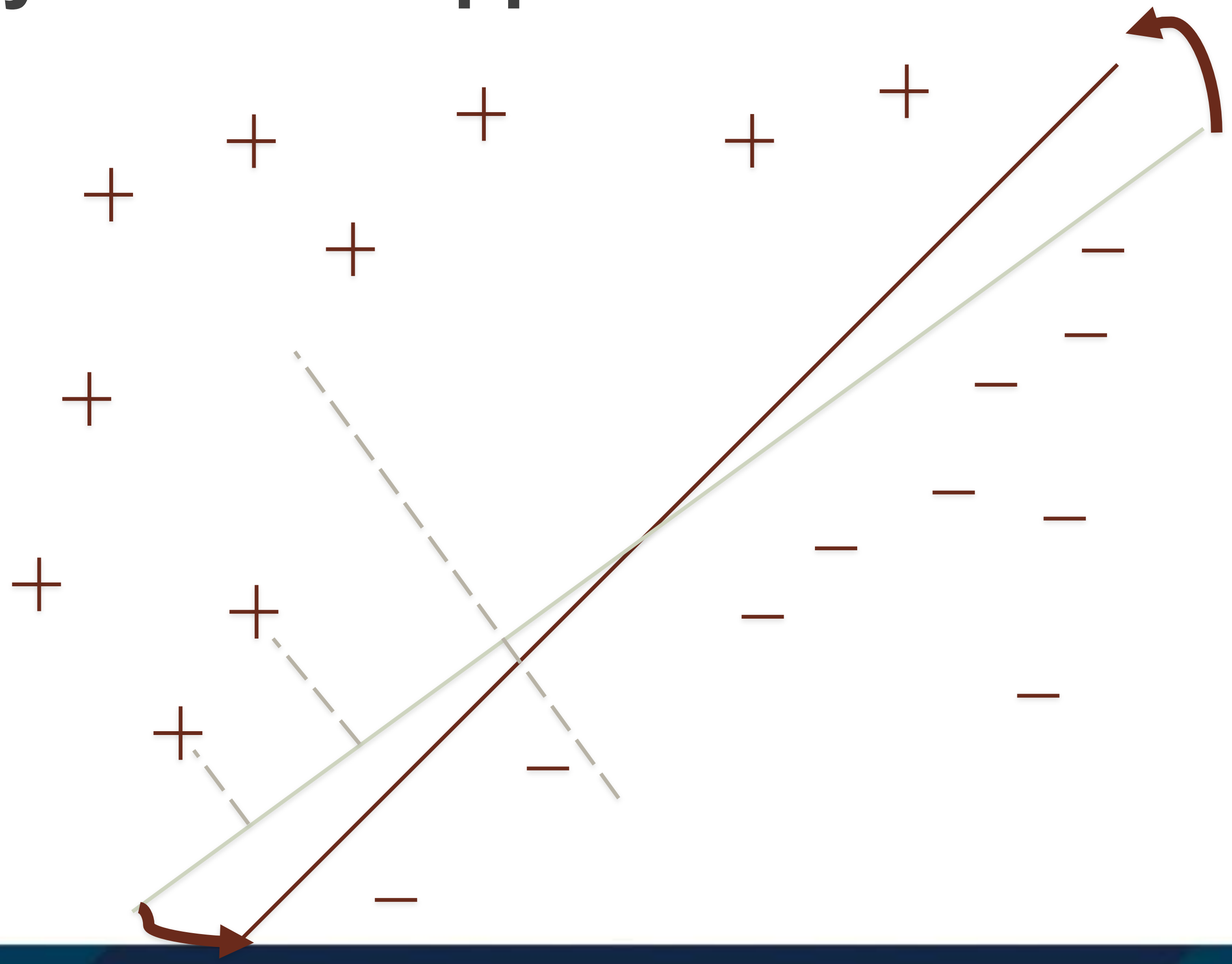
But Which One is the “Best”?





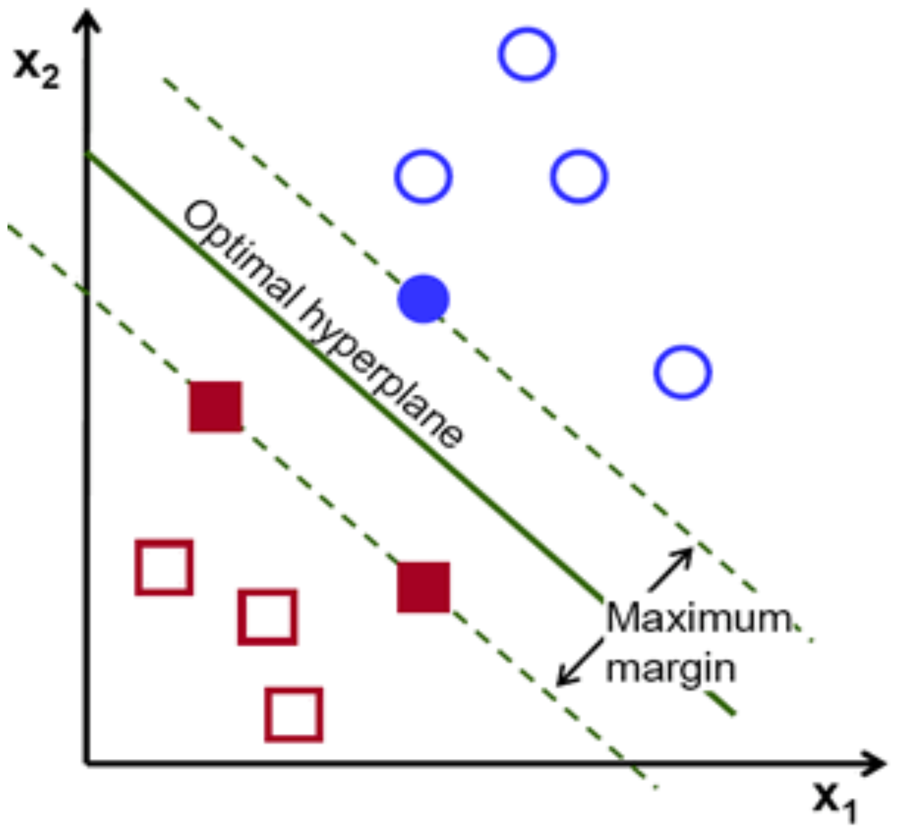
Based on Winston

We Need a Systematic Approach



Based on Domingos

The Widest Street

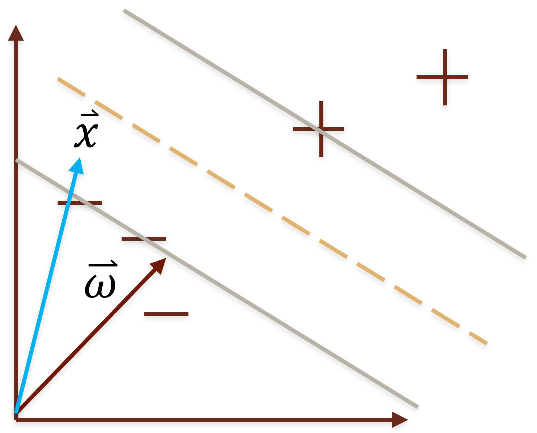


Based on Winston & Sam Kirkiles

Maximizing the Margin

$\vec{x}$ is an unknown sample &
$\vec{\omega}$ is perpendicular to the dividing line

$\vec{w} \cdot \vec{x} \geq c$



margin - The distance between our clusters and the decision line.
gutter - The area on each side of the decision line.
hyperplane - A subspace of one dimension less than its ambient space. (Our decision line)
vector - A quantity having a direction and magnitude.

$\vec{s}$ *is a sample point*. Project $\vec{s}$ *onto* $\vec{w}$ *to determine what* sign it is $\rightarrow$ *what side of the street is it on?*
If $\vec{w} \cdot \vec{x} + b \geq 0$ *then* $+$ (*postive example*) → "Decision Rule"
Note: $c = -b$

In the figure, we have defined a vector w which is the normal (perpendicular) to the gutter. For this example, we are going to use the convention of calling the gutter and decision line the "street". Think of the w as our model weights.

https://medium.com/samkirkiles/support-vector-machines-from-scratch-part-1-intuition-c1e6a0ac026b

Adding Some Constraints

- Let's say we have a known positive sample x_+ then we require:

  $\vec{w} \cdot x_+ + b \geq 1$ for "+" samples

  - Likewise for a known negative sample x_- :

  $\vec{w} \cdot x_- + b \leq -1$ for "$-$" $samples$

- Define y_i such that y_i = 1 for + samples and y_i = -1 for $-$ samples
- Multiply y_i by the two equations above:

$y_i\ (\vec{w} \cdot x_i + b) \geq 1$ for positive examples since (y_i = 1)

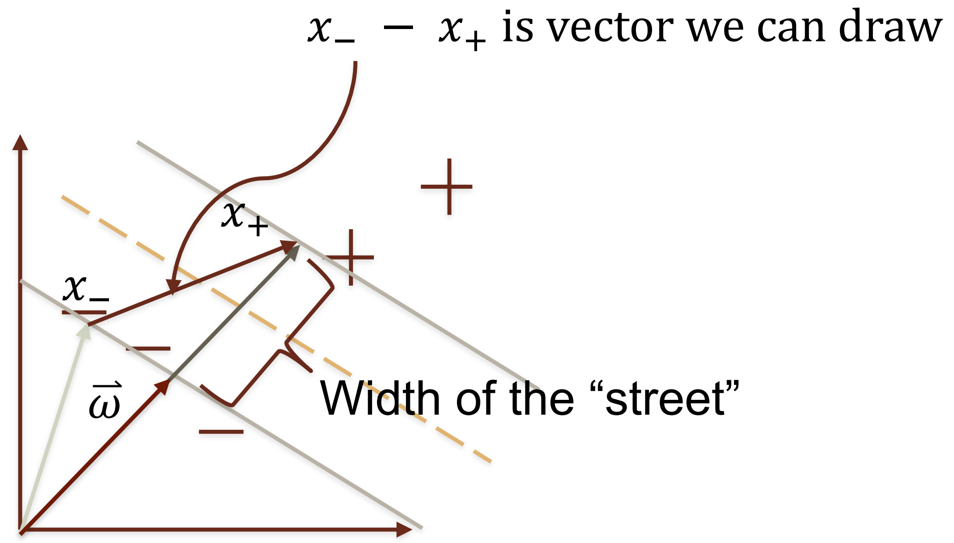
$y_i\ (\vec{w} \cdot x_- + b) \leq -1 \Rightarrow y_i\ (\vec{w} \cdot x_i + b) \geq 1$ for negative examples since (y_i = -1)

- ***So both expressions are the same***: $y_i\ (\vec{w} \cdot x_i + b) \geq 1$ and it follows that:

$y_i\ (\vec{w} \cdot x_i + b) - 1 \geq 0$ AND
$y_i\ (\vec{w} \cdot x_i + b) - 1 = 0$ for all x_i samples that are on the "edge" or in the "gutter"

Based on Winston

Back to the Picture …



- Ultimately, we want to find the width of the "street"
- We need a unit vector perpendicular to the edge in order to take the dot product to measure the width.

