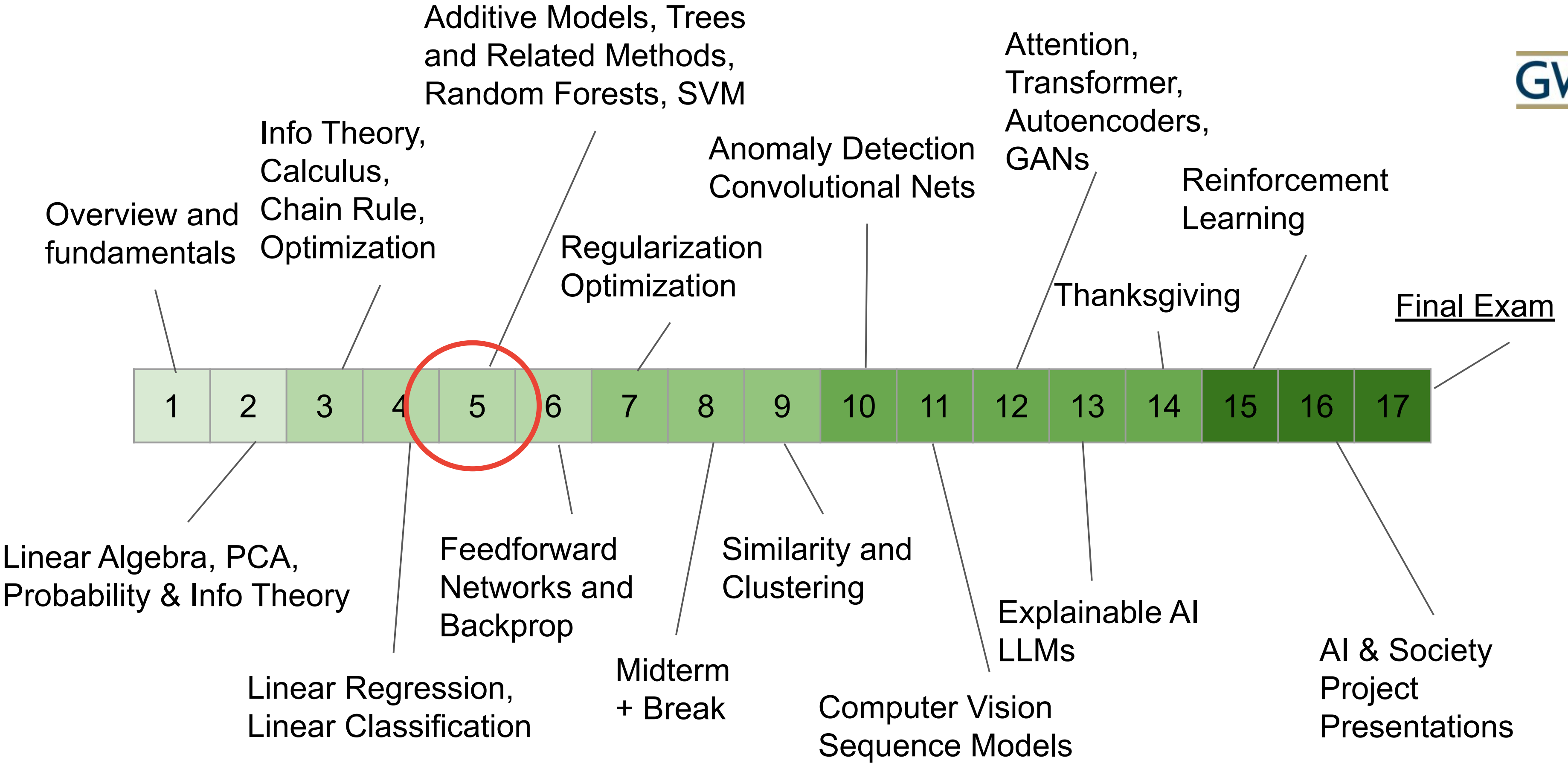
THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

