slides originally by
Dr. Richard Burns,
modified by
Dr. Stephanie Schwartz

SUPPORT VECTOR MACHINES

CSCI 450: AI

Support Vector Machines (SVM)

- What are they?
  - Developed in the 1990s
  - Computer Science community
  - Very popular
- Performance:
  - Often considered one of the best “out of the box” classifiers
  - *Applications:* handwritten digit recognition, text categorization

Support Vector Machines (SVM)

- Comparing to other statistical learning methods:
  - SVMs work well with high-dimensional data

- Unique:
  - Represents "decision boundary" using a *subset of training examples*

Terminology

1. Maximal Margin Classifier
2. Support Vector Classifier
3. Support Vector Machine

- Often all three are referred to as “Support Vector Machine”

The Path Ahead

1. Maximal Margin Classifier
2. Support Vector Classifier
   - Generalization of Maximal Margin Classifier
3. Support Vector Machine
   - Generalization of Support Vector Classifier

Maximal Margin Classifier

- First, need to define a hyperplane

- What is a hyperplane?
  - Hyperplane has *p-1* dimensions in a *p* dimensional space
  - *Example:* in 2 dimension space, a hyperplane has 1 dimension (and thus, is a line)

Hyperplane Mathematical Definition

- For two dimensions, hyperplane defined as:

$$B_0 + B_1 X_1 + B_2 X_2 = 0$$

B_0, B_1, B_2 are parameters.
X_1, X_2 are variables.

- Note that this equation is a line:
  - Hyperplane is in one-dimension

$$B_0 + B_1 X_1 + B_2 Y = 0$$

$$B_2 Y = -B_1 X_1 - B_0$$

$$Y = \frac{-B_1 X_1 - B_0}{B_2}$$

Hyperplane Mathematical Definition

$$B_0 + B_1 X_1 + B_2 X_2 = 0$$

- We're going to "find" values for B_0, B_1, B_2.
- Then, for any values X_1 and X_2:
  1. if $B_0 + B_1 X_1 + B_2 X_2 = 0$
     - Point is on the line.

Hyperplane Mathematical Definition

$$B_0 + B_1 X_1 + B_2 X_2 = 0$$

- We're going to "find" values for B_0, B_1, B_2.
- Then, for any values X_1 and X_2:

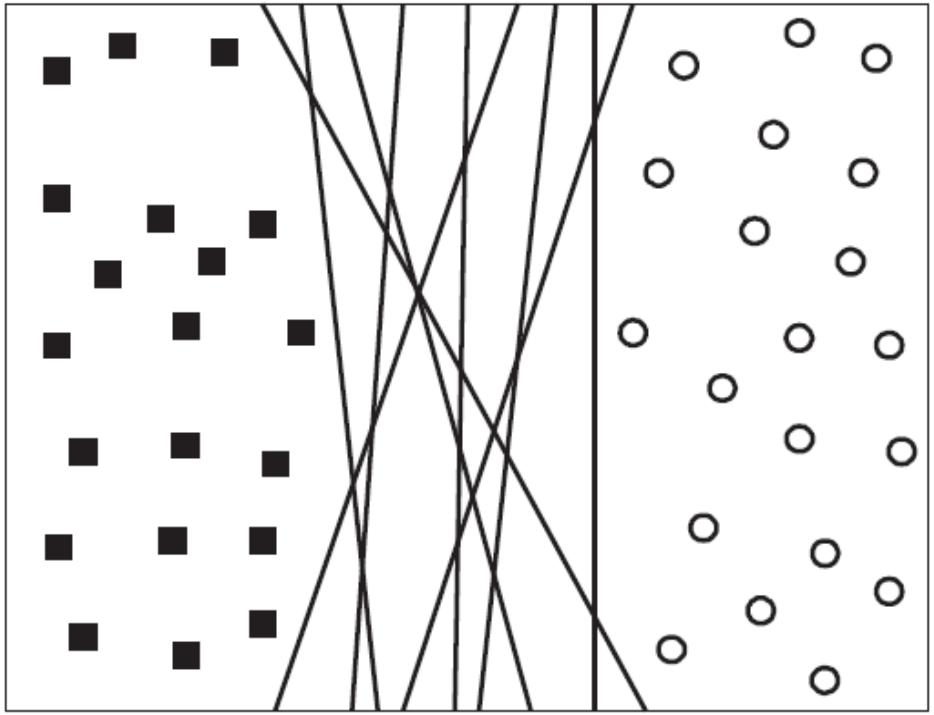
2. if $B_0 + B_1 X_1 + B_2 X_2 > 0$
   - Point is not on the line. On one side of the line.
3. if $B_0 + B_1 X_1 + B_2 X_2 < 0$
   - Point is on the other side of the line.

Hyperplane

- … is dividing 2-dimensional space into two halves by a line.

Separating Hyperplane

Note: a separating hyperplane means zero training errors.