Recall that the vector: $\vec{w}$ *is perpendicular to the boundary.*

So the width is = $(x_+ - x_-) \cdot \frac{\vec{w}}{\|\vec{w}\|}$

$Remember\colon\ y_i\ (\vec{w} \cdot x_i + b) - 1 = 0$ from the previous slide

Based on Winston
https://medium.com/samkirkiles/support-vector-machines-from-scratch-part-2-developing-a-decision-boundary-5b2b4b43d280

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

But We Can Even Do Better …

- The width = $(x_+ \; - \; x_-)\cdot \frac{\vec{w}}{\|\vec{w}\|}$
- Since $y_i\ (\vec{w} \cdot x_i + b) - 1 = 0$ for all x_i samples that are on the "edge" we

can plug in for x_+ & x_-:

- If y_i =1 and $x_i = x_+$ then $(\vec{w} \cdot x_+ + b) - 1 = 0$ and $\vec{w} \cdot x_+ = 1 - \; b$

Likewise:

- If y_i =−1 and $x_i = x_-$ then $-(\vec{w} \cdot x_- + b) - 1 = 0$ and $\vec{w} \cdot x_- = -1 - b$
- $(x_+ \; - \; x_-) \cdot \vec{w} \; = (1 - b) - (-1 - b) = 2$
- So the width = $(x_+ \; - \; x_-)\cdot \frac{\vec{w}}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|}$

Of course the max of $\frac{2}{\|\vec{w}\|}$ ➔ which is the min$\frac{\|\vec{w}\|}{2}$

- Why not square the $\|\vec{w}\|$? Why not? ➔ min $\frac{\|\vec{w}\|^2}{2}$ (mathematical convenience)

Based on Winston
https://medium.com/samkirkiles/support-vector-machines-from-scratch-part-3-the-optimization-objective-85e67e22c8bf

Lagrangian

- Is a strategy for finding the local maxima and minima of a function subject to the condition that one or more equations have to be satisfied exactly by the chosen values of the variables

We want to minimize $\frac{\|\vec{w}\|^2}{2}$ with constraints: $y_i\ (\vec{w} \cdot x_i + b) - 1 = 0$

L = $\frac{1}{2}\|\vec{w}\|^2 - \sum \alpha_i[y_i(\vec{w} \cdot \vec{x}_i + b) - 1]$

$\frac{\partial L}{\partial \vec{w}} = \vec{w} - \sum_i \alpha_i y_i \vec{x}_i = 0 \qquad \Rightarrow \qquad \vec{w} = \sum_i \alpha_i y_i \vec{x}_i$

$\frac{\partial L}{\partial b} = -\sum_i \alpha_i y_i = 0 \qquad \Rightarrow \qquad \sum_i \alpha_i y_i = 0$

Note: The Lagrangian gives us a new expression that takes into account the constraints with a summation over constraints.

The $\alpha_i$are the Lagrange Multipliers. The α_i that are non-zero will lie in the “gutter” the rest of the samples have α_i=0!

Lagrangian + Some Algebra

- We know that: ⇒ $\overrightarrow{w} = \sum_i \alpha_i y_i \overrightarrow{x}_i$
- Note that the decision vector is the sum of the samples
- Substituting in for w:

$$L = \frac{1}{2}\|\overrightarrow{w}\|^2 - \sum \alpha_i [y_i(\overrightarrow{w} \cdot \overrightarrow{x}_i + b) - 1]$$

$$L = \frac{1}{2}\left(\sum \alpha_i y_i \vec{x}_i\right)\left(\sum \alpha_j y_j \vec{x}_j\right) - \left(\sum \alpha_i y_i \vec{x}_i\right)\left(\sum \alpha_j y_j \vec{x}_j\right) - \sum \alpha_i y_i b + \sum \alpha_i$$

0 per previous slide

$$L = \frac{1}{2}\left(\sum \alpha_i y_i \vec{x}_i\right)\left(\sum \alpha_j y_j \vec{x}_j\right) - \left(\sum \alpha_i y_i \vec{x}_i\right)\left(\sum \alpha_j y_j \vec{x}_j\right) - b \cancel{\sum \alpha_i y_i} + \sum \alpha_i$$

$$= \sum \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \overrightarrow{x}_i \cdot \overrightarrow{x}_j$$

As it turns out, the Optimization only depends on the dot product of pairs of samples

We want to find the maximum of the above expression.

Update the decision rule: If $\vec{w} \cdot \vec{s} + b \geq 0 \; then \; " + "(postive \; example)$

Substitute for $\overrightarrow{w}$: $\sum_i \alpha_i y_i \vec{x}_i \cdot \overrightarrow{s} + b \geq 0 \; then \; "+"$

Based on Winston

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Scaling

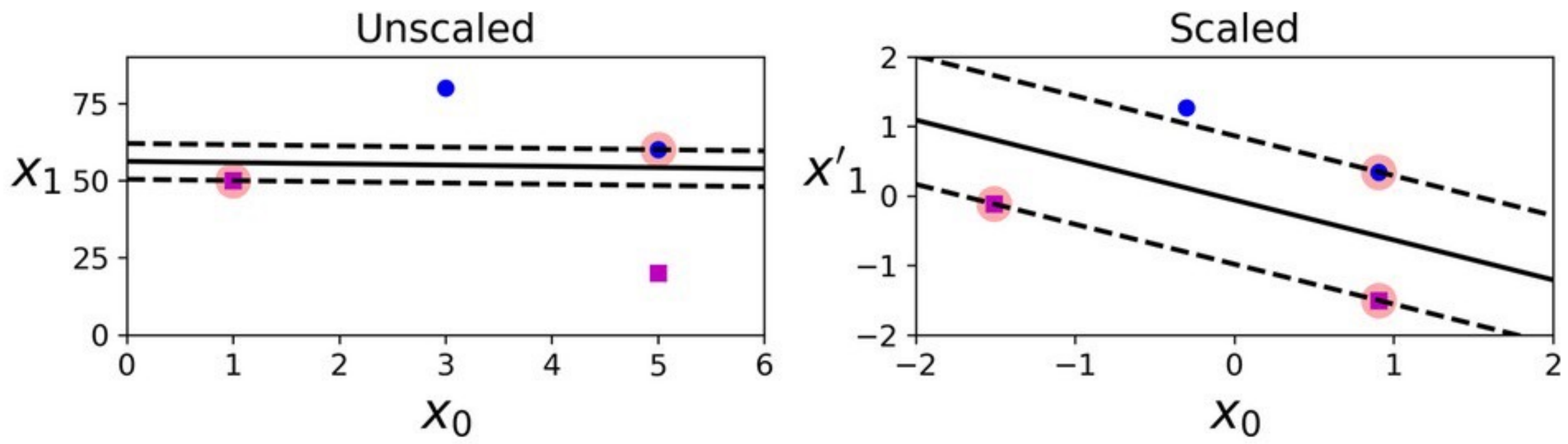


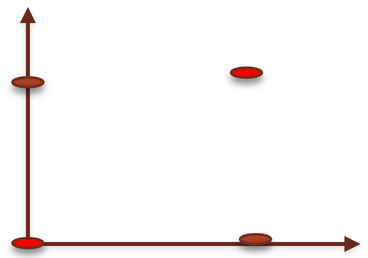
Figure 5-2. Sensitivity to feature scales

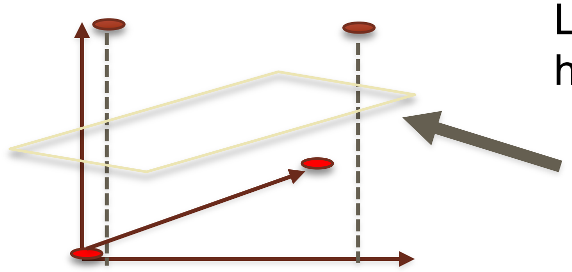
- SVMs are sensitive to the feature scales
  - The left plot - the vertical scale is much larger than the horizontal scale, so the widest possible street is close to horizontal
  - The right plot - after feature scaling, the decision boundary looks much better

Back to the "Big Picture"

What if the samples are not linearly separable?**

$$\sum_i \alpha_i y_i \vec{x}_i \cdot \vec{s} + b \geq 0 \; then \; "+"$$





What can we do? ➔ Transform the Problem to a higher space

(X,Y) space ➔ $\emptyset(x) \cdot \emptyset(y) \;\; to \; maximize$

Based on Winston

***** When stuck switch to a different perspective!***

Kernel Function

Need a transformation to a new space: $\emptyset(\bar{x}_i)$

Transformation from the space we are in to a space where everything is linearly separable (convenient)

$\emptyset(\bar{x}_i) \cdot \emptyset(\bar{x}_j)\ to\ maximize$

$\emptyset(\bar{x}_i) \cdot \emptyset(\bar{s})$ $\leftarrow$ Unknown/Unclassified Sample

$If\ we\ have\ a\ function$: $K(\bar{x}_i, \bar{x}_j) = \emptyset(\bar{x}_i) \cdot \emptyset(\bar{x}_j)$ then we actually do not need $\emptyset(\bar{x}_i)$

Kernel Examples

- Linear: $K(x,x^T) = x \cdot x^T$
- Polynomial: $K(x,x^T) = (x \cdot x^T)^d$
- Gaussian: $K(x,x^T) = \exp(-\frac{1}{2} \| x - x^T \| / \sigma)$

$$\mathbf{x} = \begin{bmatrix} 1,2,3 \end{bmatrix} \qquad \mathbf{x}^T = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Example: Polynomial Kernel

$u = (u_1, u_2)$

$v = (v_1, v_2)$

$$(u \cdot v)^{2} = (u_1 v_1 + u_2 v_2)^{2}$$

$$= {u_1}^2 {v_1}^2 + {u_2}^2 {v_2}^2 + 2 u_1 v_1 u_2 v_2$$

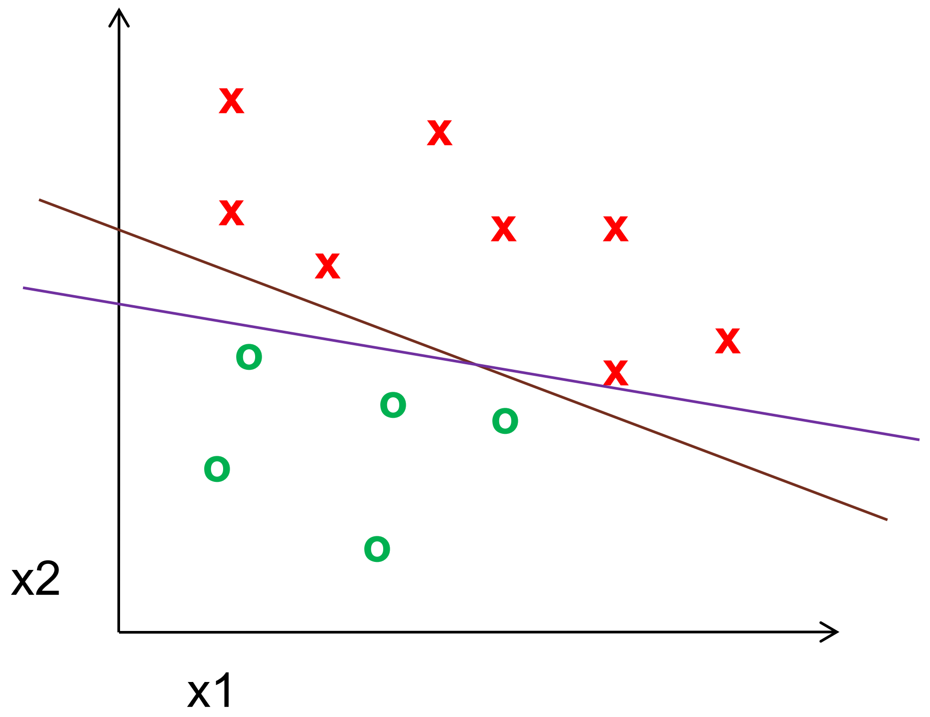
$= ({u_1}^2, {u_2}^2, \sqrt{2}\, u_1 u_2) \cdot ({v_1}^2, {v_2}^2, \sqrt{2}\, v_1 v_2)$

$= \emptyset(u) \cdot \emptyset(v)$

In this case a linear function can represent a quadratic function as the dot product of different feature vectors

NOTE: A linear kernel cannot represent a quadratic boundary (discriminator). However, a polynomial kernel can.