CS 4364/6364

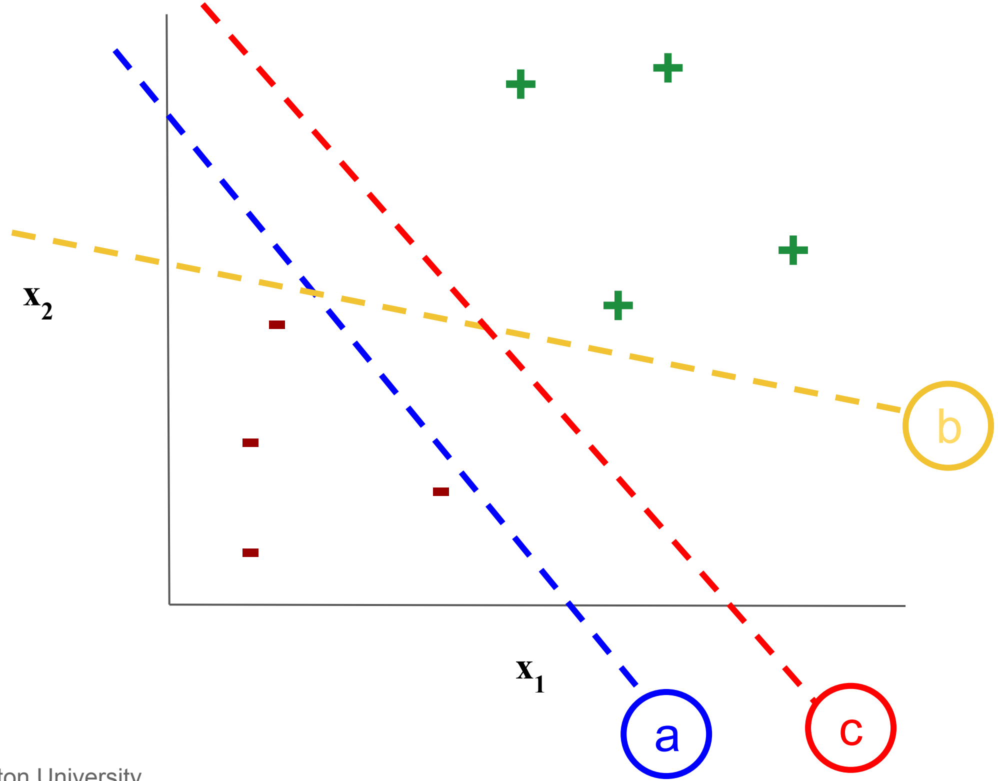
Machine Learning

Fall Semester 9/21/2023
Lecture 9.
Support Vector Machines

John Sipple
jsipple@gwu.edu


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
Overview and fundamentals
Linear Algebra, PCA, Probability & Info Theory
Info Theory, Calculus, Chain Rule, Optimization
Linear Regression, Linear Classification
Additive Models, Trees and Related Methods, Random Forests, SVM
Feedforward Networks and Backprop
Regularization Optimization
Midterm + Break
Similarity and Clustering
Anomaly Detection Convolutional Nets
Computer Vision Sequence Models
Attention, Transformer, Autoencoders, GANs
Explainable AI LLMs
Thanksgiving
Reinforcement Learning
AI & Society Project Presentations
Final Exam