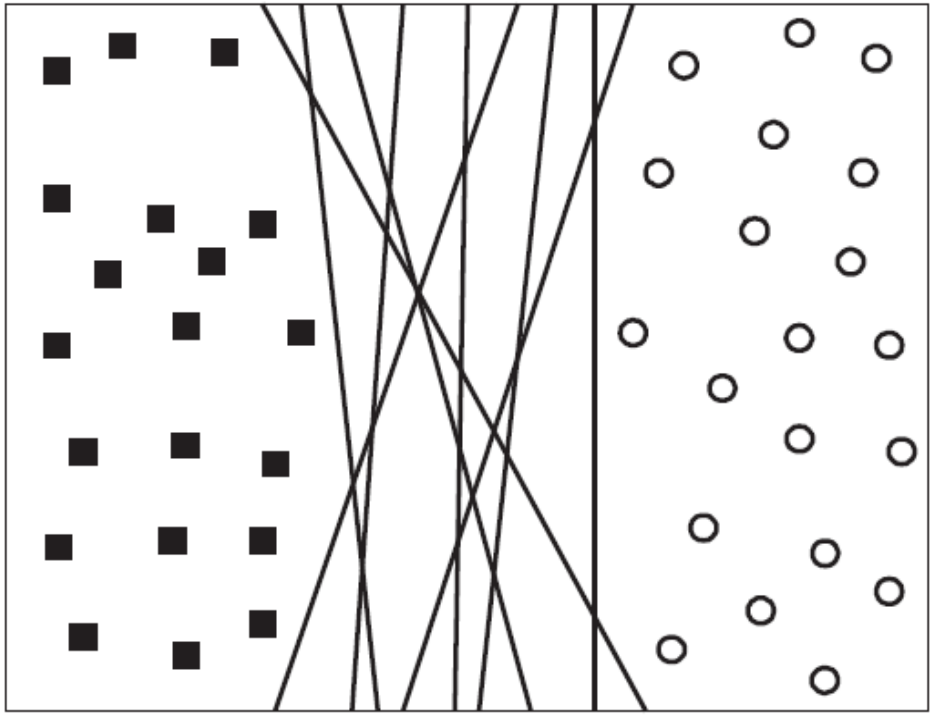
Dataset with two classes:

1. Squares
2. circles

Can find a separating hyperplane with all squares on one side and all circles on the other.

Infinitely many such hyperplanes possible.

Classification Using a Separating Hyperplane



- For a new test instance, which side of the line is it on?

- $B_0 + B_1X_1 + B_2X_2 > 0$
- $B_0 + B_1X_1 + B_2X_2 < 0$

Classification Using a Separating Hyperplane

- Standard SVM approach:
  - Label class data as either +1 or -1, depending on which class an instance belongs to.
  - Prediction:

$$y_i = \begin{cases} 1, & \text{if } B_0 + B_1 x_1 + B_2 x_2 + \ldots + B_n x_n > 0 \\ -1, & \text{if } B_0 + B_1 x_1 + B_2 x_2 + \ldots + B_n x_n < 0 \end{cases}$$

Classification Using a Separating Hyperplane

- For a new test instance, which side of the line is it on?

- $B_0 + B_1X_1 + B_2X_2 > 0$
- $B_0 + B_1X_1 + B_2X_2 < 0$

- Can also look at the *magnitude*.
  - How far from zero?
  - Greater magnitude means more confident prediction.

Some Concerns with this Approach:

- Datasets with more than 2 target classes
- What if a “separating hyperplane” can’t be formed
- Data is more than two dimensions
- Regression instead of classification

SVMs can deal with each of these.

What if Data is more than 2-Dimensions?

- Mathematical definition of hyperplane generalizes to *n*-dimensions:

$$B_0 + B_1 X_1 + B_2 X_2 = 0$$

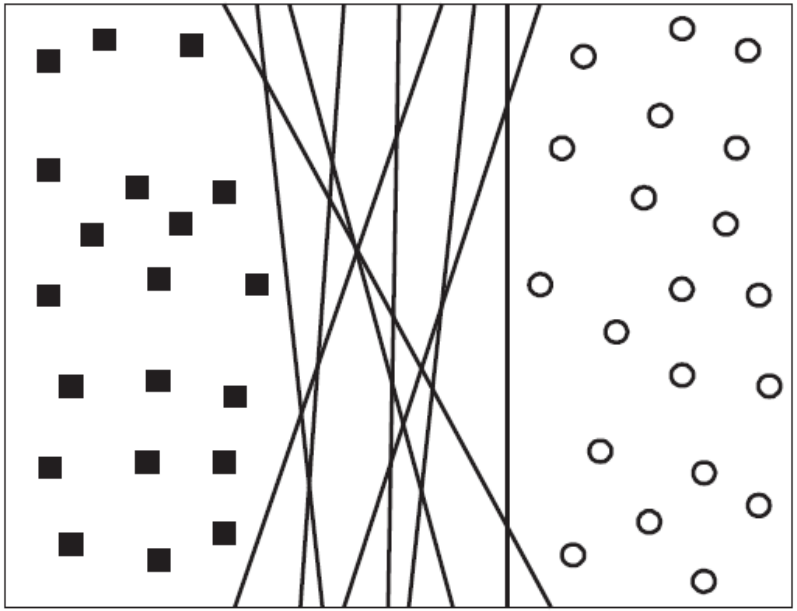
$$B_0 + B_1 X_1 + B_2 X_2 + \ldots + B_n X_n = 0$$