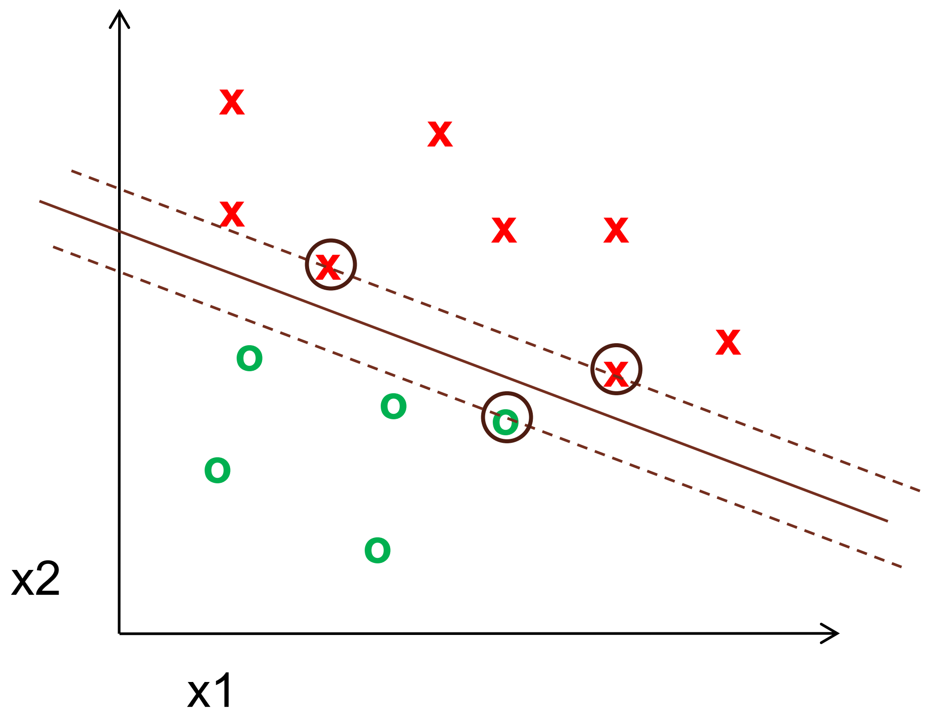
Classifiers: Linear SVM



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

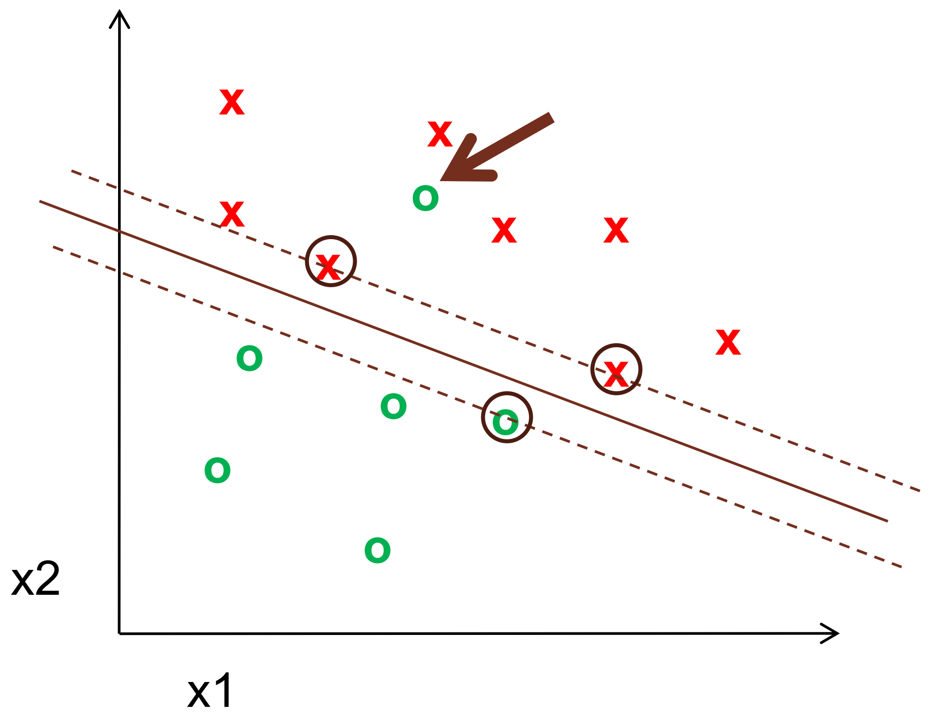
Classifiers: Linear SVM



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

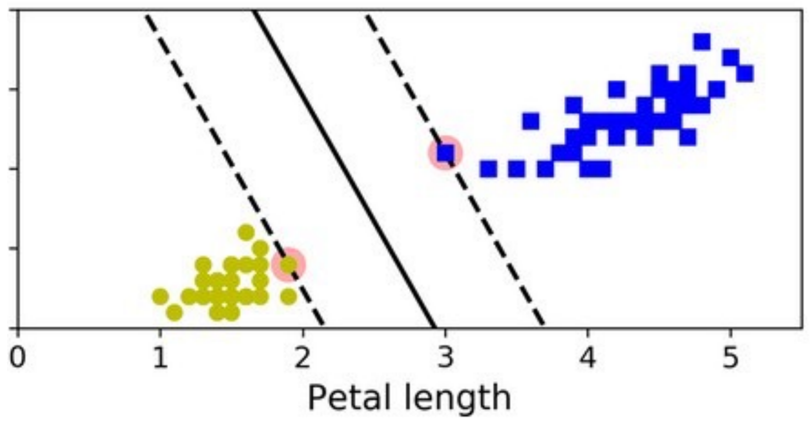
Classifiers: Linear SVM



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

Linear SVM Classification



- Solid line represents the decision boundary of an SVM classifier
  - Separates the two classes and also stays as far away from the closest training instances as possible
- Think of an SVM classifier as fitting the widest possible street (represented by the parallel dashed lines) between the classes
  - Called large margin classification since its main goal it to maximize the margin between the two classes (the margin is the space between the decision boundary and the nearest data points from each class)
  - Support Vectors highlighted in figure (yellow and blue support vectors)

Decision Function as a 2D Plane

Equation 5-2. Linear SVM classifier prediction

$$\hat{y} = \begin{cases} 0 & \text{if } \mathbf{w}^\top \mathbf{x} + b < 0, \\ 1 & \text{if } \mathbf{w}^\top \mathbf{x} + b \geq 0 \end{cases}$$

- The decision boundary is the set of points where the decision function is equal to 0 (thick solid line)
- The dashed lines represent the points where the decision function is equal to 1 or –1 around decision boundary and form a margin around it.
- Training a linear SVM classifier means finding the values of **w** and *b* that make this margin as wide as possible while avoiding margin violations (hard margin) or limiting them (soft margin).

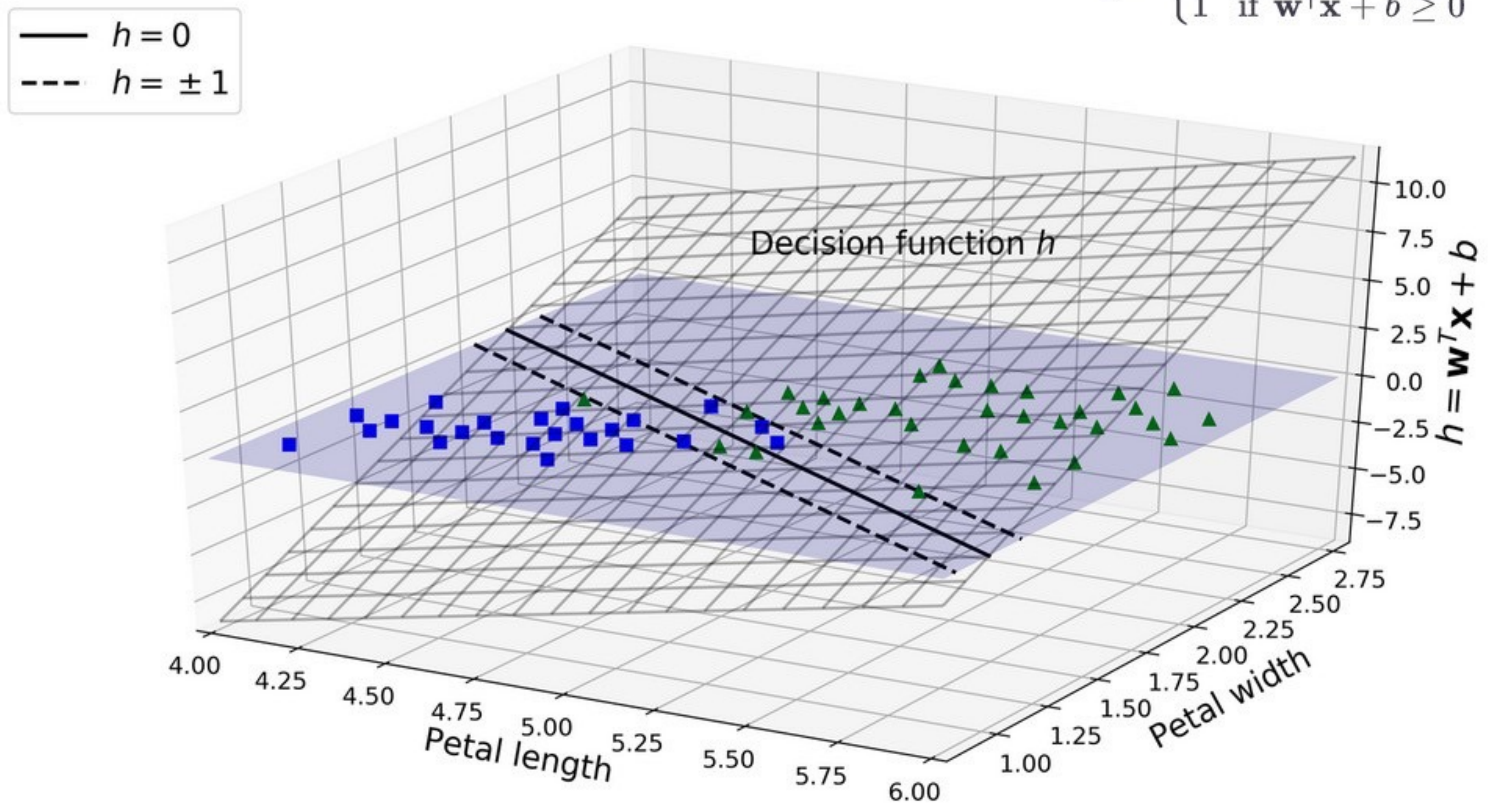


Figure 5-12. Decision function for the iris dataset

Nonlinear SVM Classification

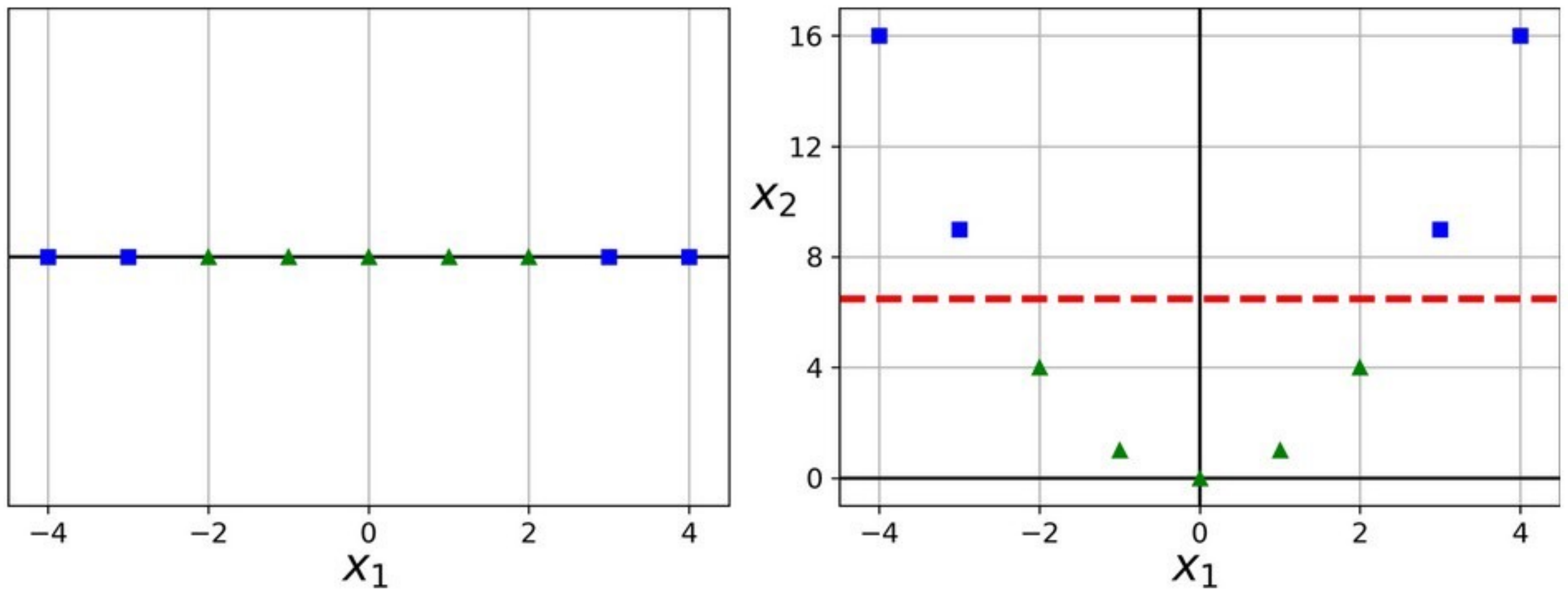
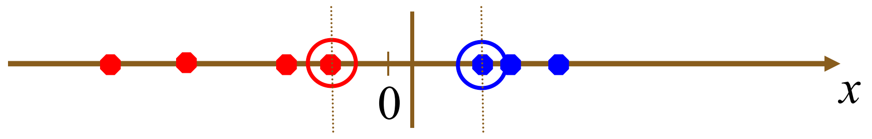


