DS 4400

Machine Learning and Data Mining I
Spring 2021

Alina Oprea
Associate Professor
Khoury College of Computer Science
Northeastern University

March 16 2021

Announcements

- Midterm exams have been graded
- HW 4 is due next Friday, March 26
- Project milestone due on March 31
  - Template in Gradescope
- Final exam on Tuesday, April 6
  - Review on Thursday, April 1

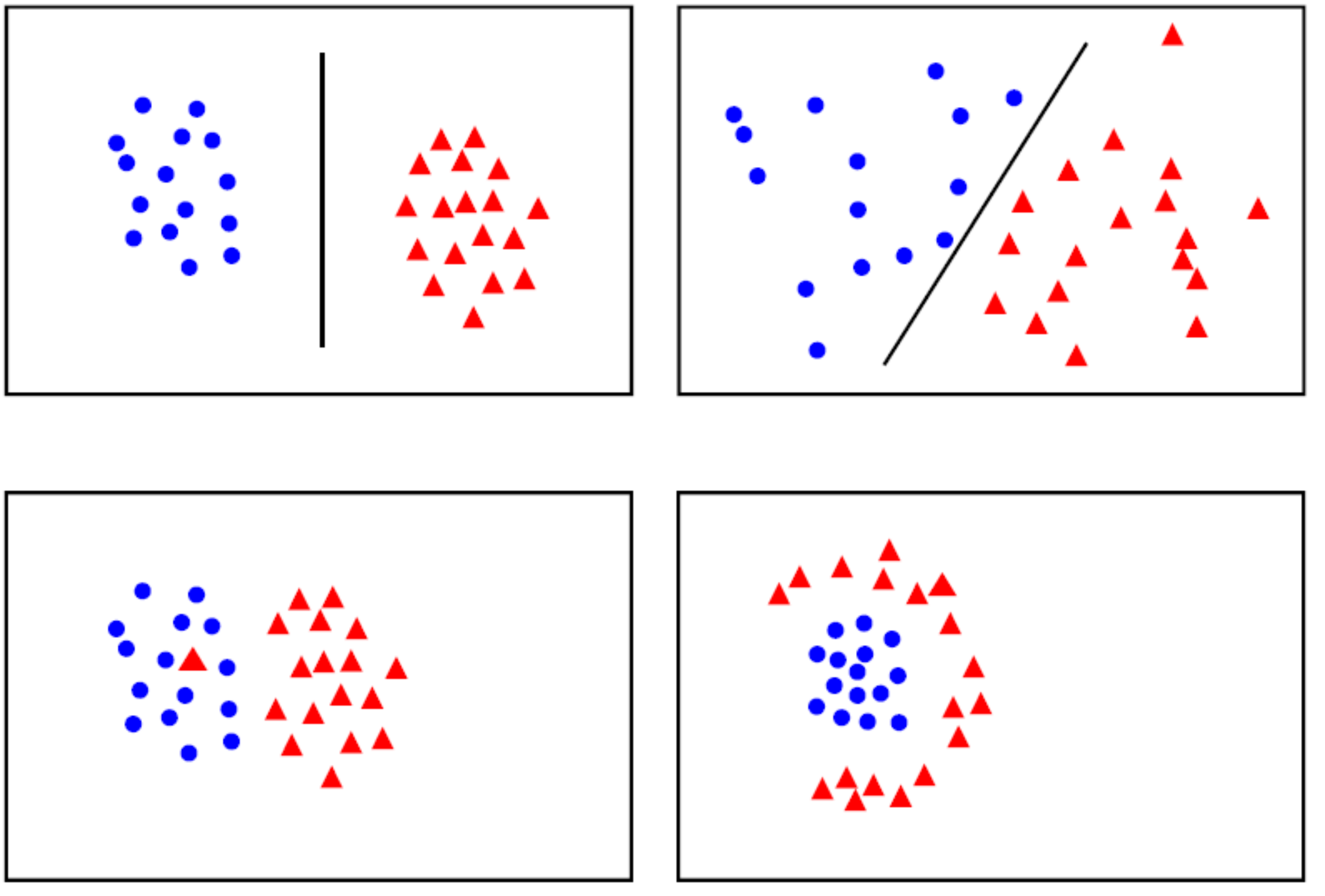
Outline

- Support Vector Machines
  - Non-linearly separable data
    - Support vector classifier
- Deep Learning
  - Motivation
  - Goals
- Deep Learning as representation learning
- Perceptron and its limitations