$$B_0 + B_1 X_1 + B_2 X_2 + \ldots + B_n X_n > 0$$

$$B_0 + B_1 X_1 + B_2 X_2 + \ldots + B_n X_n < 0$$

Maximum Margin Hyperplane

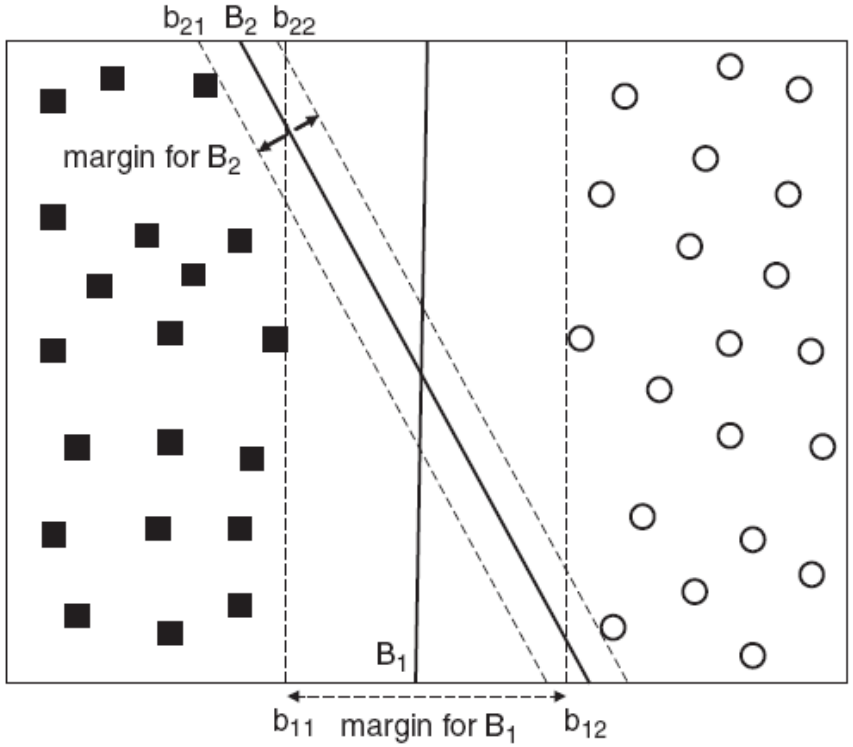
- What's the best separating hyperplane?



Intuition: the one that is farthest from the training observations.

Called the maximum margin hyperplane.

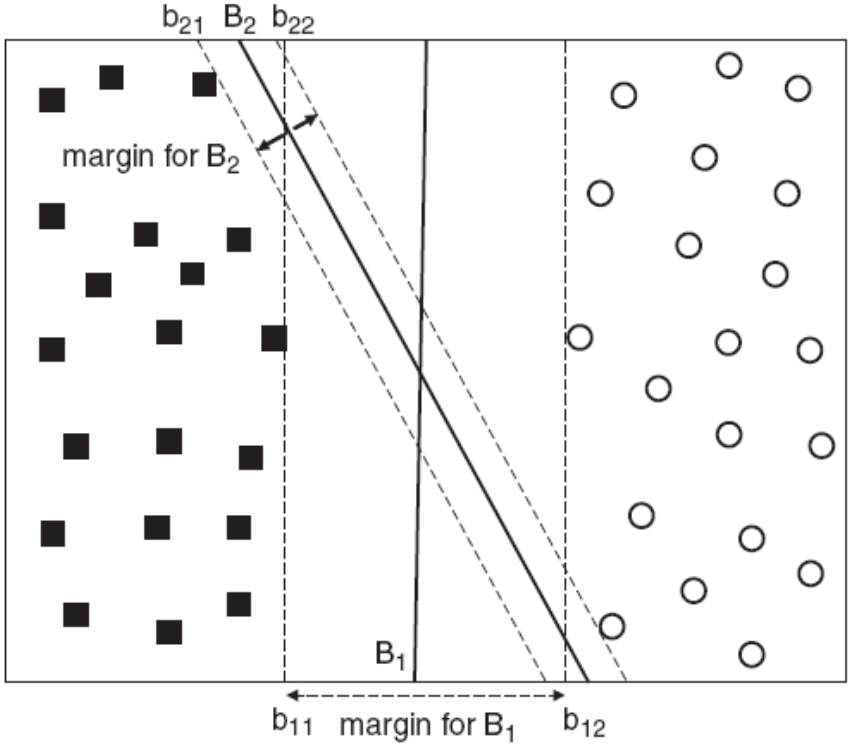
The Margin



- B_1 and B_2 are each separating hyperplanes
  - B_1 is better
- Margin: the smallest distance from the hyperplane to the training data