Figure 5-5. Adding features to make a dataset linearly separable

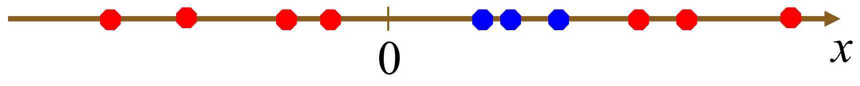
- Many datasets are not even close to being linearly separable
- In some cases, adding features (e.g. polynomial) will result in a linearly separable dataset
- For example, in Figure 5-5, a second feature $x_2 = (x_1)^2$ was added, and the resulting 2D dataset is perfectly linearly separable

Nonlinear SVMs

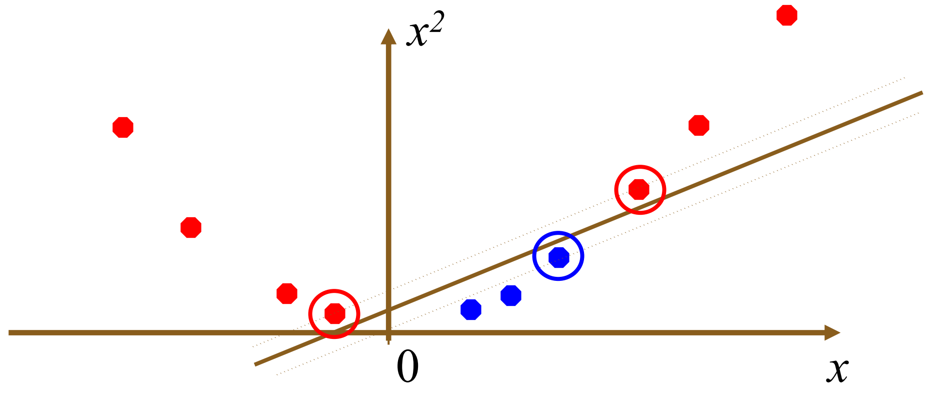
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?



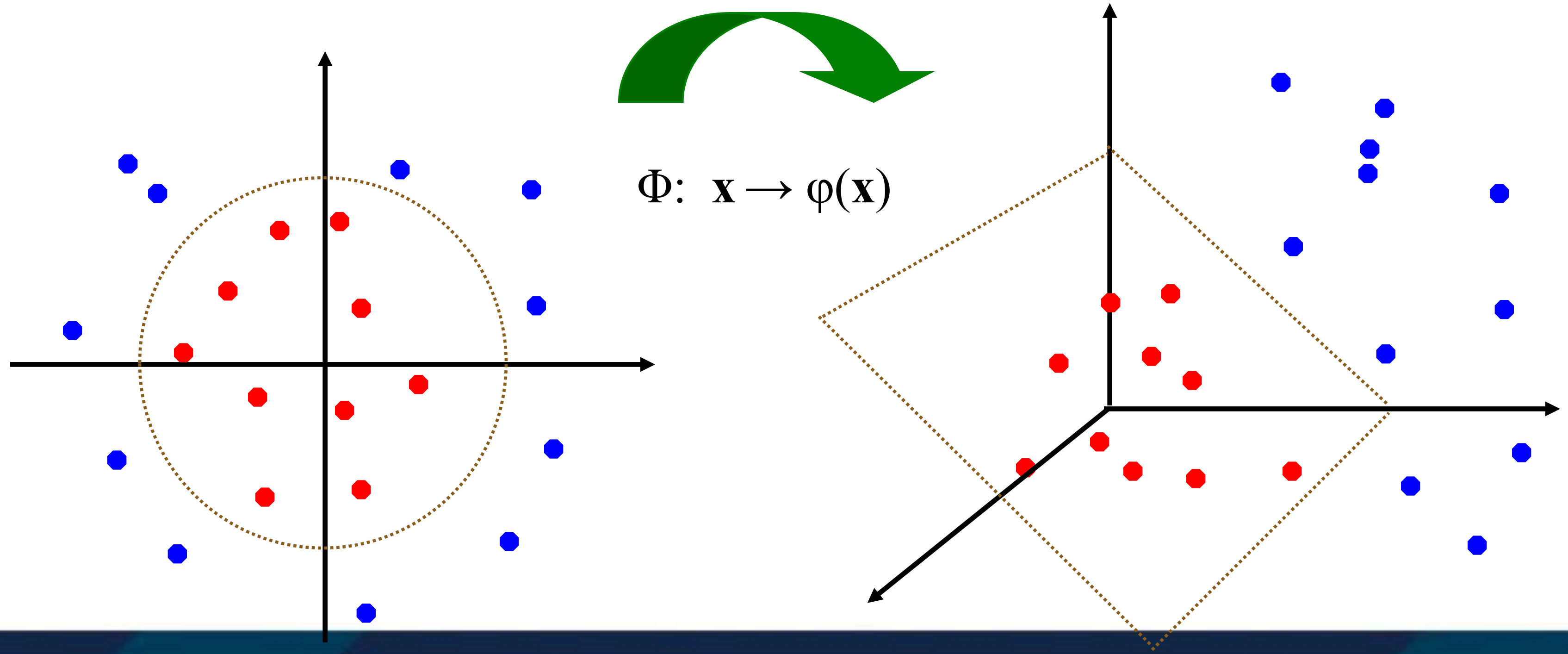
- We can map it to a higher-dimensional space:



Based on Andrew Moore

Nonlinear SVMs

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



Based on Andrew Moore

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Nonlinear SVMs

- ***The kernel trick*:** instead of explicitly computing the lifting transformation $\boldsymbol{\varphi}(\mathbf{x})$, define a kernel function K such that

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

(to be valid, the kernel function must satisfy *Mercer's condition*)

- This gives a nonlinear decision boundary in the original feature space:

$$\sum_i \alpha_i y_i \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}) + b = \sum_i \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

Mercer's condition ensures that the **inner product** in the feature space induced by the kernel function $K(x,y)$ is always **non-negative**, which is a necessary condition for the associated kernel matrix to be **positive semi-definite**. Positive semi-definiteness of the kernel matrix is crucial for the stability and convergence of algorithms that rely on kernel methods, such as SVMs.

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Kernel Trick

- In Machine Learning, a *kernel* is a function capable of computing the dot product $\phi(\mathbf{a})^\intercal \phi(\mathbf{b})$, based only on the original vectors **a** and **b**, without having to compute (or even to know about) the transformation ϕ.
- K(x, y) = (x • y + r)^d
- By using kernel functions, we can compute the dot products without explicitly expanding the feature space. We can effectively capture complex relationships as if we had added numerous polynomial features, including high-degree polynomials, without the computational and storage costs associated with adding them explicitly.

Importance of the Kernel Trick

- Adding polynomial features is simple to implement
- There is a tradeoff:
  - Using a low polynomial degree may not be able to effectively separate very complex datasets
  - As the polynomial degree increases, a huge number of features are created which results in slower computational performance
- The kernel trick gets the same result as if you had added many polynomial features, even with very high-degree polynomials, without actually having to add them
- d is the degree of the polynomial/complexity of decision boundary, and r is the constant term added to the dot product of the data points, which can shift the kernel function away from the origin.

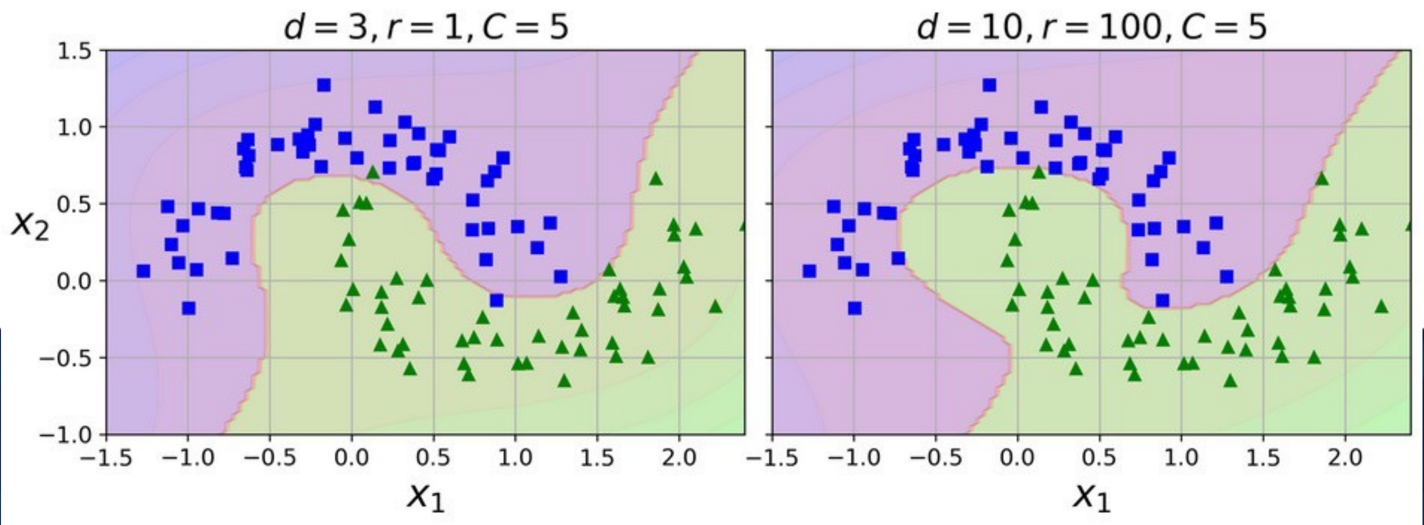
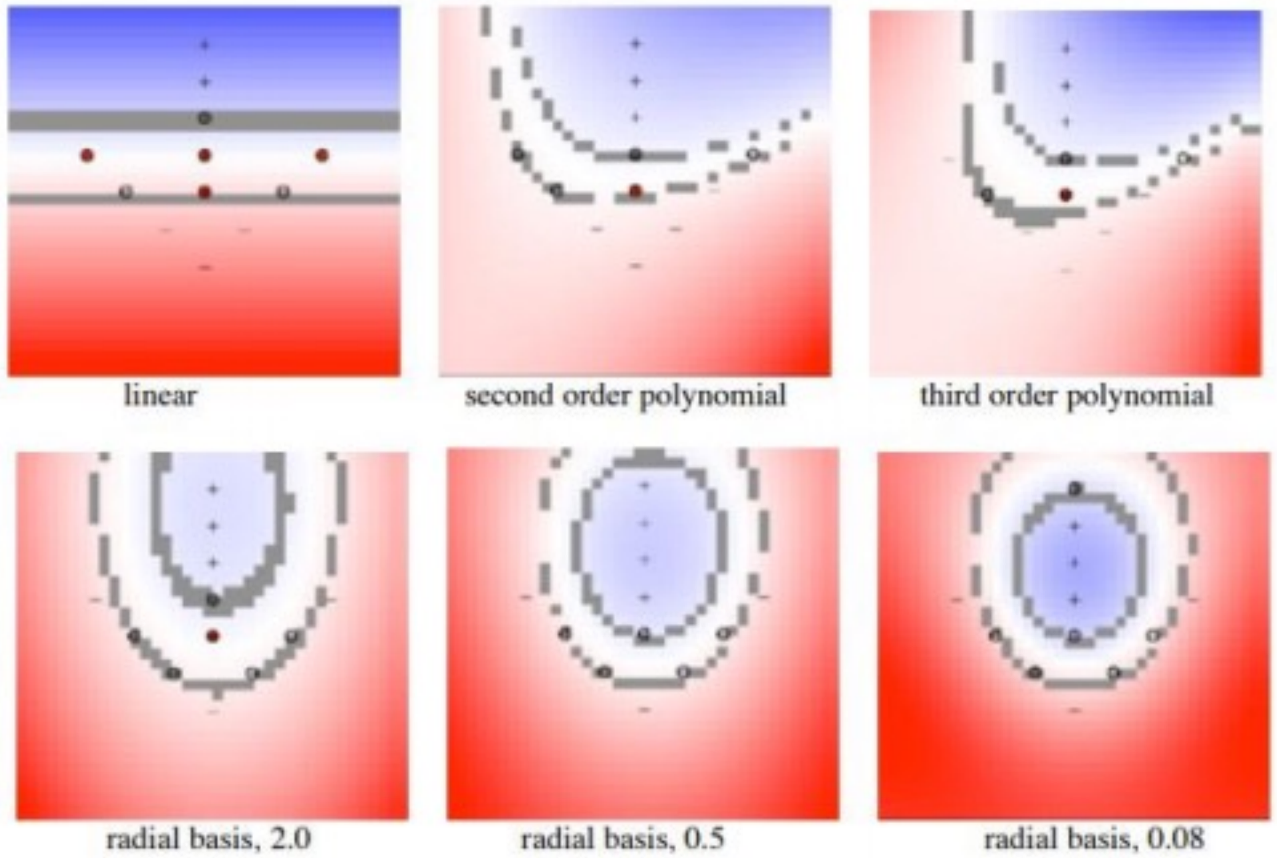
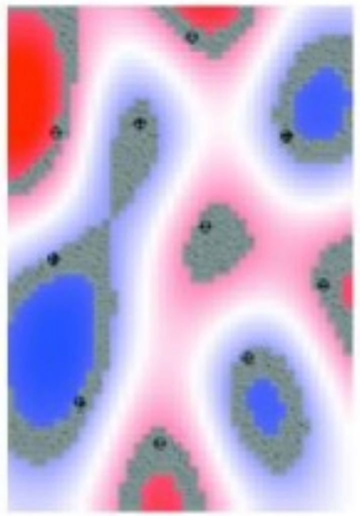