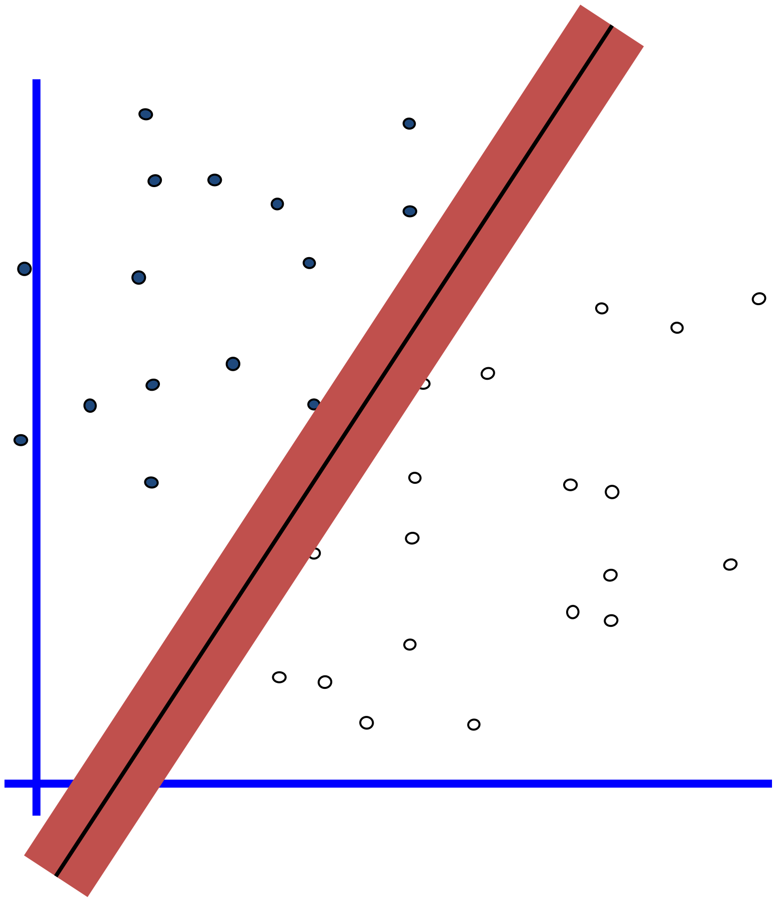
Linear separability

linearly separable

not linearly separable



Maximum Margin



Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a data point.

Choose the maximum margin linear classifier: the linear classifier with the maximum margin.

Maximum margin classifier

- Training data $x_1, \dots, x_N$ with $x_i = \left(x_{i1}, \dots, x_{id}\right)^{\mathrm{T}}$
- Labels are from 2 classes: $y_i \in \{-1,1\}$

maximize M

$$y_i\left(\theta_0 + \theta_1 x_{i1} + \cdots \theta_d x_{id}\right) \geq M \ \forall i$$

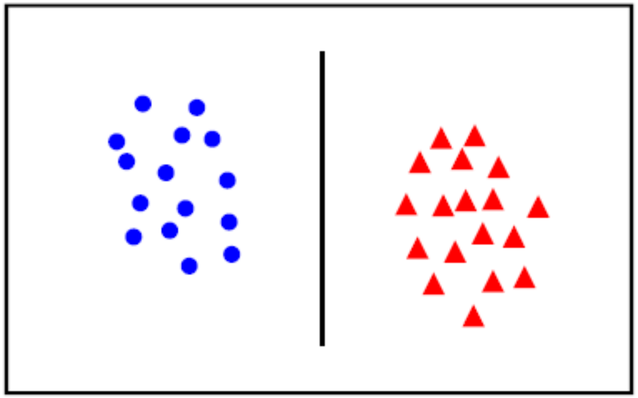
$$\left|\left|\theta\right|\right|_2 = 1$$