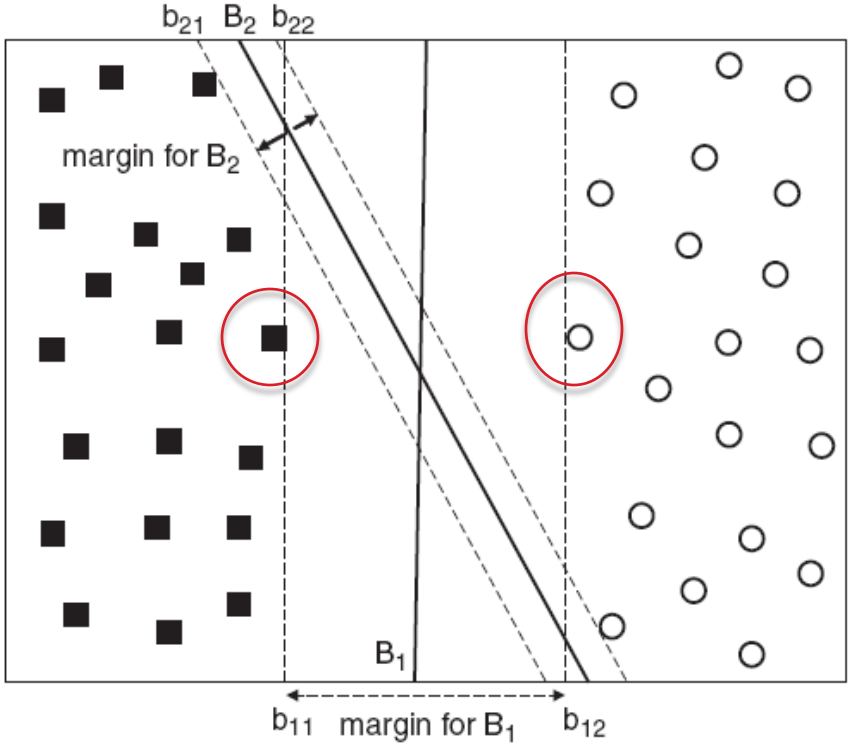
Represents the mid-line of the widest "slab" that can be inserted between the two classes.

Maximal Margin Hyperplane



- We want the hyperplane that has the greatest margin.
  - That is, B_1 instead of B_2 or any of the other infinitely many separating hyperplanes

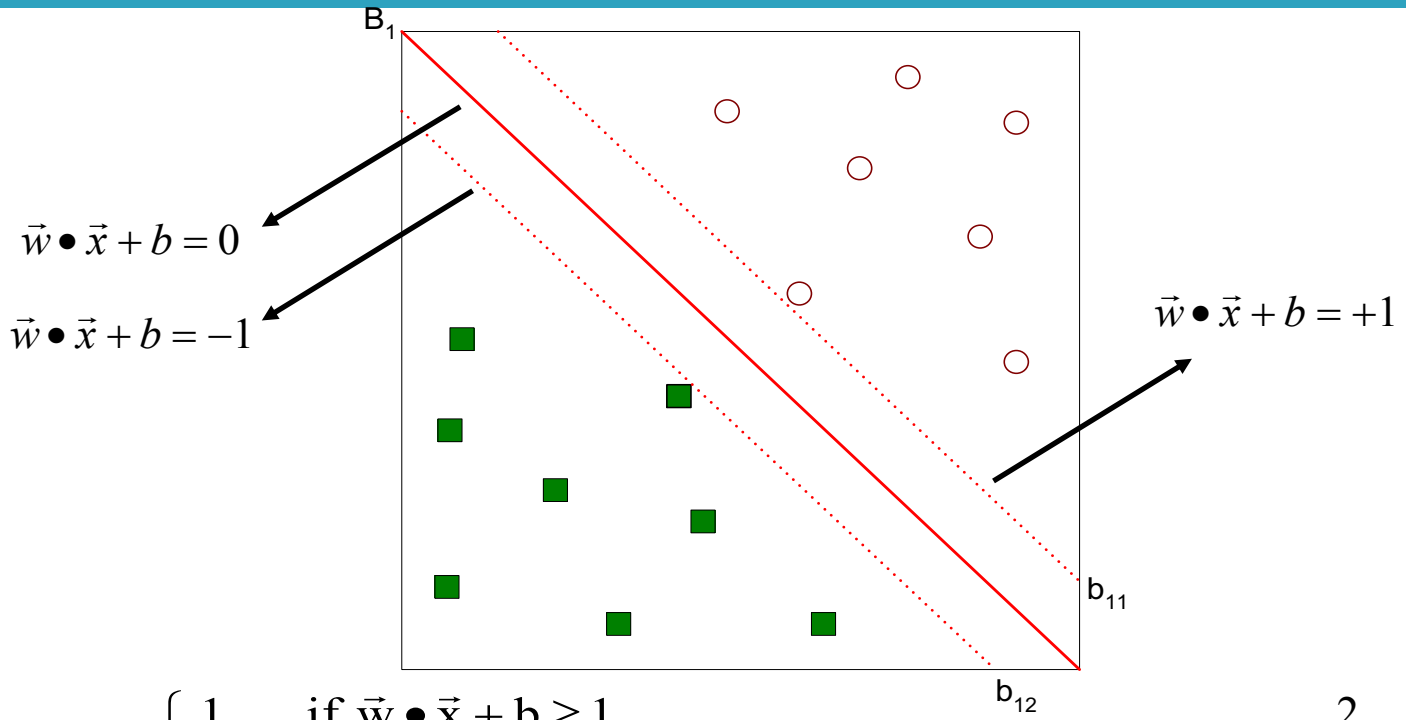
Maximal Margin Hyperplane



- Support Vectors: the points in the data, that if moved, the maximal margin hyperplane would move as well.

Moving any of the other data points would not affect the model.