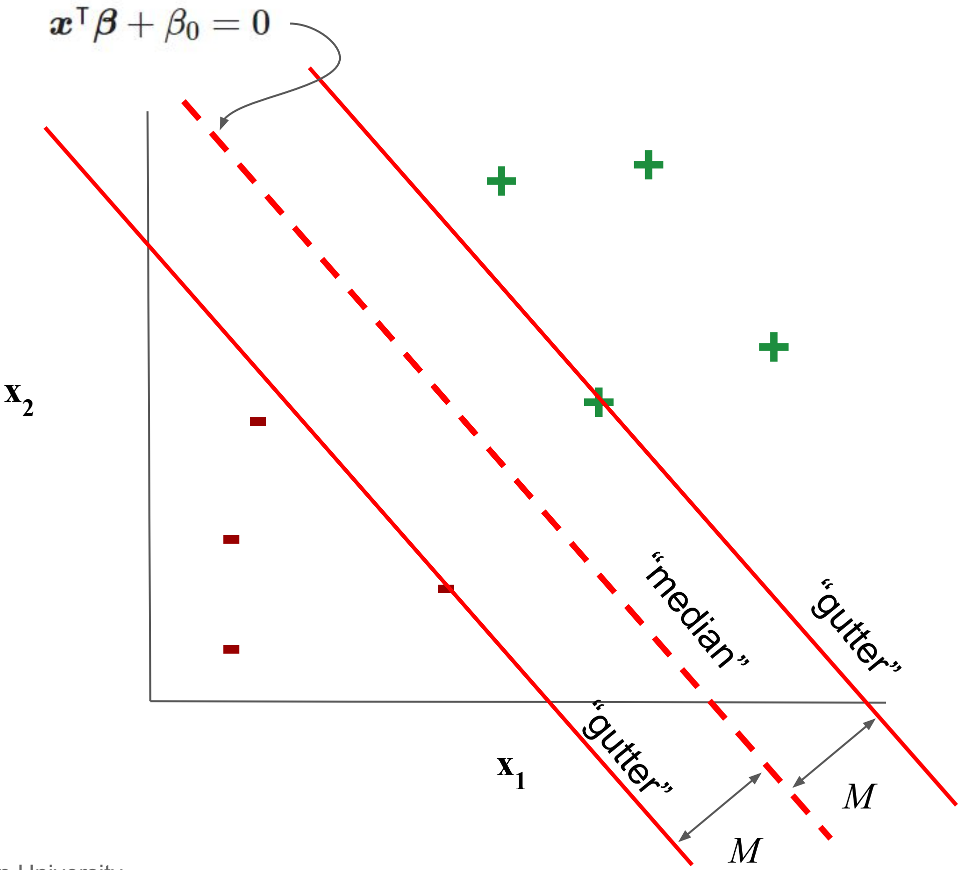
Linear Classification Decision Boundaries



Given a series of points $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$, where y_1=1 if +, and y_1=-1 if -:

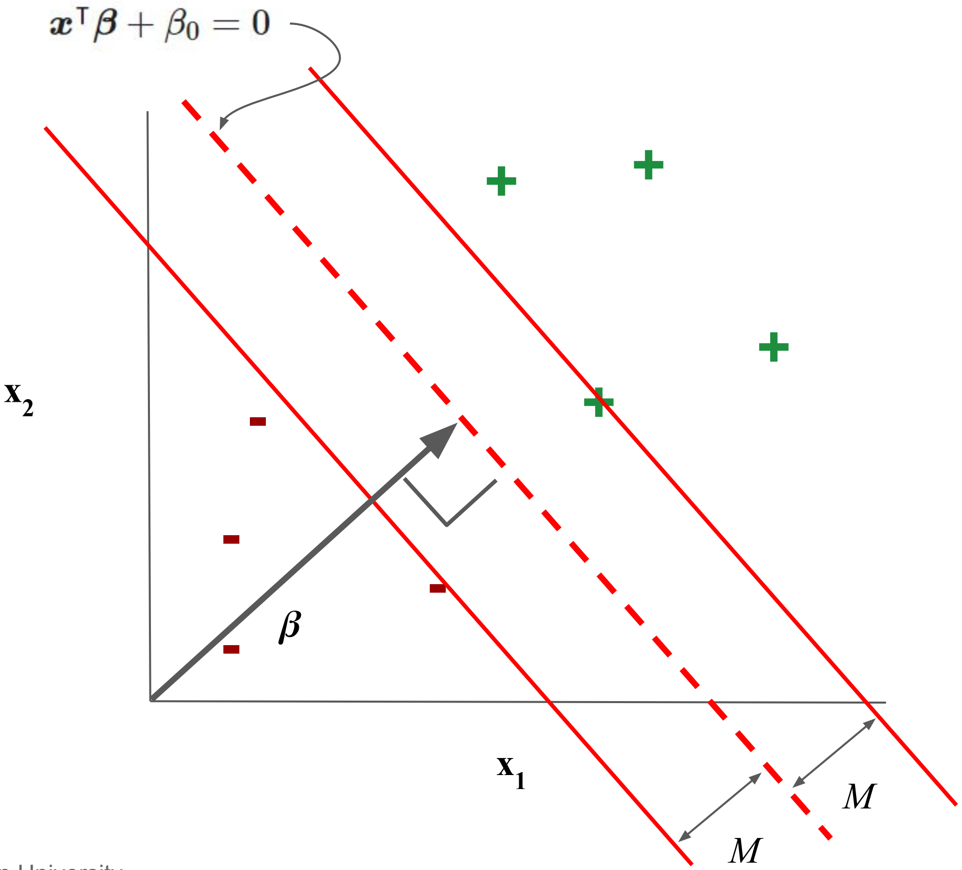
Which of the following is the best decision boundary?

