Introduction to Machine Learning

Support Vector Machines & Kernels

Ashley Gao

William & Mary

October 02, 2024

Overview

- Prediction
  - Why might predictions be wrong?
- Support vector machines
  - Do really well with linear models
- Kernels
  - Making the non-linear linear

Why Might Predictions be Wrong?

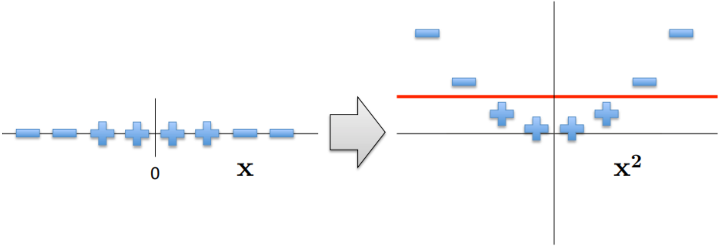
- True non-determinism
  - Flip a biased coin
  - $p(\text{heads}) = \theta$
  - Estimate θ
  - If $\theta > 0.5$, predict "heads", else "tails"
- Lots of ML research on problems like this:
  - Learn a model
  - Do the best you can in expectation

Why Might Predictions be Wrong?

- Partial observability
  - Something needed to predict y is missing from observation $\boldsymbol{x}$
  - N-bit parity problem
    - Determine the parity (even or odd) of a sequence of N binary bits.
    - The goal is to build a model that can correctly predict the parity of any given N-bit sequence.
- Noise in the observation $\boldsymbol{x}$
  - Measurement error
  - Instrument limitations
- Representational bias
- Algorithmic bias
- Bounded resources

Representational Bias

- Having the right features for $\boldsymbol{x}$ is crucial



Support Vector Machines

Strengths of SVMs

- Good generalization
  - in theory
  - in practice
- Works well with frew training instances
- Find globally best model
- Efficient algorithms
- Amenable to the kernel trick

Minor Notation Change

- To better match notations used in SVMs and to make matrix formulas simpler
- We will drop using superscripts for the i^{th} instance

i^{th} instance	$\boldsymbol{x}^{(i)}$	$\Rightarrow$	$\mathbf{x}_i$	**Bold** denotes vector
i^{th} instance label	$y^{(i)}$	$\Rightarrow$	y_i	**Non-bold** denotes scalar
j^{th} feature of i^{th} instance	$x_j^{(i)}$	$\Rightarrow$	x_{ij}	

Minor Notation Change

- Training instances: $\boldsymbol{x} \in \mathbb{R}^{d+1}, x_0 = 1, y \in -1, 1$
- Model parameters: $\boldsymbol{\theta} \in \mathbb{R}^{d+1}$
- Hyperplane: $\boldsymbol{\theta}^\top \boldsymbol{x} = \langle \boldsymbol{\theta}, \boldsymbol{x} \rangle = 0$
    - the vectors are orthogonal to each other
- Recall the inner (dot) product:

$$\langle \boldsymbol{\theta}, \boldsymbol{x} \rangle = \boldsymbol{\theta} \cdot \boldsymbol{x} = \boldsymbol{\theta}^\top \boldsymbol{x} = \sum_i \theta_i x_i \tag{1}$$

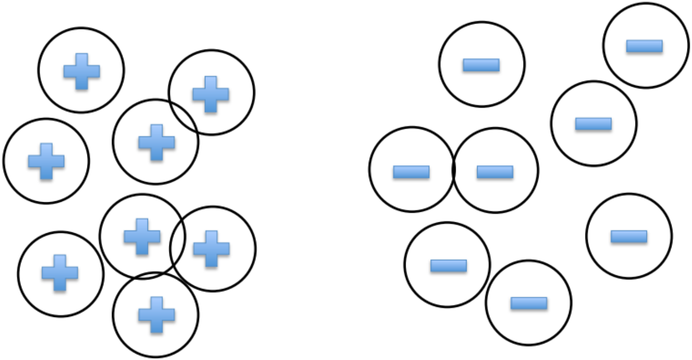
- Decision function: $h(\boldsymbol{x}) = \text{sign}(\boldsymbol{\theta}^\top \boldsymbol{x}) = \text{sign}(\langle \boldsymbol{\theta}, \boldsymbol{x} \rangle)$

Intuition

- Which line or classifier is better?