Maximal Margin Hyperplane



$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\| \vec{w} \|^2}$$

Figuring Out the Maximal Margin Classifier

- We want to maximize: $\text{Margin} = \frac{2}{\|\vec{w}\|^2}$
- Which is equivalent to minimizing: $L(w) = \frac{\|\vec{w}\|^2}{2}$
- But subjected to the following constraints:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 \end{cases}$$

  - This is a constrained optimization problem
    - Numerical approaches to solve it (e.g., quadratic programming)

Support Vector Classifier

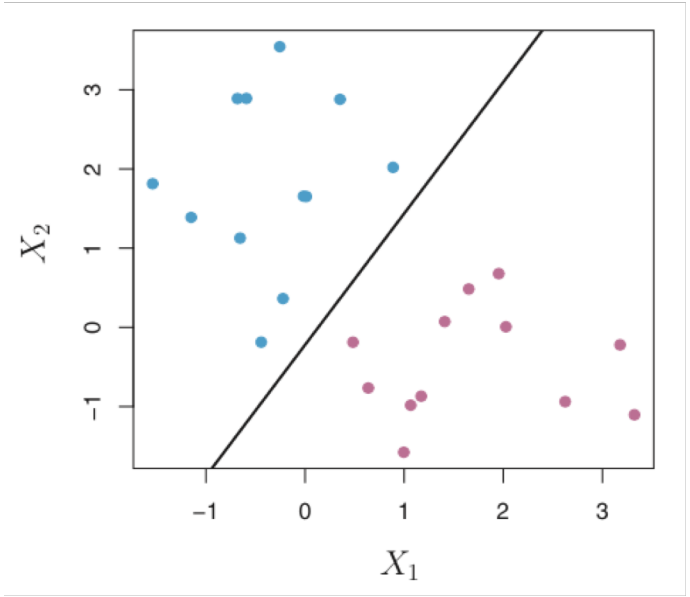
- Maximum Margin Classifier is natural way to perform classification *if a separating hyperplane exists.*
  - Perfect segmentation between two classes
- In many cases, no separating hyperplane will exist
  - Find a hyperplane that *almost* perfectly segments the classes
  - This generalization is called: support vector classifier

Support Vector Classifier

- Maximal Margin Classifier: no training errors allowed
- Support Margin Classifier: tolerate training errors
  - *Approach:* Soft margin
  - Will allow construction of linear decision boundary even when classes are not linearly separable

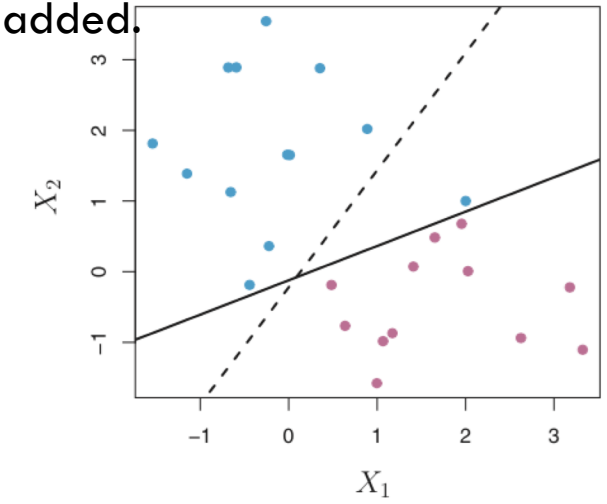
Support Vector Classifier

Additional motivation:



Maximum margin classifier.
Perfectly segments training data.

New data point added.

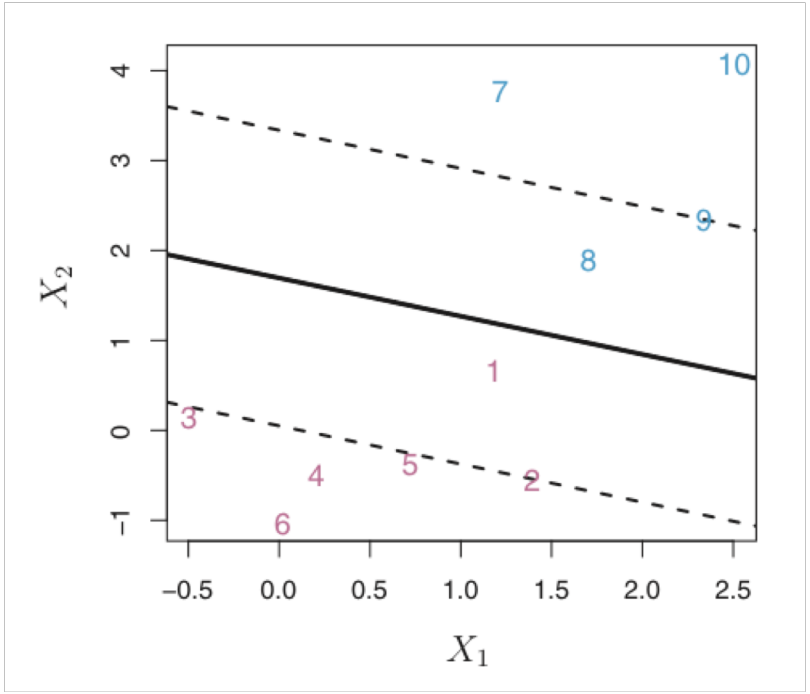


Dramatic shift in maximal margin hyperplane.
Model has *high variance* when trying to maintain perfect segmentation.