Linear Decision Boundaries

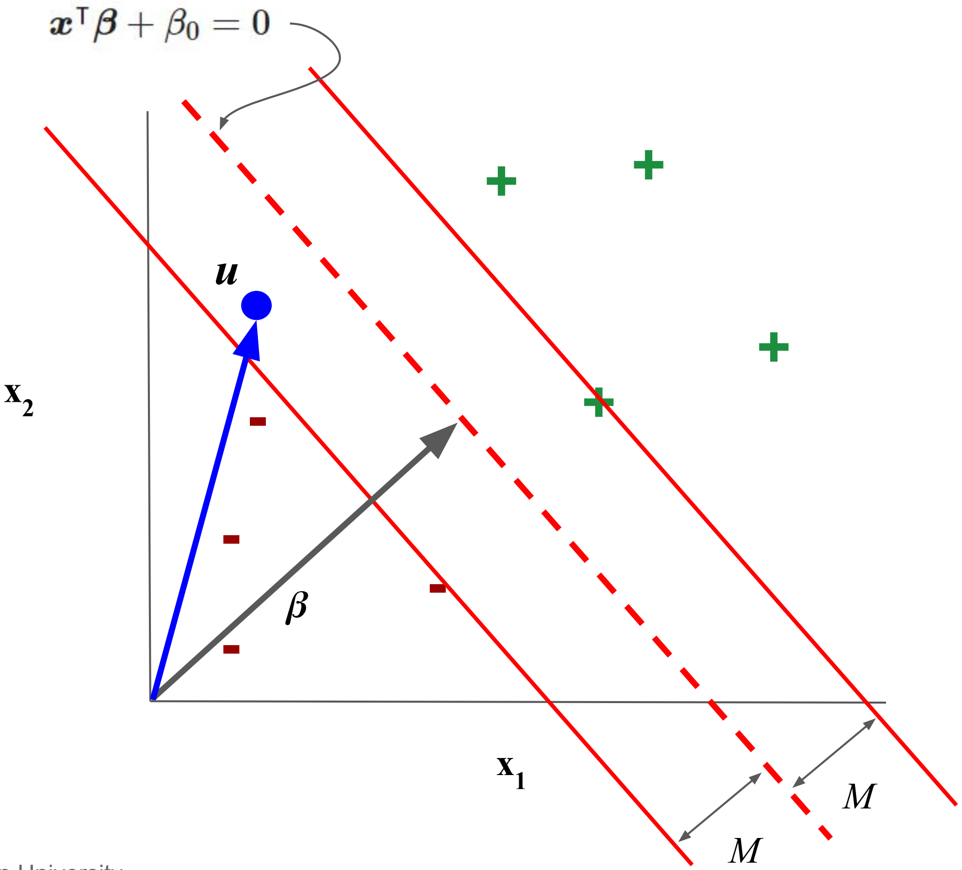


Idea: Let's create the widest street between the two classes with as wide of a margin M as possible, and make the decision boundary the centerline.