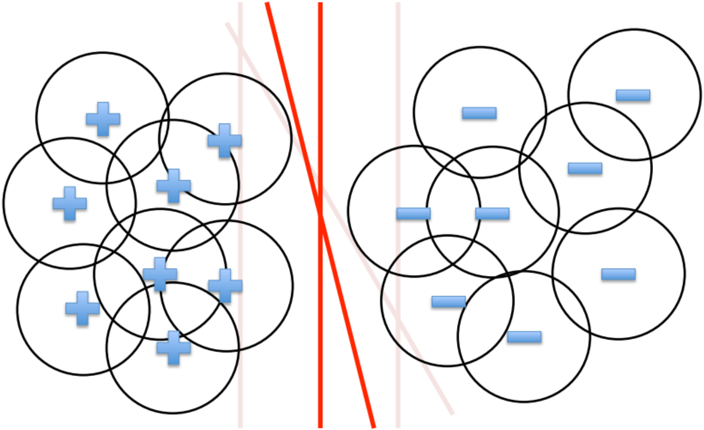
Noise in the observations

- Each circle denotes the “noise” that can happen when the sample is observed (e.g. faulty measuring equipment)
- A sample’s actual reading, in terms of features, can fall anywhere in the circle around the “true” values



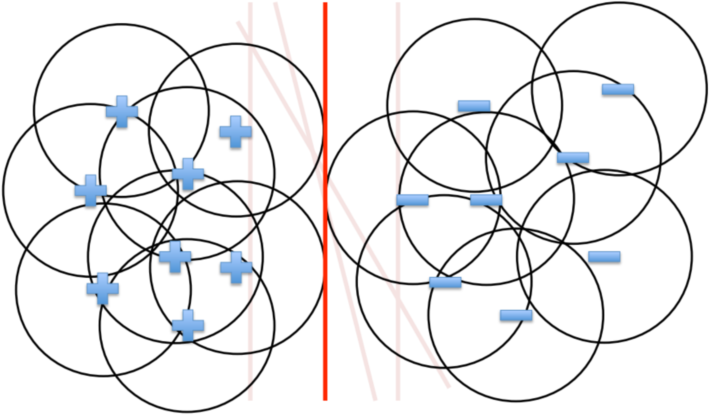
More Noise; Ruling Out Some Seperators

- When the readings (the values of features) become noisier, we can rule out some separators or classifiers

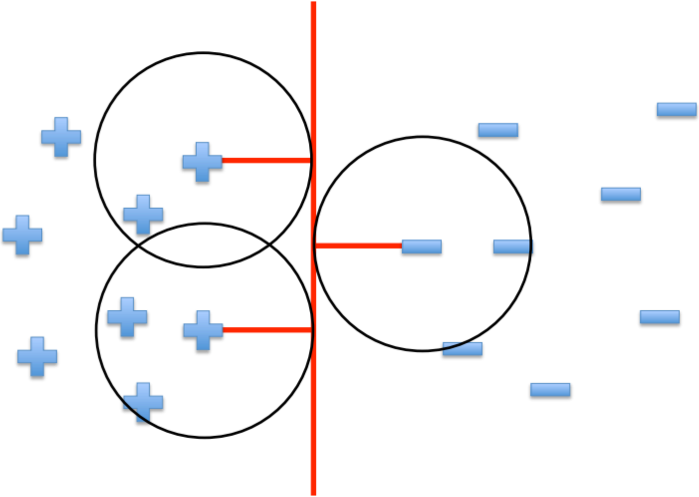


Only One Separator Remains

- Assuming that the values of the features are as noisy as they can get, provided that the samples are still linearly separable in the feature space.



Maximizing the Margin



“Wide” Separators

- We want the separators as “wide” as possible, to allow for more noise in the features of the samples.