Figure 5-7. SVM classifiers with a polynomial kernel

Kernel Examples



SVM that has overfit training data



Learning SVMs → How do We …

- Choose the Kernel? It is a magical transformation!
- Choose the Examples? Side effect of choosing the weights
- Choose the weights? Maximize the Margin

Based on Domingos

How to Choose

- With so many kernels to choose from, how can you decide which one to use?
- As a rule of thumb, you should always try the linear kernel first
- If the training set is not too large, you should also try the Gaussian Radial Basis Function (RBF) kernel; it works well in most cases.
- There may be specialized kernels for your training set's data structure.
- You can experiment with a few other kernels, using cross-validation and grid search.

SVM Regression

- To use SVMs for regression instead of classification, reverse the objective:
  - instead of trying to fit the largest possible street between two classes while limiting margin violations, SVM Regression tries to fit as many instances as possible on the street while limiting margin violations (i.e., instances off the street).
- The width of the street is controlled by a hyperparameter (ε).
- A larger ε may lead to a bias in favor of simpler models with larger margins but may result in more prediction errors. A smaller ε may increase the risk of overfitting to noise in the data.

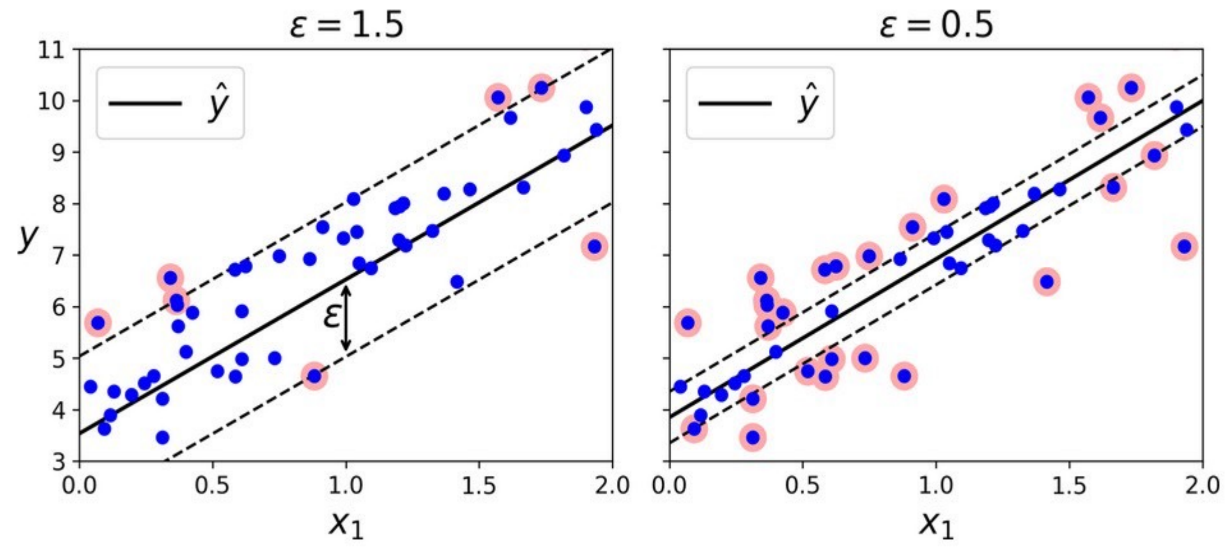


Figure 5-10. SVM Regression

Kernelized SVM Model

- For nonlinear regression tasks

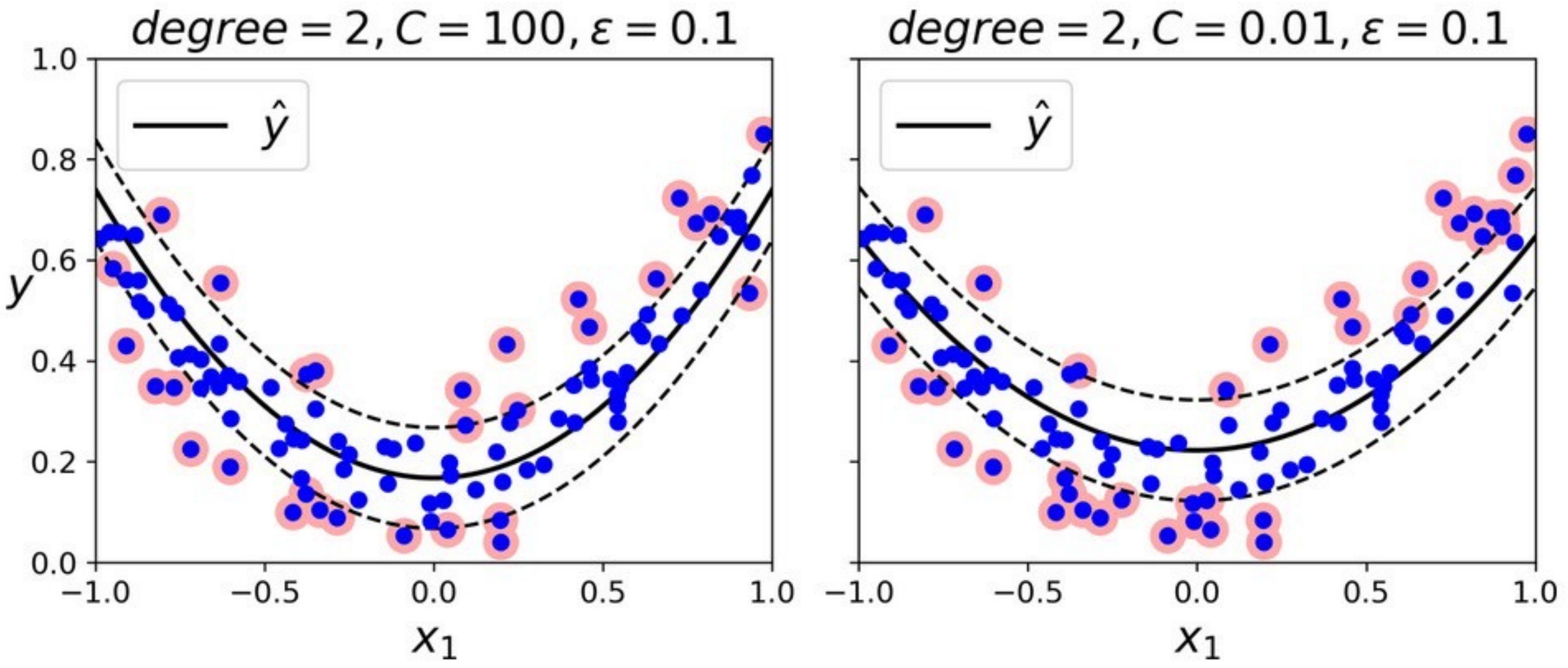
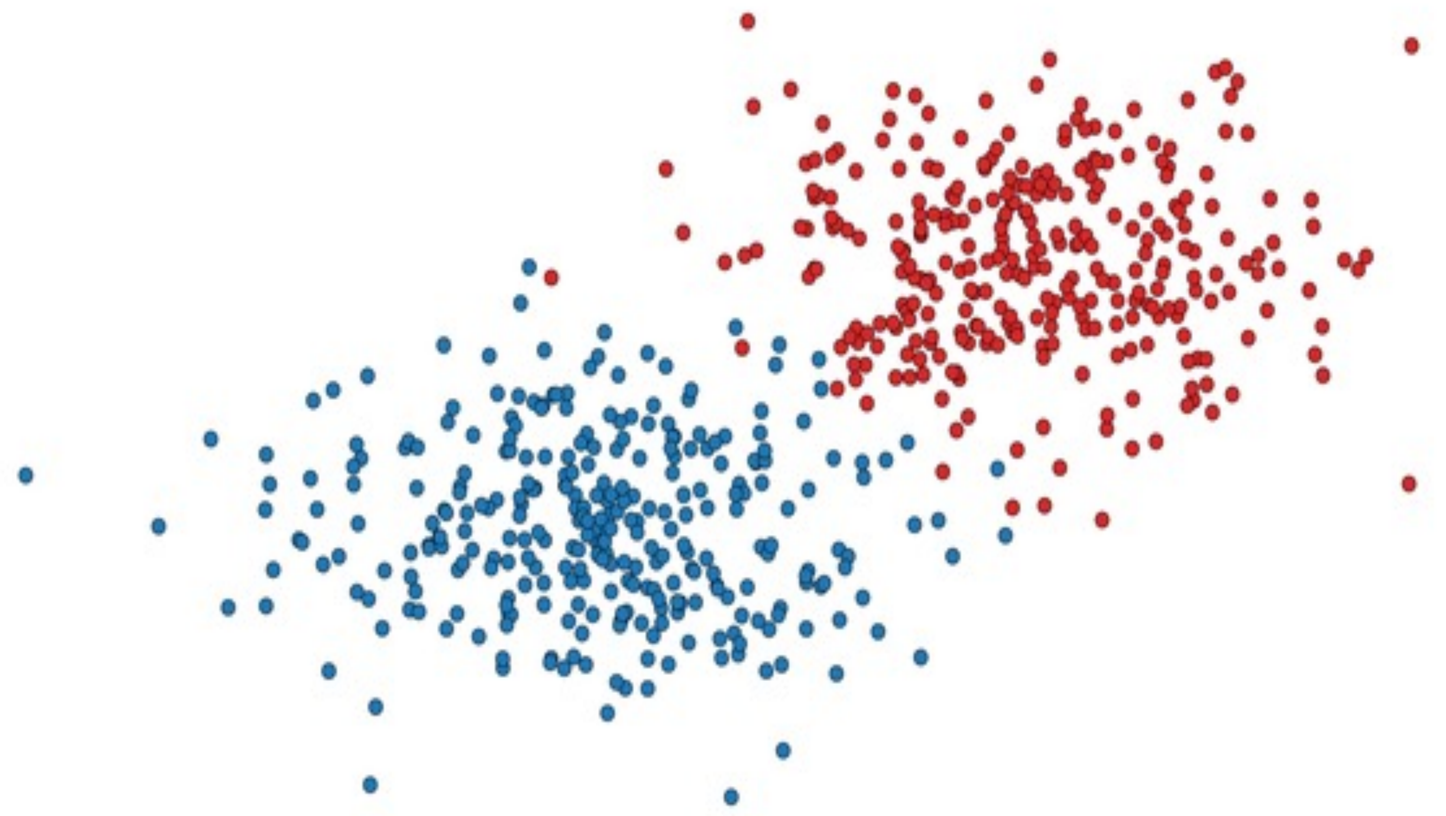