Linear Decision Boundaries with Margin



Support Vector Decision Rule

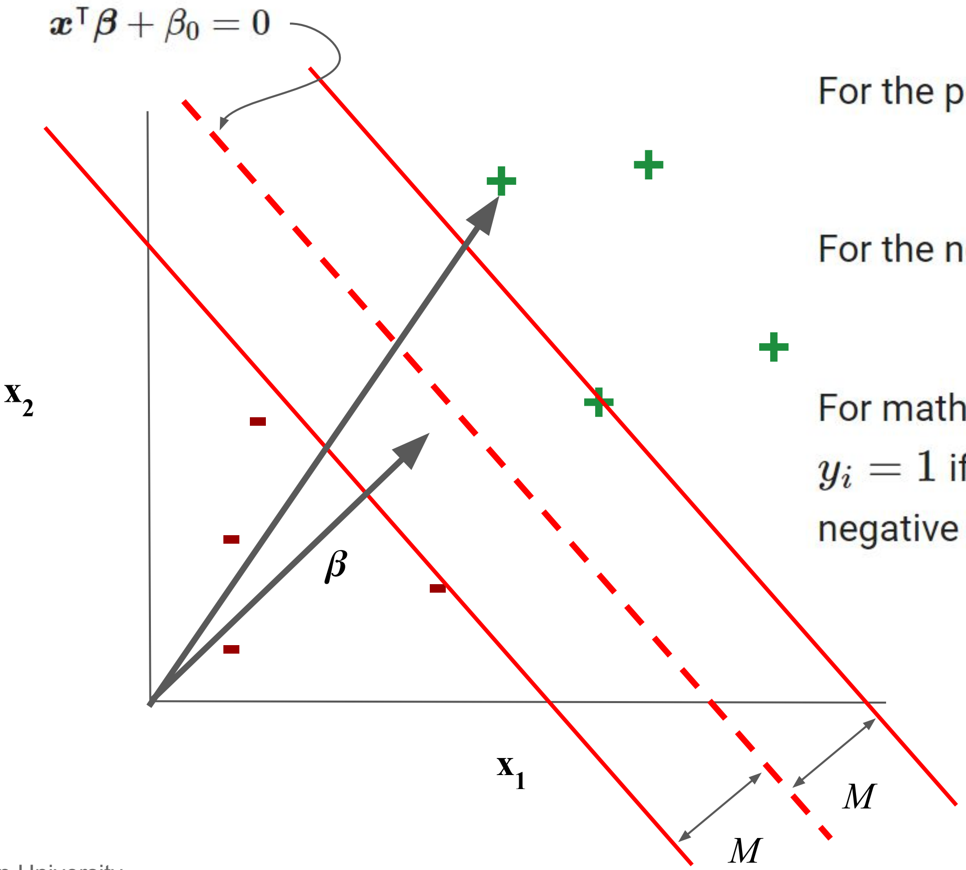


Decision Rule:
Classify unknown point $\boldsymbol{u}$ as class + if

$$\boldsymbol{u}^\top \cdot \boldsymbol{\beta} + \beta_0 \geq 0$$

Otherwise, classify $\boldsymbol{u}$ as class -.

Define the + and - Margin Boundary



For the positive sample:

$$\boldsymbol{x}_+^\intercal \cdot \boldsymbol{\beta} + \beta_0 \geq 1$$

For the negative sample:

$$\boldsymbol{x}_-^\intercal \cdot \boldsymbol{\beta} + \beta_0 \leq -1$$

For mathematical convenience, assign lables $y_i = 1$ if positive class and $y_i = -1$ for the negative class.