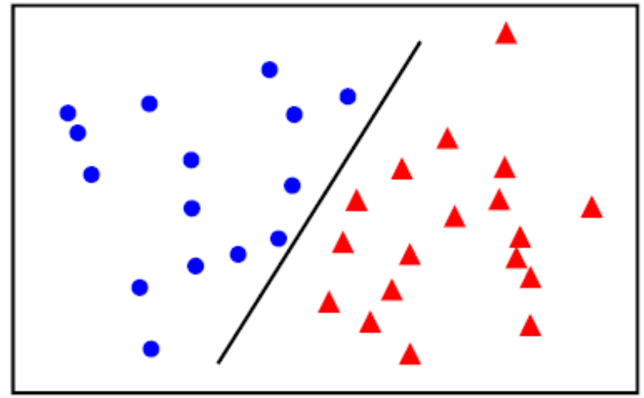
Normalization constraint (to have unique solution)

Each point is on the right side of hyper-plane at distance $\geq M$

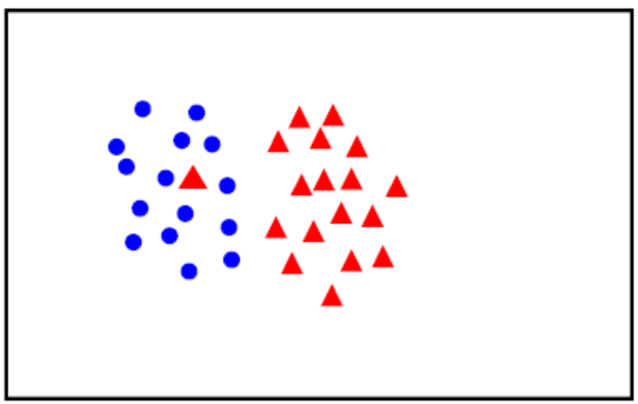
Linear separability

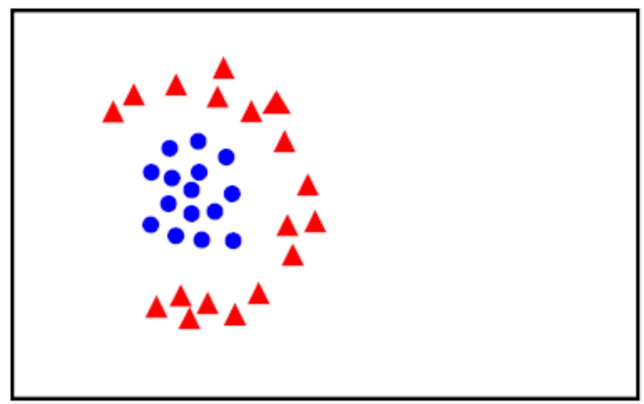
linearly
separable





not
linearly
separable
(but almost)





Support vector classifier

- Allow for small number of mistakes on training data
- Soft margin classifier

max M

$$y_i(\theta_0 + \theta_1 x_{i1} + \cdots \theta_d x_{id}) \geq M(1 - \epsilon_i)\forall i$$