Figure 5-11. SVM Regression using a second-degree polynomial kernel

Ambiguous Sample Separation Example

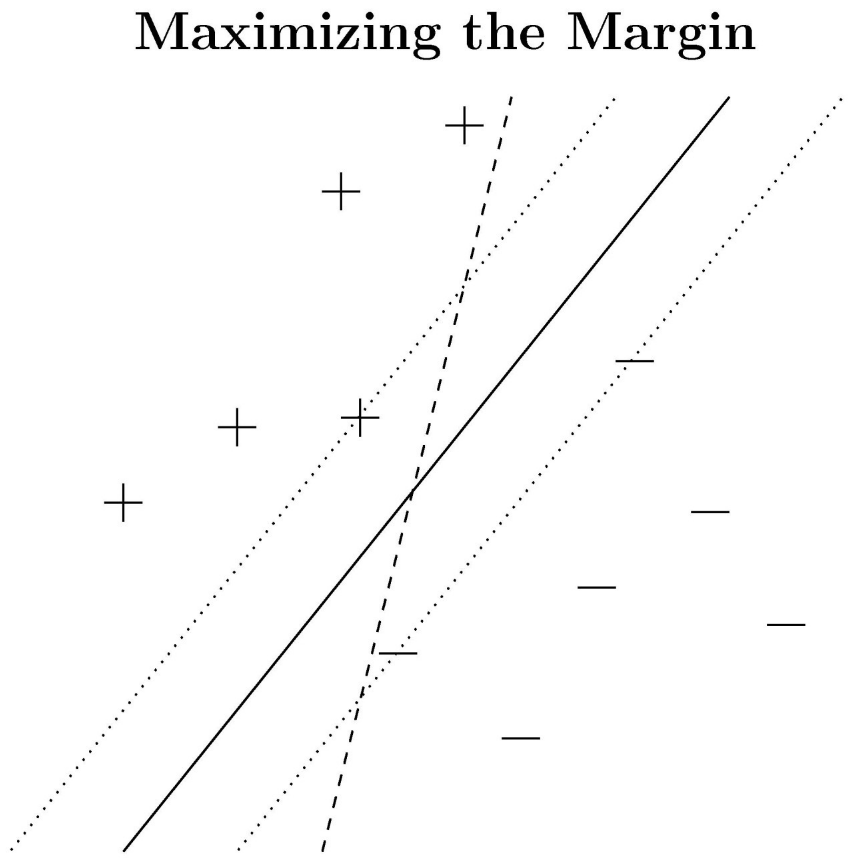
- In the figure here we see that blue and red points are spatially separated, but there is a small overlapping region.
- Drawing a line somewhere between the two blobs wouldn't be very objective and each time you try it would be a different boundary.



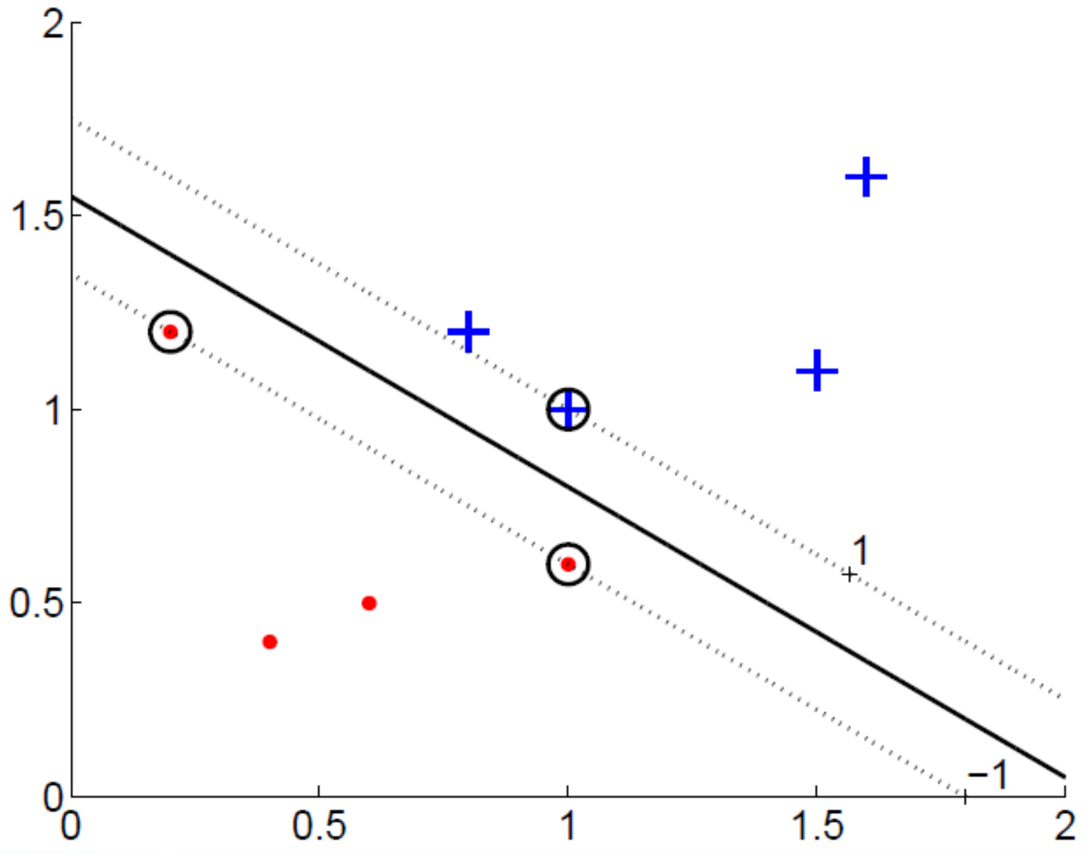
Based on Dariusz Kajtoch

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Maximizing the Margin

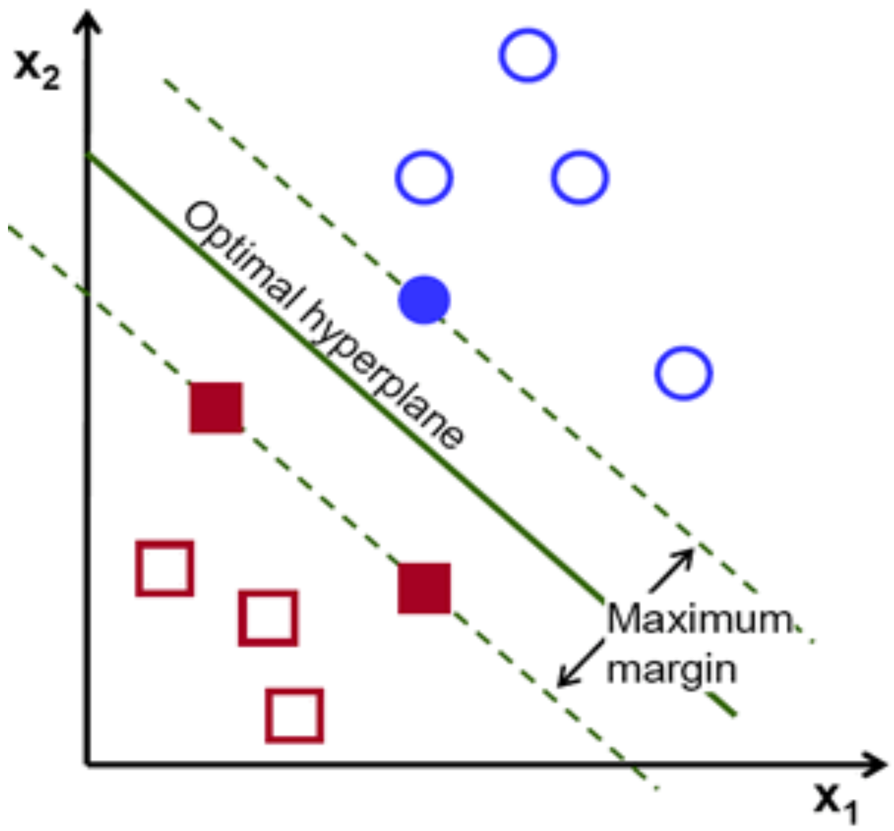


Margin (When Data is Nicely Separated)



Hard Margins

- It would be nice if problems in the real world always worked out this way.
- However, with noisy data and/or errors in data this will not work very well.