Define the + and - Boundary for the Margin

For the positive class:

$$y_i(\boldsymbol{x}_{i+}^{\top} \cdot \boldsymbol{\beta} + \beta_0) \geq 1$$

$$y_i(\boldsymbol{x}_{i+}^{\top} \cdot \boldsymbol{\beta} + \beta_0) - 1 \geq 0$$

And the negative class:

$$y_i(\boldsymbol{x}_{i-}^{\top} \cdot \boldsymbol{\beta} + \beta_0) \geq 1$$

$$y_i(\boldsymbol{x}_{i-}^{\top} \cdot \boldsymbol{\beta} + \beta_0) - 1 \geq 0$$

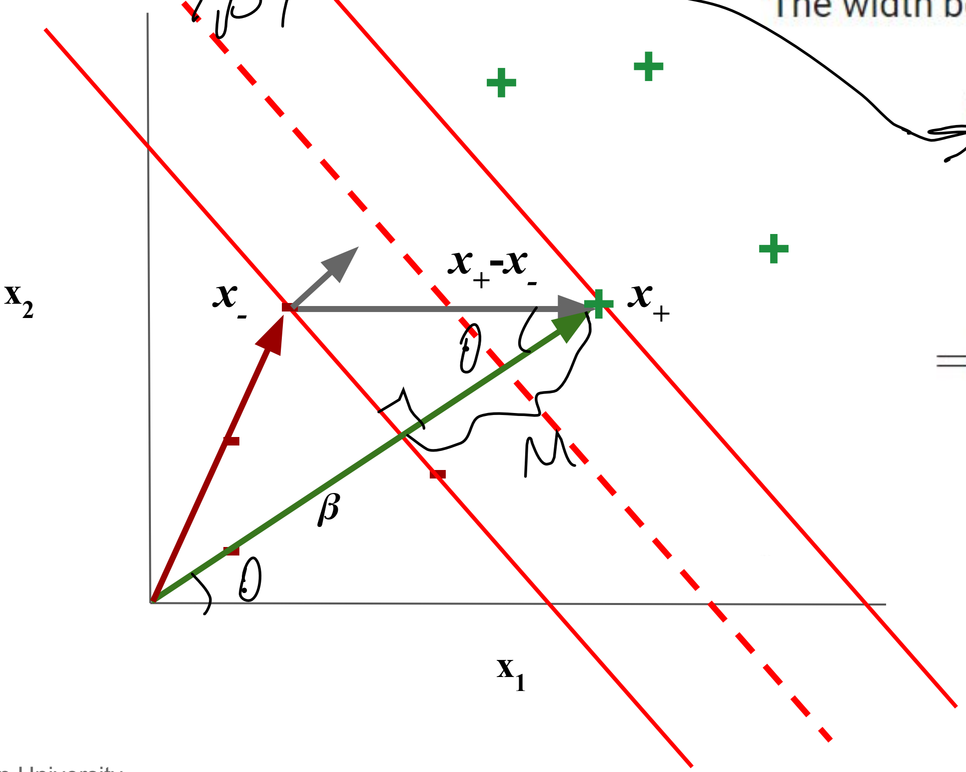
The same holds for both negative and positive classes in the "gutter":

$$y_i(\boldsymbol{x}_{i}^{\top} \cdot \boldsymbol{\beta} + \beta_0) - 1 = 0$$

Evaluate the Width of the Margin

$M = |x_+ - x_-| \cos\theta$

$|x| \cos\theta = \frac{x \beta}{|\beta|}$ $\leftarrow$ $x \cdot \beta = |x| \cdot |\beta| \cdot \cos\theta$



The width between boundaries are calculated as:

$$
\begin{aligned}
\text{width} &= (\boldsymbol{x}_+^{\mathsf{T}} - \boldsymbol{x}_-^{\mathsf{T}}) \cdot \frac{\boldsymbol{\beta}}{||\boldsymbol{\beta}||} \\
&= (\boldsymbol{x}_+^{\mathsf{T}} \cdot \boldsymbol{\beta} - \boldsymbol{x}_-^{\mathsf{T}} \cdot \boldsymbol{\beta}) \frac{1}{||\boldsymbol{\beta}||} \\
&= ((1 - \beta_0) - (-1 - \beta_0)) \frac{1}{||\boldsymbol{\beta}||} \\
&= \frac{2}{||\boldsymbol{\beta}||}
\end{aligned}
$$