Support Vector Classifier

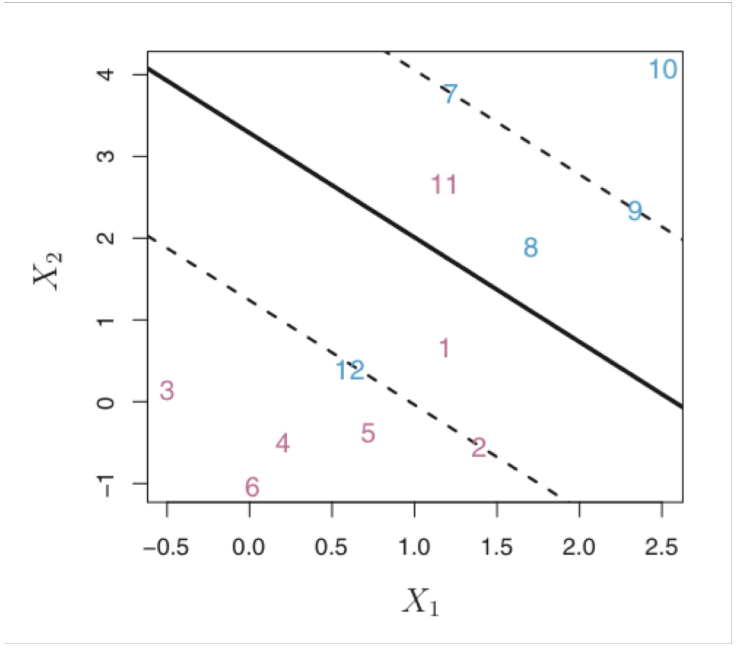
- So, interested in:
  - Greater robustness to individual data instances
  - Better classification of *most* of the training data
- Some misclassifications permitted:
  - "Soft" margin: because margin can be violated by some of the instances
  - Introduce "slack" variables

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 - \xi_i \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 + \xi_i \end{cases}$$



Red Instances:

- 3 4 5 6 on correct side of margin
- 2 is on the margin
- 1 is on the wrong side of the margin



Red Instances:

- 3 4 5 6 on correct side of margin
- 2 is on the margin
- 1 is on the wrong side of the margin
- 11 is on the wrong side of the hyperplane

Slack variables

- How to interpret – we can't allow unlimited violations that all fall within the allowance or we'll end up with margins that don't mean anything
- Need to minimize with C and k being parameters – can assume k = 1 to simplify the problem:

$$L(w) = \frac{\| \vec{w} \|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k \right)$$

Using Support Vector Classifier for Classification

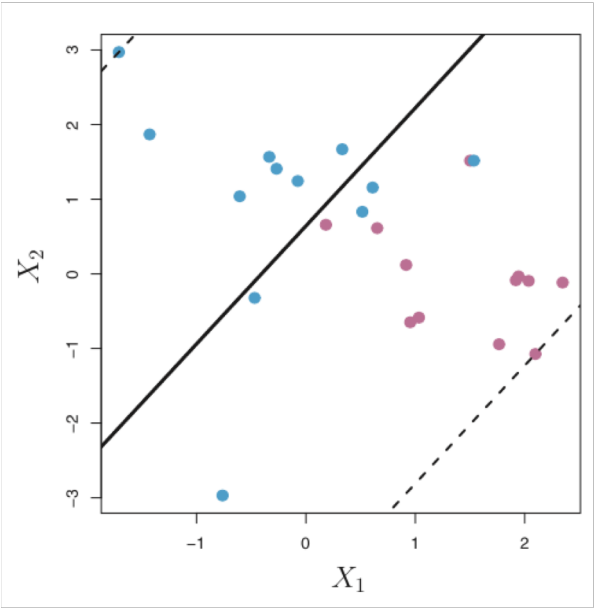
- Same as before.
- Which side of the line is the test instance on?

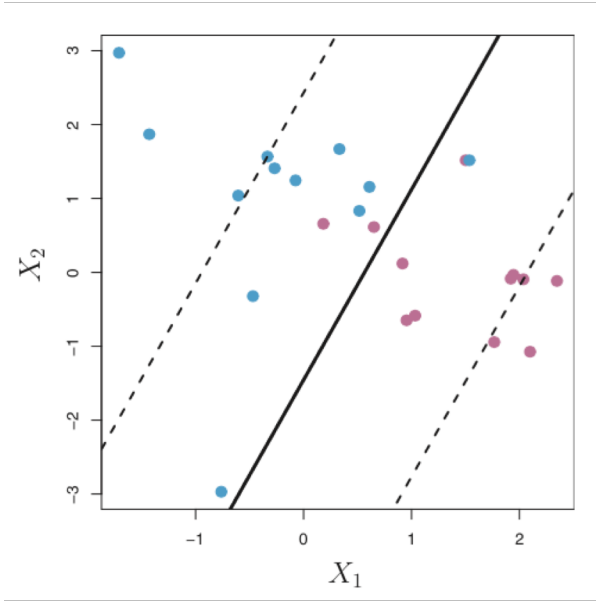
Constructing the Support Vector Classifier