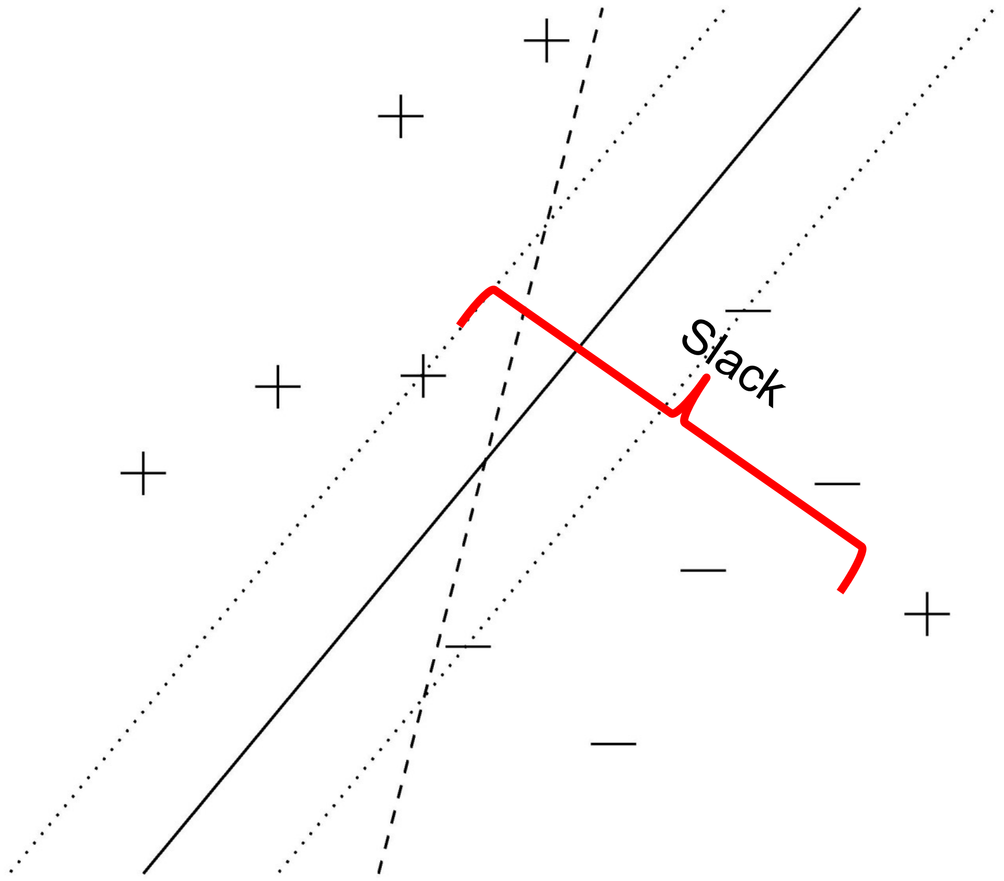
Based on Dariusz Kajtoch

Handling Noisy Data

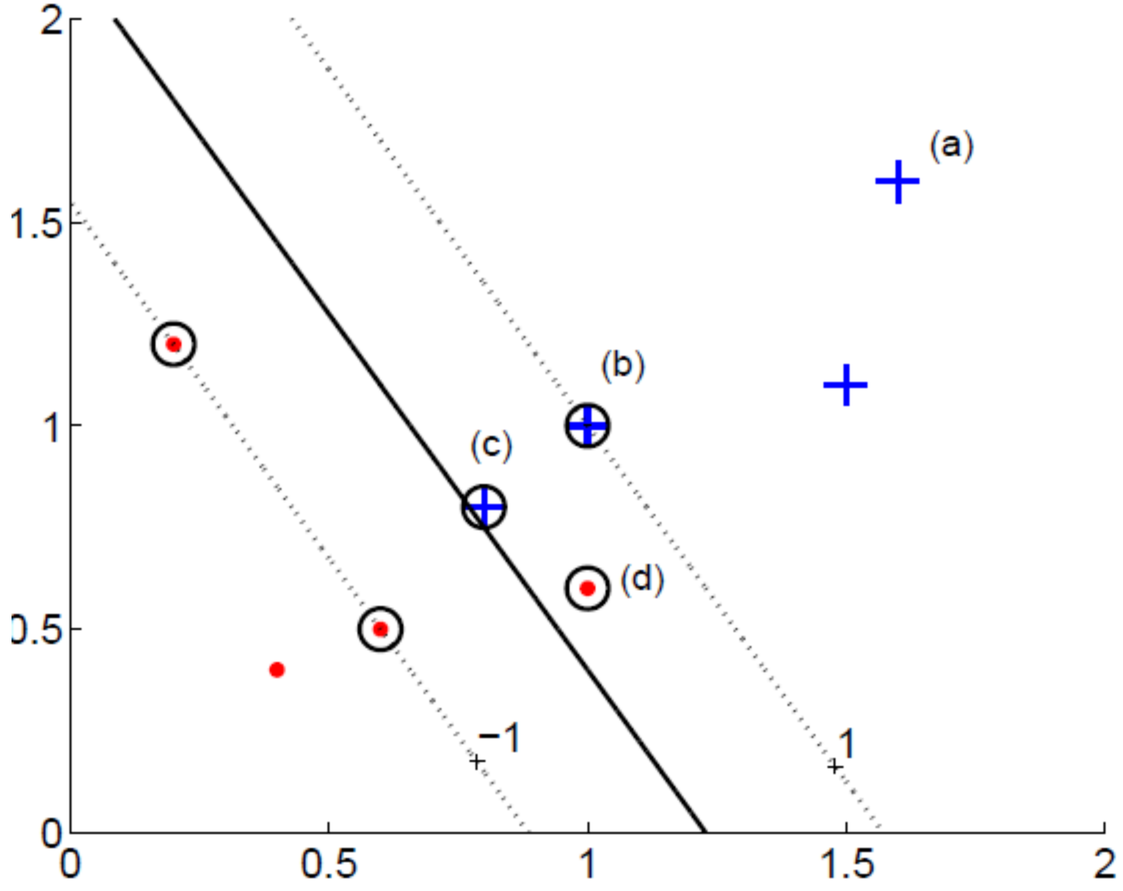
Introduce Slack Varibles: ξ_i

Minimize: $w \cdot w + C \sum_i \xi_i$

Subject to: $y_i(w \cdot x_i) \geq 1 - \xi_i$ for all i



Noisy Example (More Line Real Life Data)



Hard Margin Classification

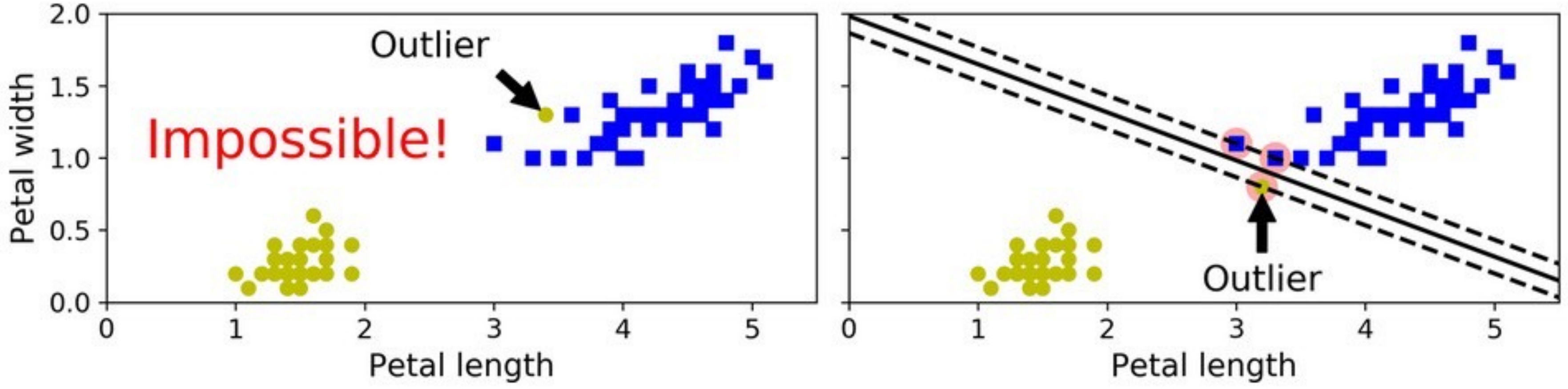


Figure 5-3. Hard margin sensitivity to outliers

- Strictly impose that all instances must be off the street and on the right side – does not allow for any misclassifications to occur
- There are two main issues with hard margin classification:
  - Data needs to be separable by a line
  - Sensitive to outliers

Soft Margin Classification

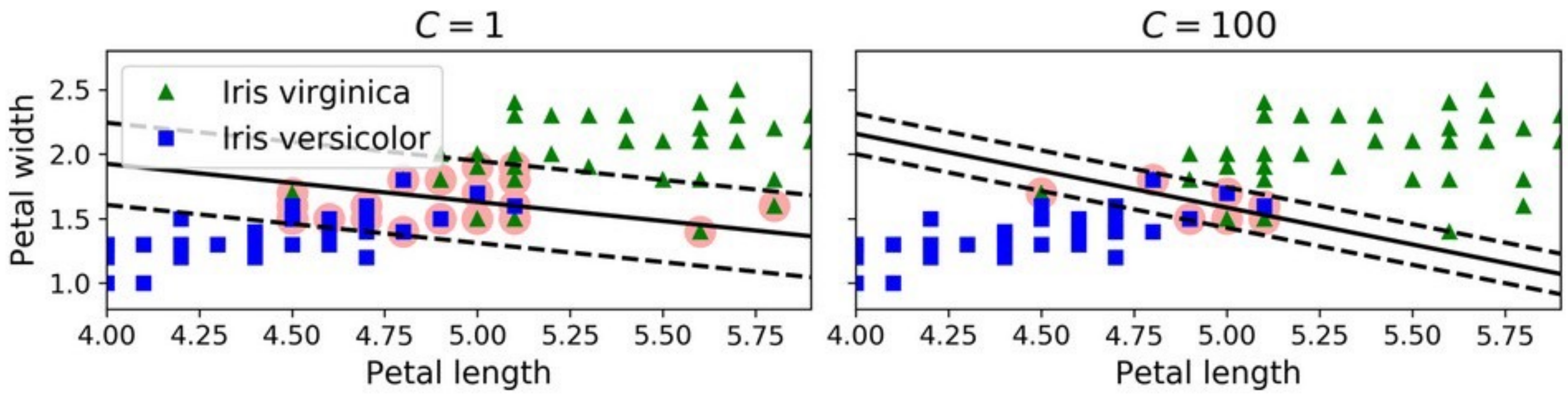


Figure 5-4. Large margin (left) versus fewer margin violations (right)

- More flexible model than hard margin classification
- Balance between keeping the street as large as possible and limiting the number of instances that end up in the middle of the street or even on the wrong side
- If your SVM model is overfitting, you can try regularizing it by making street wider

The Dual Problem

- Primal problem - constrained optimization problem
- Dual problem is a different but closely related problem
- The solution to the dual problem typically gives a lower bound to the solution of the primal problem, but under some conditions it can have the same solution as the primal problem.
- The SVM problem happens to meet these conditions so you can choose to solve the primal problem or the dual problem; both will have the same solution.
- The dual problem is faster to solve than the primal one when the number of training instances is smaller than the number of features.
- More importantly, the dual problem makes the kernel trick possible, while the primal does not.

Soft Margin Hyperplane

- Not linearly separable

$$r^t\left(\mathbf{w}^T x^t + w_0\right) \geq 1 - \xi^t \longleftarrow$$

Relaxes the Original Constraint

- Soft error

$$\sum_t \xi^t$$

- New primal is

  "The Dual"

The Lagrangian

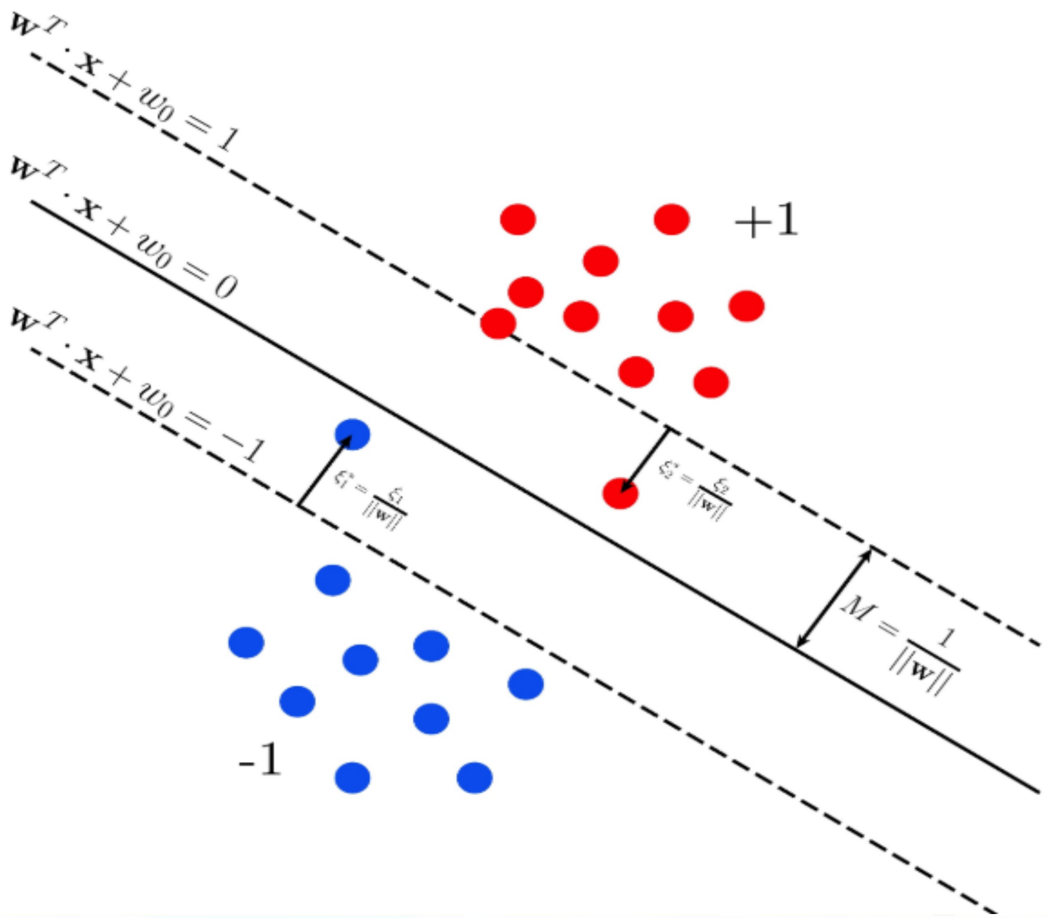
$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 + \underbrace{C\sum_t \xi^t}_{\text{Penalty}} - \underbrace{\sum_t \alpha^t[r^t(\mathbf{w}^T x^t + w_0) - 1 + \xi^t] - \sum_t \mu^t \xi^t}_{\text{Constraints}}$$

Slack Variable

- Introduce a slack variable to get the soft margin objective

- Two conflicting objectives:
  - Make the slack variables as small as possible to reduce the margin violations.
  - Increase the margin.
  - This is where the C hyperparameter comes in: it allows us to define the tradeoff between these two objectives.
- Large C, the focus is on having a low training error (smaller margin) - Overfitting
- Small C, the focus is on having a low generalization error (larger margin) - Underfitting

Illustration of Slack Variables



$\mathbf{w}^T \cdot \mathbf{x} + w_0 = 1$
$\mathbf{w}^T \cdot \mathbf{x} + w_0 = 0$
$\mathbf{w}^T \cdot \mathbf{x} + w_0 = -1$
+1
-1
$\xi_1^* = \frac{\xi_1}{||\mathbf{w}||}$
$\xi_2^* = \frac{\xi_2}{||\mathbf{w}||}$
$M = \frac{1}{||\mathbf{w}||}$