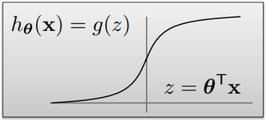
Why Maximize Margin

- Increasing margin reduces capacity
  - i.e. fewer possible models
- Lesson from Learning Theory:
  - If the following holds:
    - H is sufficiently constrained in size
    - and/or the size of the training dataset N is large
  - Then low training error is likely to be evidence of low generalization error

Alternative View of Logistic Regression

- if $y = 1$ we want $h_{\boldsymbol{\theta}} \approx 1, \boldsymbol{\theta}^\top \boldsymbol{x} \gg 0$
- if $y = 0$ we want $h_{\boldsymbol{\theta}} \approx 0, \boldsymbol{\theta}^\top \boldsymbol{x} \ll 0$

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\top \boldsymbol{x}}} \tag{2}$$



- We want to minimize the cross-entropy cost, by finding the $\boldsymbol{\theta}$ summing the losses across the classifications on all the samples

$$\mathcal{J}(\boldsymbol{\theta}) = -\sum_{i=1}^{N} [y_i \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + (1 - y_i)\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_i))] \tag{3}$$

- $\text{cost}_1(\boldsymbol{\theta}^\top \boldsymbol{x}_i) \Longleftrightarrow \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$
- $\text{cost}_0(\boldsymbol{\theta}^\top \boldsymbol{x}_i) \Longleftrightarrow \log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_i))$

Alternative View of Logistic Regression

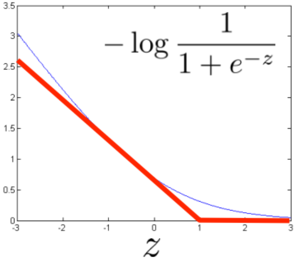
- Cost of one sample:

$$\mathcal{L}(\boldsymbol{\theta}) = -y_i \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - (1 - y_i)\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) \tag{4}$$

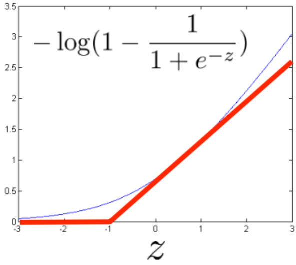
$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\top \boldsymbol{x}}} \tag{5}$$

$$z = \boldsymbol{\theta}^\top \boldsymbol{x} \tag{6}$$

If $y = 1$ (want $\boldsymbol{\theta}^\top \mathbf{x} \gg 0$):



If $y = 0$ (want $\boldsymbol{\theta}^\top \mathbf{x} \ll 0$):



Logistic Regression to SVMs

- Logistic Regression:

$$\min_{\boldsymbol{\theta}} -\sum_{i=1}^{N}[y_i \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + (1-y_i)\log(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}_i))] + \frac{\lambda}{2}\sum_{j=1}^{d}\theta_j^2 \quad (7)$$

- Support Vector Machines:

$$\min_{\boldsymbol{\theta}} C\sum_{i=1}^{N}[y_i \mathrm{cost}_1(\boldsymbol{\theta}^\top \boldsymbol{x}_i) + (1-y_i)\mathrm{cost}_0(\boldsymbol{\theta}^\top \boldsymbol{x}_i)] + \frac{1}{2}\sum_{j=1}^{d}\theta_j^2 \quad (8)$$

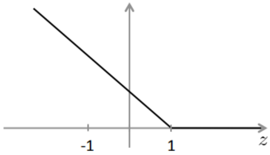
- C is a constant, a tunable hyperparameter. You can imagine it as similar to $\frac{1}{\lambda}$

The Hinge Loss

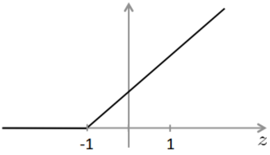
- Support Vector Machines:

$$\min_{\boldsymbol{\theta}} C \sum_{i=1}^{N} [y_i \text{cost}_1(\boldsymbol{\theta}^\top \boldsymbol{x}_i) + (1 - y_i)\text{cost}_0(\boldsymbol{\theta}^\top \boldsymbol{x}_i)] + \frac{1}{2} \sum_{j=1}^{d} \theta_j^2 \qquad (9)$$

If $y = 1$ (want $\boldsymbol{\theta}^\mathsf{T}\mathbf{x} \geq 1$):