Maximizing the margin's width

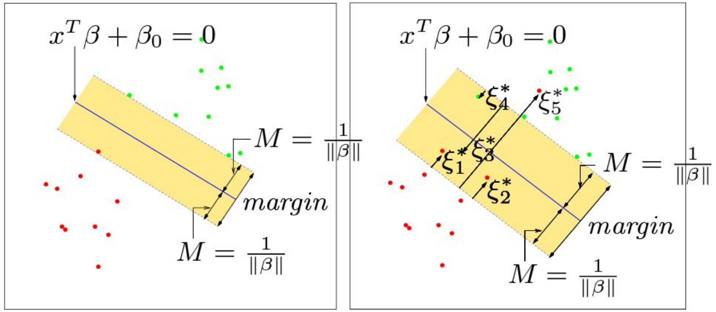


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width $2M = 2/\|\beta\|$. The right panel shows the nonseparable (overlap) case. The points labeled ξ_j^* are on the wrong side of their margin by an amount $\xi_j^* = M\xi_j$; points on the correct side have $\xi_j^* = 0$. The margin is maximized subject to a total budget $\sum \xi_i \leq$ constant. Hence $\sum \xi_j^*$ is the total distance of points on the wrong side of their margin.*

Since $\mathrm{width} = 2M$, implies $M = \frac{1}{||\boldsymbol{\beta}||}$ as illustrated in Figure 12.1

We would like to maximize $M = \frac{1}{||\boldsymbol{\beta}||}$, which implies minimizing $||\boldsymbol{\beta}||$.

But, we can also choose to minimize a more convenient expression:

$$\beta^* = \arg\min_{\boldsymbol{\beta},\beta_0} \frac{1}{2}||\boldsymbol{\beta}||^2$$

Which should look like L2 regularization.

Constrained minimization problem

Define a constrained minimization problem:

$$L_P = \frac{1}{2}||\boldsymbol{\beta}||^2$$

$$\text{subject to } y_i(\boldsymbol{x}_i^\intercal \cdot \boldsymbol{\beta} + \beta_0 - 1) = 0$$

And in the Lagrange function:

$$L_P = \frac{1}{2}||\boldsymbol{\beta}||^2 - \sum_{i=1}^{N} \alpha_i \left[y_i(\boldsymbol{x}_i^\intercal \cdot \boldsymbol{\beta} + \beta_0) - 1 \right]$$

Compute the derivative of L_P with respect to $\boldsymbol{\beta}$

Next, we compute the gradient of L_P with respect to $\boldsymbol{\beta}$:

$$\frac{\partial L_P}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i = 0$$

This gives us a nice expression for $\boldsymbol{\beta}$:

$$\boldsymbol{\beta} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i$$

Note that β is a linear sum of the training sample.

Compute the derivative of L_P with respect to β_0

Next, take the partial derivative of L_P with respect to β_0 and set that to zero too:

$$\frac{\partial L_P}{\partial \beta_0} = -\sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

Combining terms and rewriting the Lagrangian

Next, let's plug in these terms into the Lagrange function:

$$L_P = \frac{1}{2}||\boldsymbol{\beta}||^2 - \sum_{i=1}^{N} \alpha_i \left[y_i(\boldsymbol{x}_i^\intercal \cdot \boldsymbol{\beta} + \beta_0) - 1 \right]$$

becomes:

$$L_P = \frac{1}{2}\left(\sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i\right)^\intercal \cdot \left(\sum_{j=1}^{N} \alpha_j y_j \boldsymbol{x}_j\right) - \left(\sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i\right)^\intercal \cdot \left(\sum_{j=1}^{N} \alpha_j y_j \boldsymbol{x}_j\right) - \sum_{i=1}^{N} \alpha_i y_i \beta_0 + \sum_{i=1}^{N} \alpha_i$$

Simplifying, and evaluating the double summation results in Equation 12.13:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^\intercal \boldsymbol{x}_j$$

Now, we can apply iterative gradiant ascent to maximize L_D.