The slack variable measures the distance of a point to its marginal hyperplane

Original Optimization Problem:

$$\min_{\mathbf{w},w_0} \left\{ \frac{1}{2} ||\mathbf{w}||^2 \right\}$$

New Optimization Problem:

$$\min_{\mathbf{w},w_0,\xi} \left\{ \sum_{i=1}^{N} \xi_i \right\}$$

Based on Dariusz Kajtoch

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Solving for Slack

Slack variables are positive (or zero), local quantities that relax the stiff condition of linear separability, where each training point is seeing the same marginal hyperplane. Now, **each individual** training point can see a different, but parallel hyperplane:

$$\begin{cases} \mathbf{w}^T \cdot \mathbf{x}_i + w_0 \geqslant 1 - \xi_i, & \text{if } y_i = +1 \\ \mathbf{w}^T \cdot \mathbf{x}_i + w_0 \leqslant -1 + \xi_i, & \text{if } y_i = -1 \end{cases}$$

which can be cast into a single equation as we did in the hard margin case:

$$y_i(\mathbf{w}^T \cdot \mathbf{x} + w_0) \geqslant 1 - \xi_i.$$

The right hand side is also called the *Hinge loss* function. The next step is to find a hyperplane that minimizes the collective Hinge loss (discuss later):

$$\min_{\mathbf{w}, w_0} \left\{ \sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}^T \cdot \mathbf{x}_i + w_0)) \right\}$$

We can now consider a combined effect. We would like to maximize the margin and allow for total error of some magnitude. The Optimization problem takes the form:

$$\min_{\mathbf{w}, w_0, \xi} \left\{ \tfrac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{N} \xi_i \right\},$$

subject to constraints,

$$y_i(\mathbf{w}^T \cdot \mathbf{x} + w_0) \geqslant 1 - \xi_i, \ \xi_i \geqslant 0$$

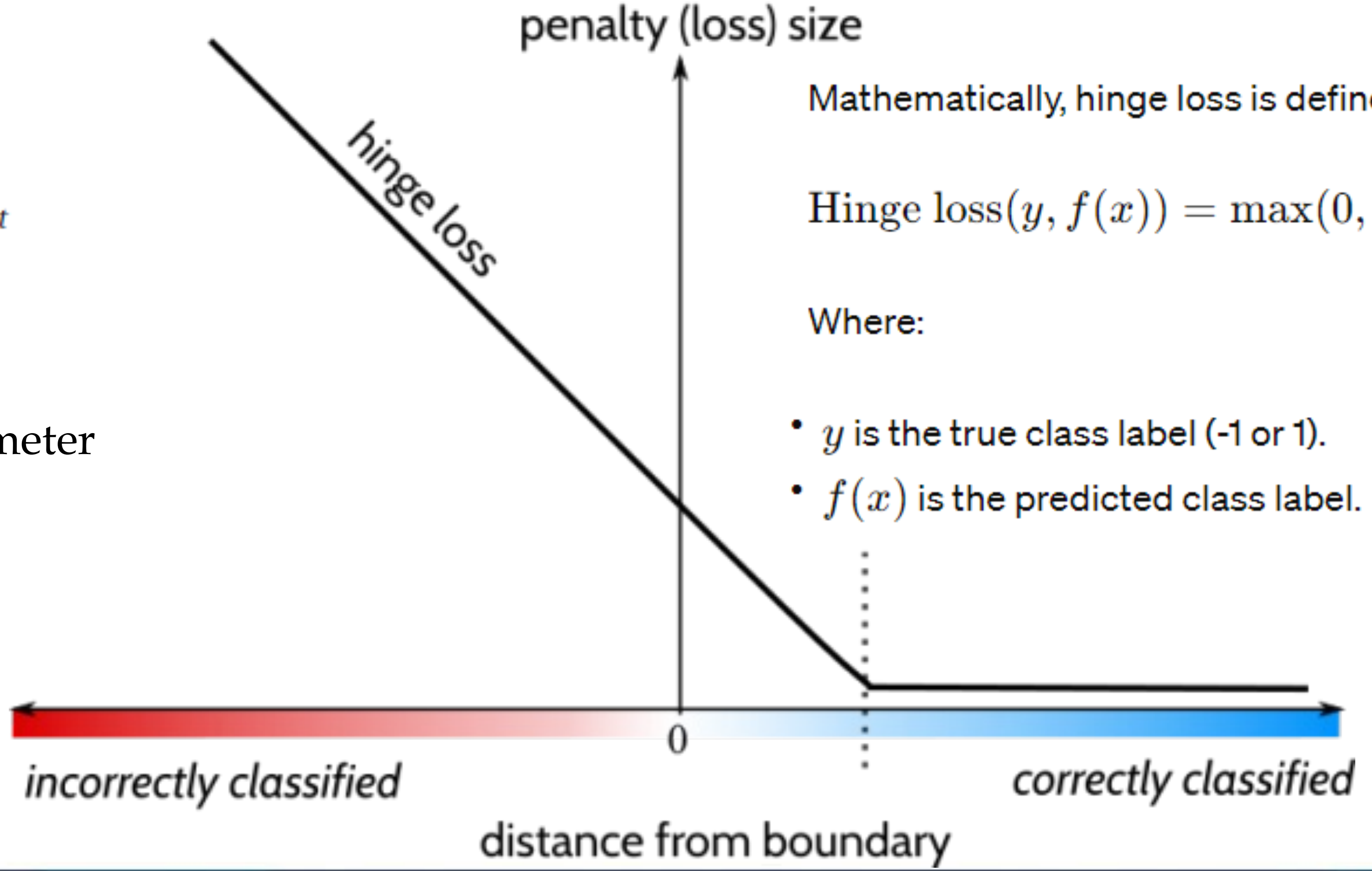
The variable C is the penalty strength, which specifies how much do we care about errors (training points that are on the wrong side).

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Hinge Loss

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_t \xi^t$$

Regularization parameter



Mathematically, hinge loss is defined as:

$$\text{Hinge loss}(y, f(x)) = \max(0, 1 - y \cdot f(x))$$

Where:

- y is the true class label (-1 or 1).
- $f(x)$ is the predicted class label.

Image credit: https://math.stackexchange.com/questions/782586/how-do-you-minimize-hinge-loss

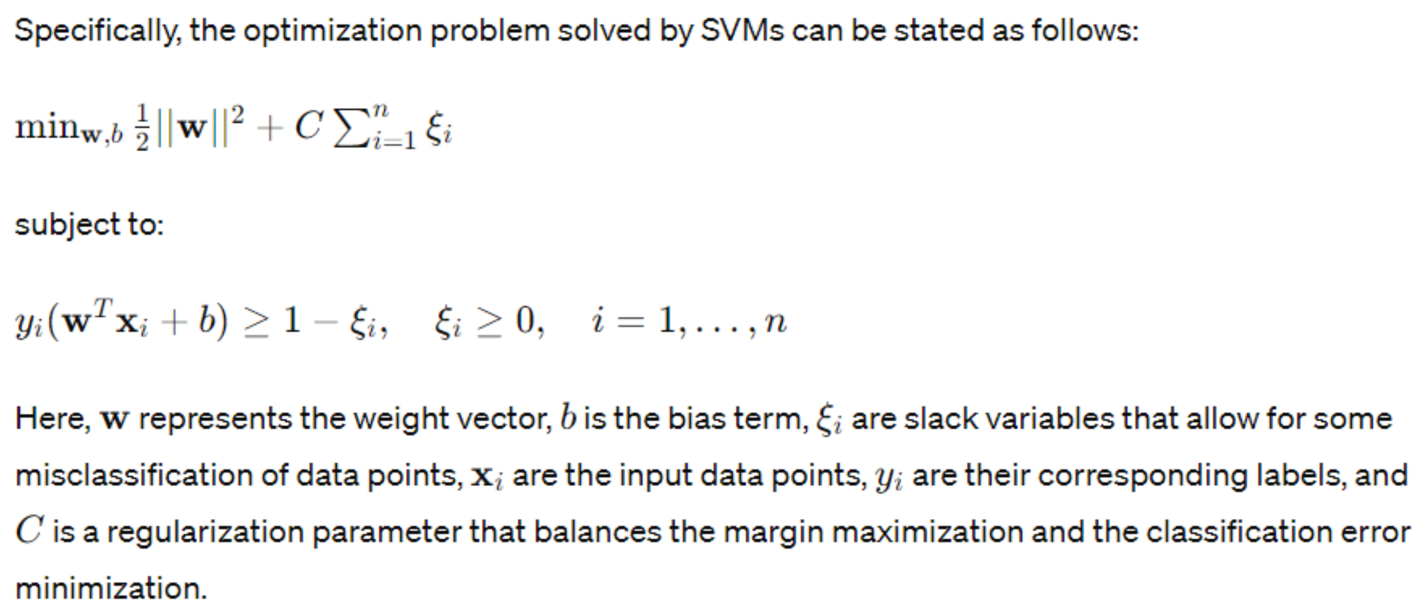
Beyond SVM: Applications of the Kernel Trick

- Take the perceptron
- Replace the dot product with an arbitrary similarity function
- Now you have a much more powerful learner
- Kernel Matrix K(x,x') for x,x' $\in Data$
- K(x,x') = $\emptyset(x) \cdot \emptyset(x^\wedge{\prime})$ is still a dot product
- Also guarantees convex weight optimization
- This is a Very general trick, and not just for SVMs:
- **Examples:**
  - Extreme Learning Machines
  - Kernel Ridge Regression (KRR)
  - Kernel Principal Component Analysis (Kernel PCA)
  - Kernel Logistic Gaussian Processes (GP)

Based on Domingos

SVMs are ”Kernel Machines”

- Discriminant-based: No need to estimate densities first
- Define the discriminant in terms of support vectors
- The use of kernel functions, application-specific measures of similarity
- No need to represent instances as vectors
- Convex optimization problems with a unique solution



Specifically, the optimization problem solved by SVMs can be stated as follows:

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

subject to:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n$$

Here, $\mathbf{w}$ represents the weight vector, b is the bias term, ξ_i are slack variables that allow for some misclassification of data points, $\mathbf{x}_i$ are the input data points, y_i are their corresponding labels, and C is a regularization parameter that balances the margin maximization and the classification error minimization.

https://www.youtube.com/watch?v=qF0aDJfEa4Y
https://michielstock.github.io/ConvexSummary/

Practical Considerations for SVMs

- Use kernel functions wisely
  - Experiment with different kernels for your problem
- Choose the right C value
  - Optimal values of C depends on your problem
  - Can use grid search
- Use cross-validation to evaluate performance
- Be aware of class imbalance
  - SVMs struggle with imbalanced datasets
    - Some techniques to counter this are oversampling the minority class or using weighted SVMs
- Add Regularization
  - Prevent overfitting and improve generalization performance of the model

https://medium.com/@mun.articles/svms-in-practice-applications-and-use-cases-for-machine-learnings-most-effective-model-2ae25f4207ef

SVM Summary

1. Discriminant Based Method
2. After training, the parameter of the linear model, the weight vector, can be written down in terms of a subset of the training set, which are the support vectors.
3. The output is written as a sum of the influences of support vectors and given by a *kernel function*
4. Kernel functions allow us to calculate shared paths such as $K(G1,G2)$ without needing to represent $G1$ or $G2$ explicitly as vectors.
5. Kernel-based algorithms are formulated as convex optimization problems, and there is a single optimum that we can solve for analytically.

- ***There are many types of SVMs but no matter what type, we have a way to maximize the separability, or margin, of instances subject to a constraint of the smoothness of solution.***
- ***Solving for it, we get the support vectors. The kernel function defines the space according to its notion of similarity and a kernel function is good if we have better separation in its corresponding space.***

THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Homework Overview

This Week

- Reading:
  - Chapter 5 in the textbook
- HW #3 (Run/Write Python Script in Google Colab first, and then answer the homework questions)
- Discussion #3

- Reminder: No extensions provided. Start assignments early!

Next Steps

- Come to office hours with any questions you may have.
- Work on your HW and Discussion and submit them by 9:00 am ET on Saturday.
- See you next class!

Thank you!