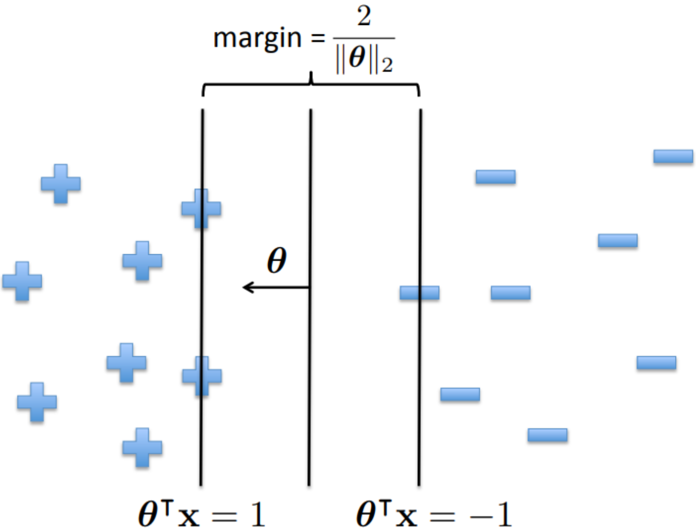


If $y = 0$ (want $\boldsymbol{\theta}^\mathsf{T}\mathbf{x} \leq -1$):

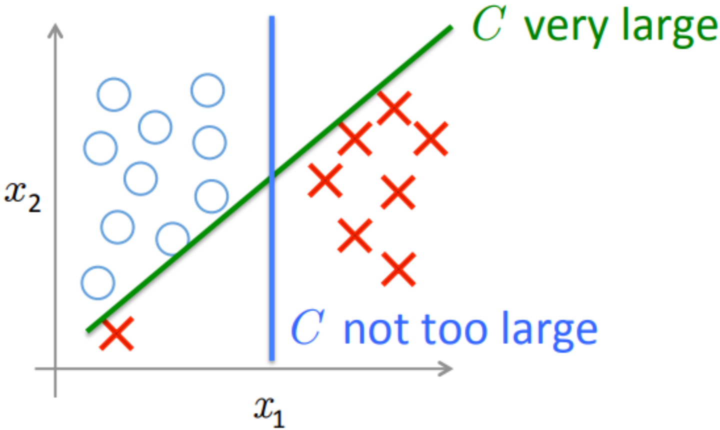


$$\ell_{\text{hinge}} = \max(0, 1 - y \cdot h(\boldsymbol{x})) \qquad (10)$$

Maximum Margin Hyperplane

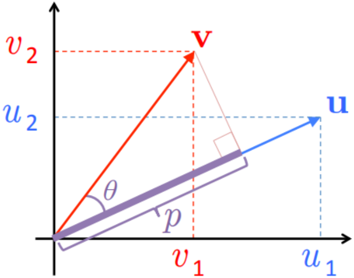


Large Margin Classifier in Presence of Outliers



Vector Inner Product

- Some quick review on the vector inner product:



$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$$\begin{aligned} \|\mathbf{u}\|_2 &= \text{length}(\mathbf{u}) \in \mathbb{R} \\ &= \sqrt{u_1^2 + u_2^2} \end{aligned}$$

Vector Inner Product

- Continued from the previous slide:

$$\boldsymbol{u}^\top \boldsymbol{v} = \boldsymbol{v}^\top \boldsymbol{u} \tag{11}$$

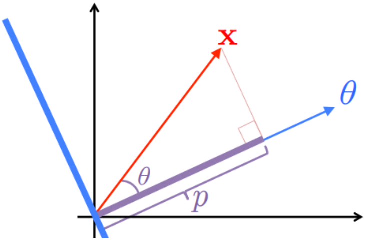
$$\boldsymbol{u}^\top \boldsymbol{v} = u_1 v_1 + u_2 v_2 \tag{12}$$

$$\boldsymbol{u}^\top \boldsymbol{v} = ||\boldsymbol{u}||_2 ||\boldsymbol{v}||_2 \cos\theta \tag{13}$$

$$\boldsymbol{u}^\top \boldsymbol{v} = p||\boldsymbol{u}||_2, \text{where } p = ||\boldsymbol{v}||_2 \cos\theta \tag{14}$$

Understanding the Hyperplane

- The hyperplane is orthogonal to the vector $\boldsymbol{\theta}$:



$$\boldsymbol{\theta}^\intercal \mathbf{x} = \|\boldsymbol{\theta}\|_2 \underbrace{\|\mathbf{x}\|_2 \cos\theta}_{p}$$
$$= p\|\boldsymbol{\theta}\|_2$$

- Assume $\theta_0 = 0$ so that the hyperplane is centered at the origin, and that $d = 2$ for it to be visually rendered in 2D. All for the purpose of simplicity of the demo.