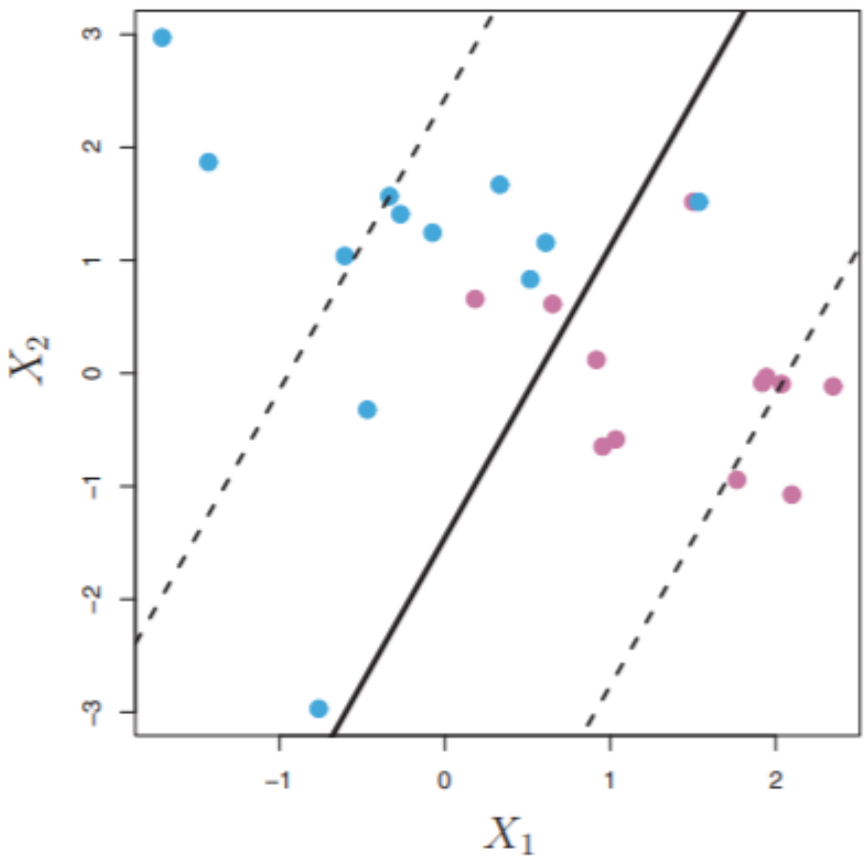
$$||\theta||_2 = 1$$

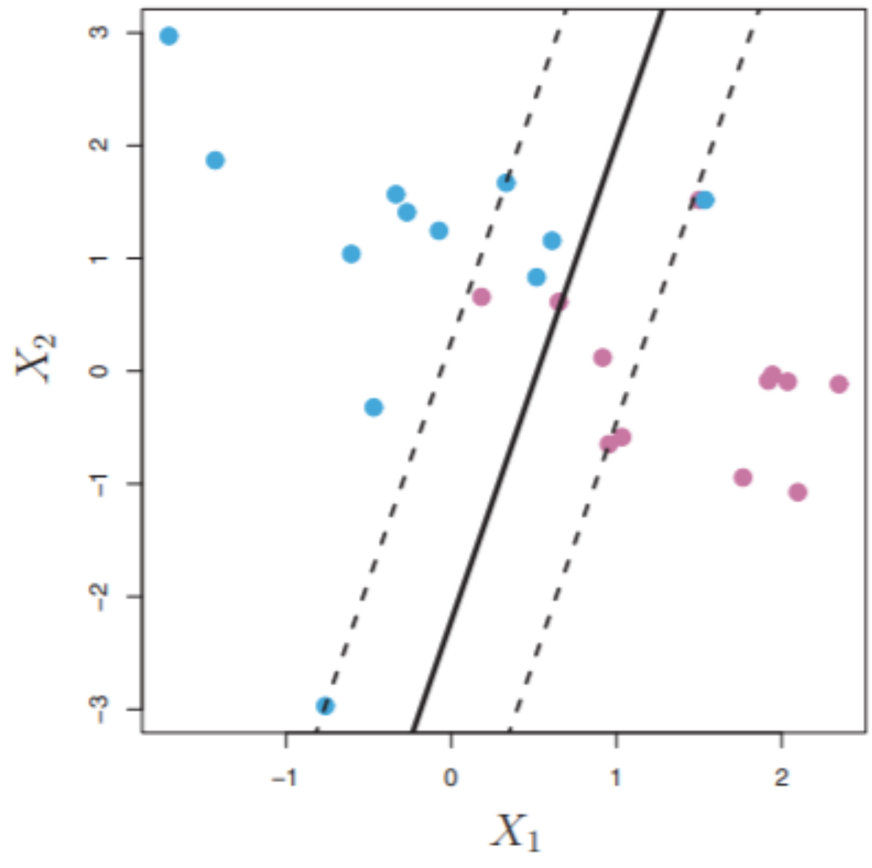
$$\epsilon_i \geq 0, \sum_i \epsilon_i \leq C$$