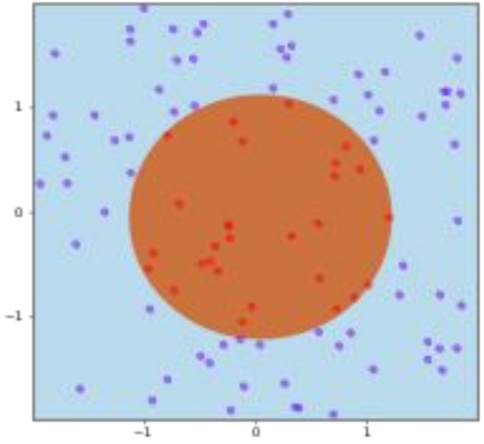
Updated Decision Rule

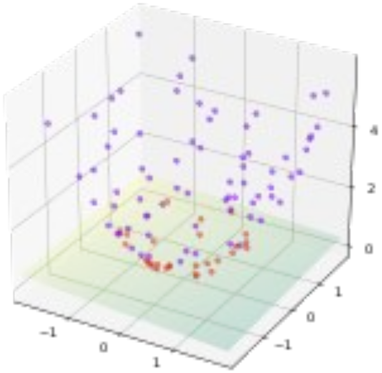
The points in the training set whose αs are nonzero, are the **support vectors**.

The decision rule then depends entirely on the dot project of the unknown point $\boldsymbol{u}$ and the training sample $\boldsymbol{x}_1, \boldsymbol{x}_2 \ldots \boldsymbol{x}_N$.

$$\text{If } \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i^{\intercal} \cdot \boldsymbol{u} + \beta_0 \geq 0 \text{ then } class = +$$

Projections can enable linear decision boundaries





Source: https://en.wikipedia.org/wiki/Kernel_method

Nonlinear decision boundaries

We can use the dot product in the decisin rule and apply the "kernel trick":

Assuming that $h(\boldsymbol{x}) : \mathbb{R}^p \to \mathbb{R}^q$ and that $q > p$, we could maximize in training:

$$h(\boldsymbol{x}_i)^\intercal \cdot h(\boldsymbol{x}_j)$$

and then use

$$h(\boldsymbol{x}_i)^\intercal \cdot h(\boldsymbol{u})$$

to predict.

Kernel Functions for Nonlinear Decision Boundaries

We don't even need to specify the form of the projection with a kernel function:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = h(\boldsymbol{x}_i)^\intercal \cdot h(\boldsymbol{x}_j)$$

Example Kernel Functions:

- *dth*-Degree Polynomial: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \boldsymbol{x}_i^\intercal \cdot \boldsymbol{x}_j)^d$
- Radial Basis Function: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma||\boldsymbol{x}_i - \boldsymbol{x}_j||^2)$
- Neural Tangent: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\kappa_1 \boldsymbol{x}_i^\intercal \cdot \boldsymbol{x}_j + \kappa_2)$

Updated decision function:

$$\text{If } \sum_{i=1}^{N} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{u}) + \beta_0 \geq 0 \text{ then } class = +$$

Tuning the Support Vector Machine

The misclassification cost, C, is a bound placed on the αs, helping to trade off the margin size M and the misclassfication rate.

- For a **small** misclassification cost C, the margin M will be **large**.
- A **large** misclassification cost, C, the margin M will be **small**.

SVM Demo Colab

SVM Colab from Python Data Science Handbook by Jake VanderPlas

https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.07-Support-Vector-Machines.ipynb#scrollTo=qj4n6HtN1djQ

Readings

Goodfellow Chapter 6