Understanding the Hyperplane

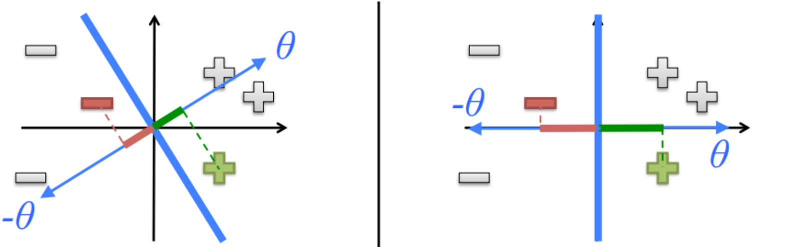
- Support Vector Machines objective to minimize:

$$\min_{\boldsymbol{\theta}} C \sum_{i=1}^{N} [y_i \text{cost}_1(\boldsymbol{\theta})^\top \boldsymbol{x}_i + (1 - y_i)\text{cost}_0(\boldsymbol{\theta})^\top \boldsymbol{x}_i] + \frac{1}{2} \sum_{j=1}^{d} \theta_j^2 \tag{15}$$

- Suppose that C is set to an arbitrarily small value $\Longleftrightarrow$ the first term becomes 0, for simplicity
- Now we are just minimizing the second term $\frac{1}{2} \sum_{j=1}^{d} \theta_j^2$
- Recall that $\boldsymbol{\theta}^\top \boldsymbol{x}_i \geq 1$ when $y_i = 1$ and $\boldsymbol{\theta}^\top \boldsymbol{x}_i \leq -1$ when $y_i = -1$

Maximizing the Margin

- Let p_i be the projection of $\boldsymbol{x}_i$ onto the vector $\boldsymbol{\theta}$



Since p is small, therefore $\|\boldsymbol{\theta}\|_2$ must be large to have $p\|\boldsymbol{\theta}\|_2 \geq 1$ (or ≤ -1)

Since p is larger, $\|\boldsymbol{\theta}\|_2$ can be smaller in order to have $p\|\boldsymbol{\theta}\|_2 \geq 1$ (or ≤ -1)

The SVN Dual Problem

- The primal SVM problem was given as

$$\frac{1}{2}\sum_{j=1}^{d}\theta_j^2, \text{ s.t. } y_i(\boldsymbol{\theta}^\top \boldsymbol{x}_i + b) \geq 1, \forall i \tag{16}$$

- Can be solved more efficiently by taking the Lagrangian dual
  - Duality is a common idea in optimization
  - It transforms a difficult optimization problem into a simpler one
  - Key idea: introduce slack variables α_i for each constraint
    - α_i indicates how important a particular constraint is to the solution

The Lagragian

- The Lagrangian dual refers to the dual formulation of an optimization problem using the Lagrange duality theory.
- It transforms a primal optimization problem into its dual problem
  - which can sometimes provide useful insights or computational advantages.
- The Lagrange duality theory is based on the concept of Lagrange multipliers
  - which are introduced to incorporate constraints into an optimization problem.
- By introducing these multipliers, the problem is transformed into a new formulation that involves maximizing or minimizing a function called the Lagrangian
  - which incorporates both the objective function and the constraints.

The SVM Dual Problem

- The Lagrangian is given as, s.t. $\alpha_i \geq 0 \; \forall i$:

$$\frac{1}{2} \sum_{j=1}^{d} \theta_j^2 - \sum_{i=1}^{n} \alpha_i (y_i (\boldsymbol{\theta}^\top x_i + b) - 1) \tag{17}$$

- We must minimize over $\boldsymbol{\theta}$ and maximize over $\boldsymbol{\alpha}$
- At optimal solution, partials w.r.t. $\boldsymbol{\theta}$'s are 0

The SVM Dual Representation

- After solving a bunch of linear algebra and calculus, want to maximize:

$$\mathcal{J}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \tag{18}$$

  Such that $\sum_i a_j y_j = 0$, s.t. $\alpha_i \geq 0, \forall i$
- The decision function is given by:

$$h(\boldsymbol{x}) = \text{sign}\left(\sum_{i \in SV} \alpha_i y_i \langle \boldsymbol{x}, \boldsymbol{x}_i \rangle + b \right) \tag{19}$$

$$b = \frac{1}{|SV|} \sum_{i \in SV} \left(y_i - \sum_{j \in SV} \alpha_j y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \right) \tag{20}$$