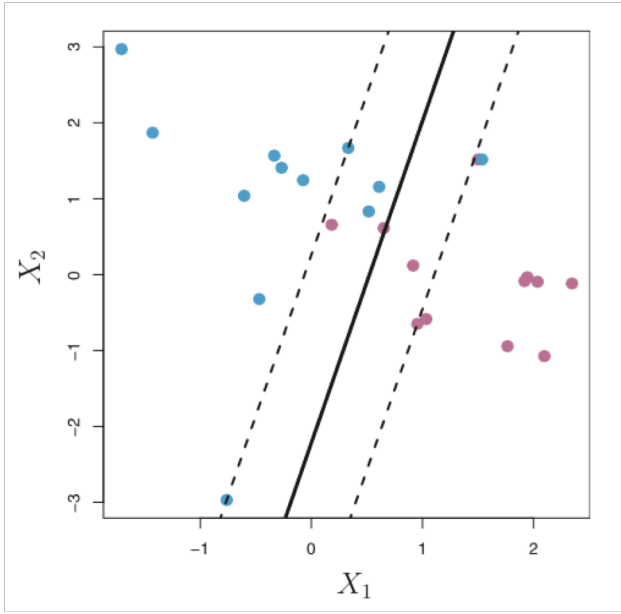
- More interesting.
- How much "softness" (misclassifications) in the *soft margin* is ideal?
- Specification of nonnegative tuning parameter C
  - Generally chosen by analyst following cross-validation
  - *Large C:* wider margin; more instances violate margin
  - *Small C:* narrower margin; less tolerance for instances that violate margin

Same data points.

Larger C to Smaller C





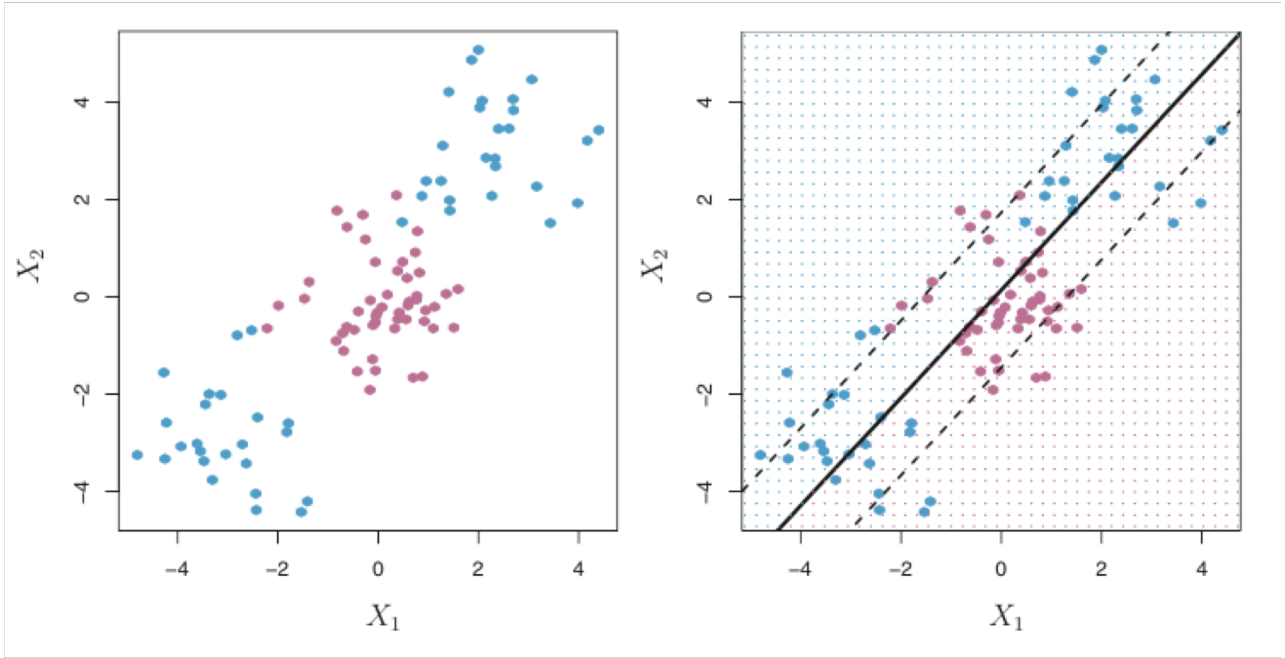


Lower variance.

Higher variance.

Support Vector Machines

- What if a non-linear decision boundary is needed?



Poor performance using this decision boundary.

Support Vector Machines

- *Idea:* transform the data from its original coordinate space in X into a new space $\Phi(X)$ so that a linear decision boundary can separate the two classes
  - Φ: nonlinear transformation
- Huh?

Instead of fitting a support vector classifier using n features:

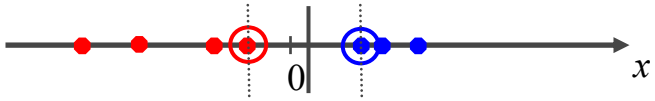
$X_1, X_2, \ldots, X_n$

… use $2n$ features:

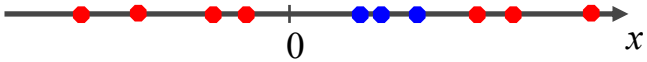
$X_1, X_1^2, X_2, X_2^2, \ldots, X_n, X_n^2$

Nonlinear SVMs

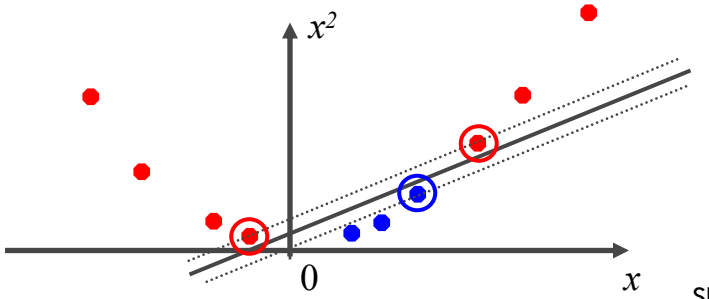
- Linearly separable dataset in 1D:



- Non-separable dataset in 1D:



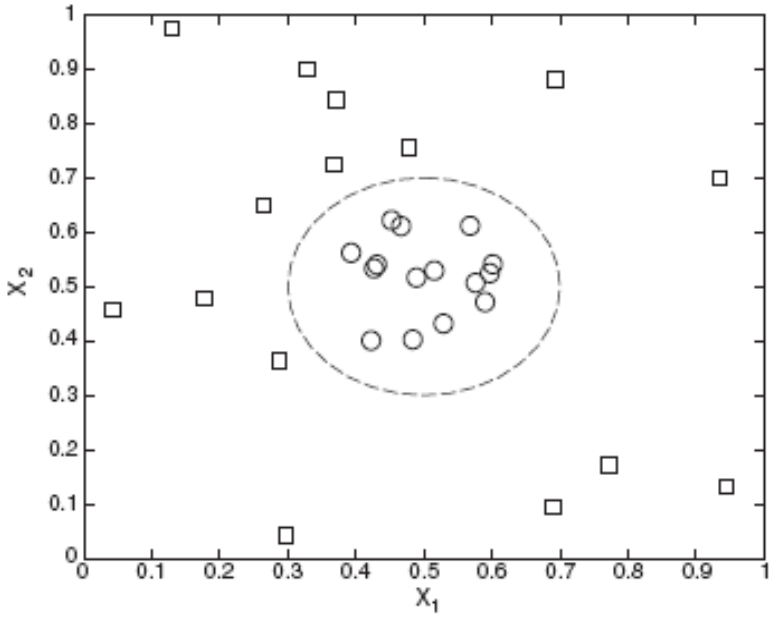
- We can map the data to a *higher-dimensional space*:



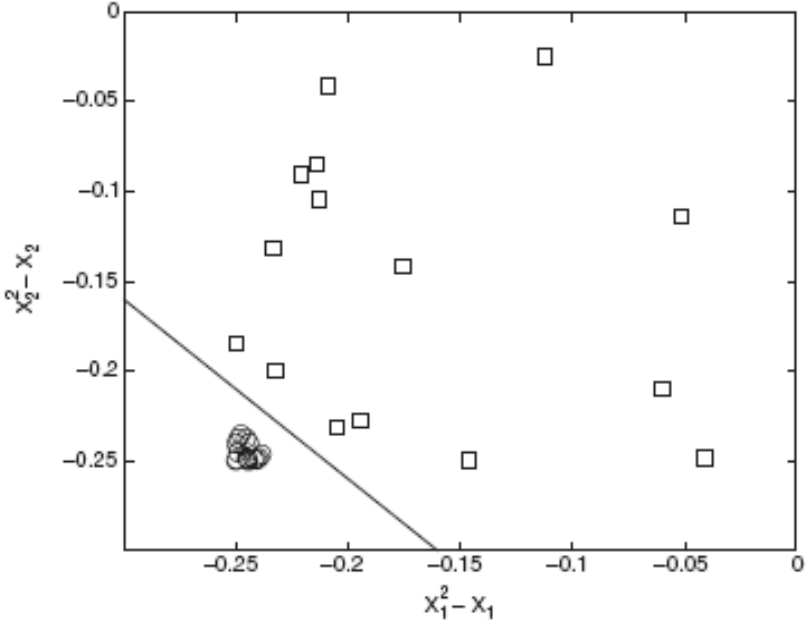
Slide credit: Andrew Moore

$$\Phi : (x_1, x_2) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$$

Attribute Transformation



(a) Decision boundary in the original two-dimensional space.



(b) Decision boundary in the transformed space.

Support Vector Machines

- Enlarged “feature space” compared to original “feature space”
- Can even extend to higher-order polynomial terms.

- *Downside:* can easily end up with huge number of features
  - overfitting

The Kernel "Trick"

- It's hard to know what transformations to apply, so maybe we can do this another way with the original feature set…
- Essentially, we need to expand our definition of "distance" from strict Euclidean
- Properties
  - Non-negative
  - Symmetry (distance between a and b = to distance between b and a)
  - Identity (distance between a and a is 0)
  - Triangle inequality (sum of distances between a and b and b and c <= distance between a and c)

Kernels

- These generalized distances are called metrics
- Metrics are defined using special mathematical functions that satisfy the distance definition conditions. These functions are known as *kernels*.
- Kernels take in two inputs and output a similarity (distance) measure
- Kernels correspond to dot products of transformed feature vectors (Mercer's Theorem)

Why Use Kernels?

- Often, computing the kernel is easy, but computing the feature vector corresponding to the kernel is really hard.
- The feature vector for even simple kernels can blow up in size, and sometimes the corresponding feature vector is infinite dimensional.
- Kernel trick not limited to SVMs, other machine learning algorithms can work with dot products

Kernel Examples

- Linear
- Polynomial
- Sigmoid
- Gaussian RBF

Other extensions to SVMs:

- *Regression* instead of *classification*
- *Multiclass* problems instead of *binary*

References

- *Introduction to Data Mining*, 1st edition, Tan et al.
- *An Introduction to Statistical Learning*, 1st edition, James et al.