Understanding the Dual

- We have $\alpha_i \geq 0, \forall i$
  - Constaint weights (α_i's cannot be negative)
- We have $\sum_i \alpha_i y_i = 0$
  - Balances between the weight of constraints for different classes

Understanding the Dual

- After solving a bunch of linear algebra and calculus, want to maximize:

$$\mathcal{J}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \tag{21}$$

  Such that $\sum_i a_j y_j = 0$, s.t. $\alpha_i \geq 0, \forall i$
- $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ measures the similarity between the points
- Points with different labels increase the sum $\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$, while points with the same label decrease the sum

Understanding the Dual

- After solving a bunch of linear algebra and calculus, want to maximize:

$$\mathcal{J}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \tag{22}$$

  Such that $\sum_i a_j y_j = 0$, s.t. $\alpha_i \geq 0, \forall i$
- $a_i \geq 0$ and the constraint is tight $y_i(\boldsymbol{\theta}^\top \boldsymbol{x}_i) = 1$
  - Point is a support vector
- $a_i = 0$
  - Point is not a support vector

What if Data Are Not Linearly Separable?

- Cannot find $\boldsymbol{\theta}$ that satisfies $y_i(\boldsymbol{\theta}^\top \boldsymbol{x}_i) \geq 1, \forall i$
- Introduce the slack variable ξ_i

$$y_i(\boldsymbol{\theta}^\top \boldsymbol{x}_i) \geq 1 - \xi_i, \forall i \tag{23}$$

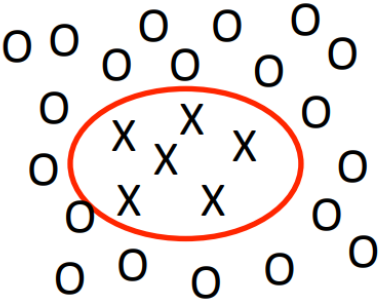
- New problem, s. t. $y_i(\boldsymbol{\theta}^\top \boldsymbol{x}_i) \geq 1 - \xi_i, \forall i$:

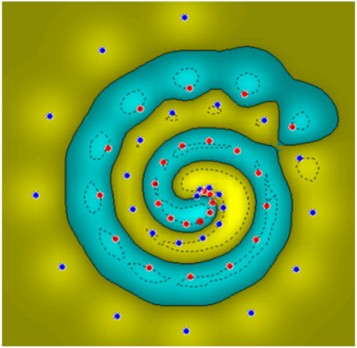
$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{j=1}^{d} \theta_j^2 + C \sum_i \xi_i \tag{24}$$

Strengths of SVMs

- Good generalization in theory
- Good generalization in practice
- Work well with few training instances
- Find the globally best model
- Efficient algorithms
- Amenable to the kernel trick ...

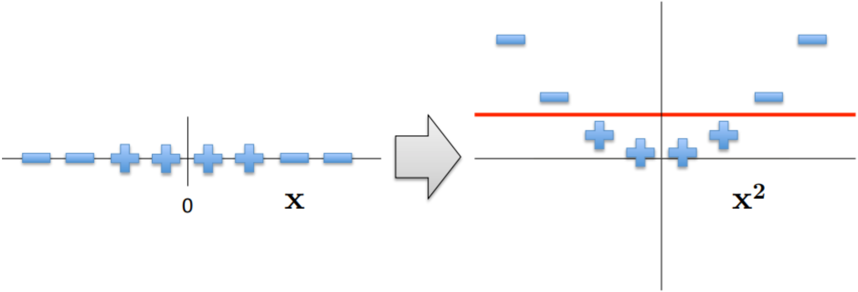
What is the Decision Boundary Is Not Linear?





Kernel Methods: Making the Non-Linear Linear

When Linear Separators Fail

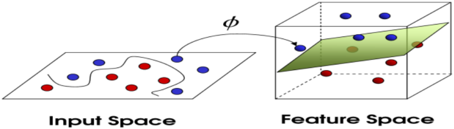


Mapping into a New Feature Space

- For example, with $\boldsymbol{x}_i \in \mathbb{R}^2$:

$$\Phi([x_{i1}, x_{i2}]) = [x_{i1}, x_{i2}, x_{i1}x_{i2}, x_{i1}^2, x_{i2}^2] \tag{25}$$

- Rather than running SVM on $\boldsymbol{x}_i$, run it on $\Phi(\boldsymbol{x}_i)$
  - Find non-linear separator in input space
- What if $\Phi(\boldsymbol{x}_i)$ is really big?
- Use kernels to compute it implicitly!



$$\Phi : \mathcal{X} \rightarrow \hat{\mathcal{X}} = \Phi(\boldsymbol{x}) \tag{26}$$

Kernels

- Find kernels K such that:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle \tag{27}$$

- Compute $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ should be efficient, much more so than computing $\Phi(\boldsymbol{x}_i)$ and $\Phi(\boldsymbol{x}_j)$
- Use $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in the SVM algorithm rather than $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$

The Polynomial Kernel

- Let $\boldsymbol{x}_i = [x_{i1}, x_{i2}]$ and $\boldsymbol{x}_j = [x_{j1}, x_{j2}]$
- Consider the following function:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle^2 \tag{28}$$

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (x_{i1}x_{j1} + x_{i2}x_{j2})^2 \tag{29}$$

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (x_{i1}^2 x_{j1}^2 + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2}) \tag{30}$$

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle \tag{31}$$

- where

$$\Phi(\boldsymbol{x}_i) = [x_{i1}^2, x_{i2}^2, \sqrt{2}x_{i1}x_{i2}] \tag{32}$$

$$\Phi(\boldsymbol{x}_j) = [x_{j1}^2, x_{j2}^2, \sqrt{2}x_{j1}x_{j2}] \tag{33}$$

The Kernel Trick

- Given an algorithm that is formulated in terms of a positive definite kernel K_1, one can construct an alternative algorithm by replacing K_1 with another positive definite kernel K_2
- SVMs can use the kernel trick

Incorporating Kernels into SVMs

- Originally we have:

$$\mathcal{J}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \tag{34}$$

  Such that $\sum_i a_j y_j = 0$, s.t. $\alpha_i \geq 0, \forall i$

- After we incorporate the kernel, it becomes:

$$\mathcal{J}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{35}$$

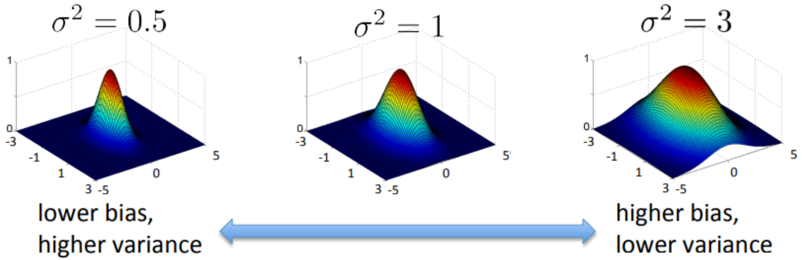
  Such that $\sum_i a_j y_j = 0$, s.t. $\alpha_i \geq 0, \forall i$

The Gaussian Kernel

- Also called Radial Basis Function (RBF) kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2}{2\sigma^2}\right) \tag{36}$$

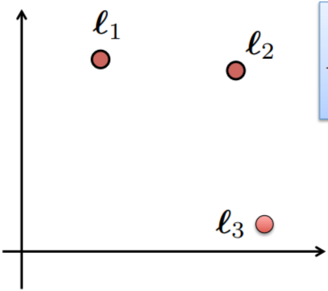
  - Has value 1 when $\boldsymbol{x}_i = \boldsymbol{x}_j$
  - Value falls off to 0 with increasing distance
  - Note: Need to do feature scaling before using the Gaussian kernel



The Gaussian Kernel: An Example

- Assume that we want to predict +1 or positive if:

$$\theta_0 + \theta_1 K(\boldsymbol{x}, \boldsymbol{\ell}_1) + \theta_2 K(\boldsymbol{x}, \boldsymbol{\ell}_2) + \theta_3 K(\boldsymbol{x}, \boldsymbol{\ell}_3) \geq 0 \tag{37}$$



$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right)$$

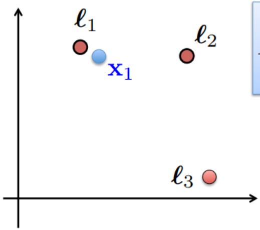
Imagine we've learned that:

$$\boldsymbol{\theta} = [-0.5, 1, 1, 0]$$

The Gaussian Kernel: An Example

- Assume that we want to predict +1 or positive if:

$$\theta_0 + \theta_1 K(\boldsymbol{x}, \boldsymbol{\ell}_1) + \theta_2 K(\boldsymbol{x}, \boldsymbol{\ell}_2) + \theta_3 K(\boldsymbol{x}, \boldsymbol{\ell}_3) \geq 0 \tag{38}$$



$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right)$$

Imagine we've learned that:

$$\boldsymbol{\theta} = [-0.5, 1, 1, 0]$$

- for $\boldsymbol{x}_1$, we have $K(\boldsymbol{x}_1, \ell_1) \approx 1$, other similarities ≈ 0

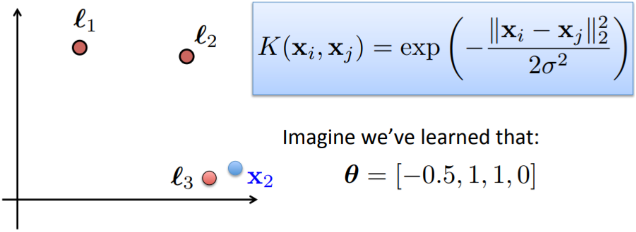
$$\theta_0 + \theta_1(1) + \theta_2(0) + \theta_3(0) = 0.5 \geq 0 \tag{39}$$

- so, predict +1 or positive

The Gaussian Kernel: An Example

- Assume that we want to predict +1 or positive if:

$$\theta_0 + \theta_1 K(\boldsymbol{x}, \boldsymbol{\ell}_1) + \theta_2 K(\boldsymbol{x}, \boldsymbol{\ell}_2) + \theta_3 K(\boldsymbol{x}, \boldsymbol{\ell}_3) \geq 0 \tag{40}$$



- for $\boldsymbol{x}_2$, we have $K(\boldsymbol{x}_2, \ell_3) \approx 1$, other similarities ≈ 0

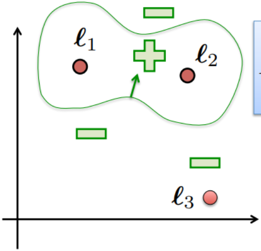
$$\theta_0 + \theta_1(0) + \theta_2(0) + \theta_3(1) = -0.5 \leq 0 \tag{41}$$

- so, predict -1 or negative

The Gaussian Kernel: An Example

- Assume that we want to predict +1 or positive if:

$$\theta_0 + \theta_1 K(\boldsymbol{x}, \boldsymbol{\ell}_1) + \theta_2 K(\boldsymbol{x}, \boldsymbol{\ell}_2) + \theta_3 K(\boldsymbol{x}, \boldsymbol{\ell}_3) \geq 0 \tag{42}$$



$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right)$$

Imagine we've learned that:

$$\boldsymbol{\theta} = [-0.5, 1, 1, 0]$$

- Here's the graph sketch of the decision boundary when projected into the 2D space

Other Kernels

- Sigmoid Kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\alpha \boldsymbol{x}_i^\top \boldsymbol{x}_j + c) \tag{43}$$

  - Neural networks use sigmoid as an activation function
  - SVM with a sigmoid kernel is equivalent to a 2-layer perceptron
- Cosine Similarity Kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\boldsymbol{x}_i^\top \boldsymbol{x}_j}{||\boldsymbol{x}_i||||\boldsymbol{x}_j||} \tag{44}$$

  - Popular choice for measuring the similarity of text documents
  - L^2 norm projects vectors onto the unit sphere; their dot product is the cosine of the angle between the vectors

Other Kernels

- Chi-squared Kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma \sum_k \frac{(x_{ik} - x_{jk})^2}{x_{ik} + xjk}\right) \tag{45}$$

  - Widely used in computer vision applications
  - Chi-squared measures the distance between probability distributions
  - Data is issued to be non-negative, often with L^1 norm
- String kernels
- Tree kernels
- Graph kernels

Conclusion

- The SVM finds the optimal linear separator
- The kernel trick makes SVMs learn non-linear decision surfaces
- Strengths of SVMs:
  - Good theoretical and empirical performance
  - Supports many types of kernels
- Weaknesses of SVMs:
  - "Slow" to train and predict for huge datasets (although relatively fast...)
  - The kernel needs to be wisely chosen and its parameters need to be tuned