Slack

Error Budget (Hyper-parameter)

Support vectors





Support vectors: all points within the margin of the classifier

Support vector classifier

- Just like in separable case, gives solution of the form:

$$f(z) = \theta_0 + \sum_i \alpha_i < z, x_i >$$

Where $\alpha_i \neq 0$ for support vectors (and $\alpha_i = 0$ for all other training points)

- This model is called
  - Support Vector Classifier (SVC)
  - Linear SVM
  - Soft-margin classifier

Logistic Regression

$$J(\theta) = -\sum_{i=1}^{N}[y_{\mathrm{i}}\log h_\theta(x_i) + (1 - y_i)\log\left(1 - h_\theta(x_i)\right)]$$

- Cost of a single instance:

$$\text{cost}\,(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

- Can re-write objective function as

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n} \underbrace{\text{cost}\left(h_\theta(x_i), y_i \right)}_{\text{Cross-entropy loss}}$$

Regularized Logistic Regression

$$J(\theta) = -\sum_{i=1}^{N} [y_{\mathrm{i}} \log h_\theta(x_i) + (1 - y_i) \log \left(1 - h_\theta(x_i)\right)]$$

- We can regularize logistic regression exactly as before:

$$\begin{aligned} J_{\text{regularized}}(\boldsymbol{\theta}) &= J(\boldsymbol{\theta}) + \lambda \sum_{j=1}^{d} \theta_j^2 \\ &= J(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}_{[1:d]}\|_2^2 \end{aligned}$$

L2 regularization

Hinge Loss

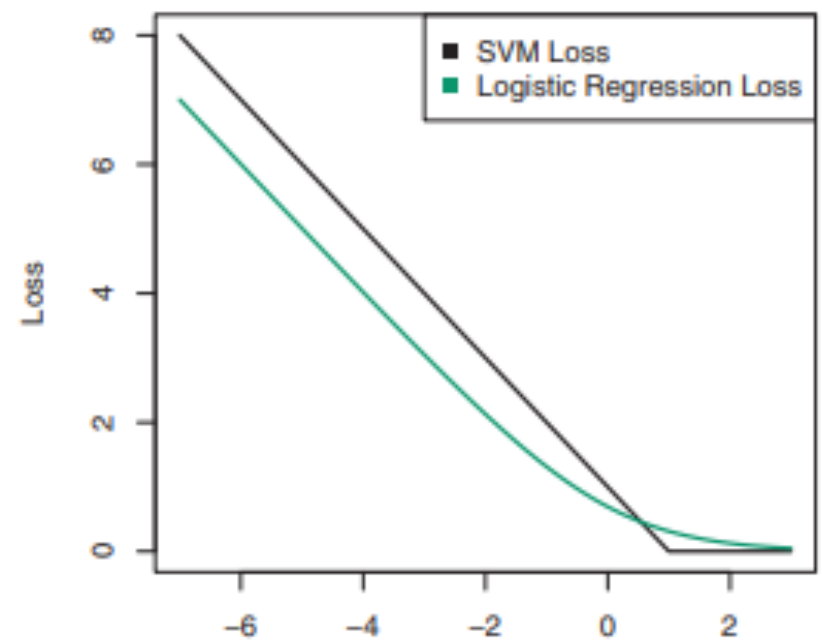
- Linear SVM: $h_\theta(x_i) = \theta^T x_i$
- Optimization solution equivalent to:
- $J(\theta) = \sum_{i=0}^{N} \max\big(0, 1 - y_i h_\theta(x_i)\big) + \lambda \sum_{j=1}^{d} \theta_j^2$

- $J(\theta) = C \sum_{i=0}^{N} \max\big(0, 1 - y_i h_\theta(x_i)\big) + \sum_{j=1}^{d} \theta_j^2$

$C = \text{regularization cost}$

Connection to Logistic Regression

- Logistic regression
  - Cross-entropy loss
- SVM
  - Hinge loss



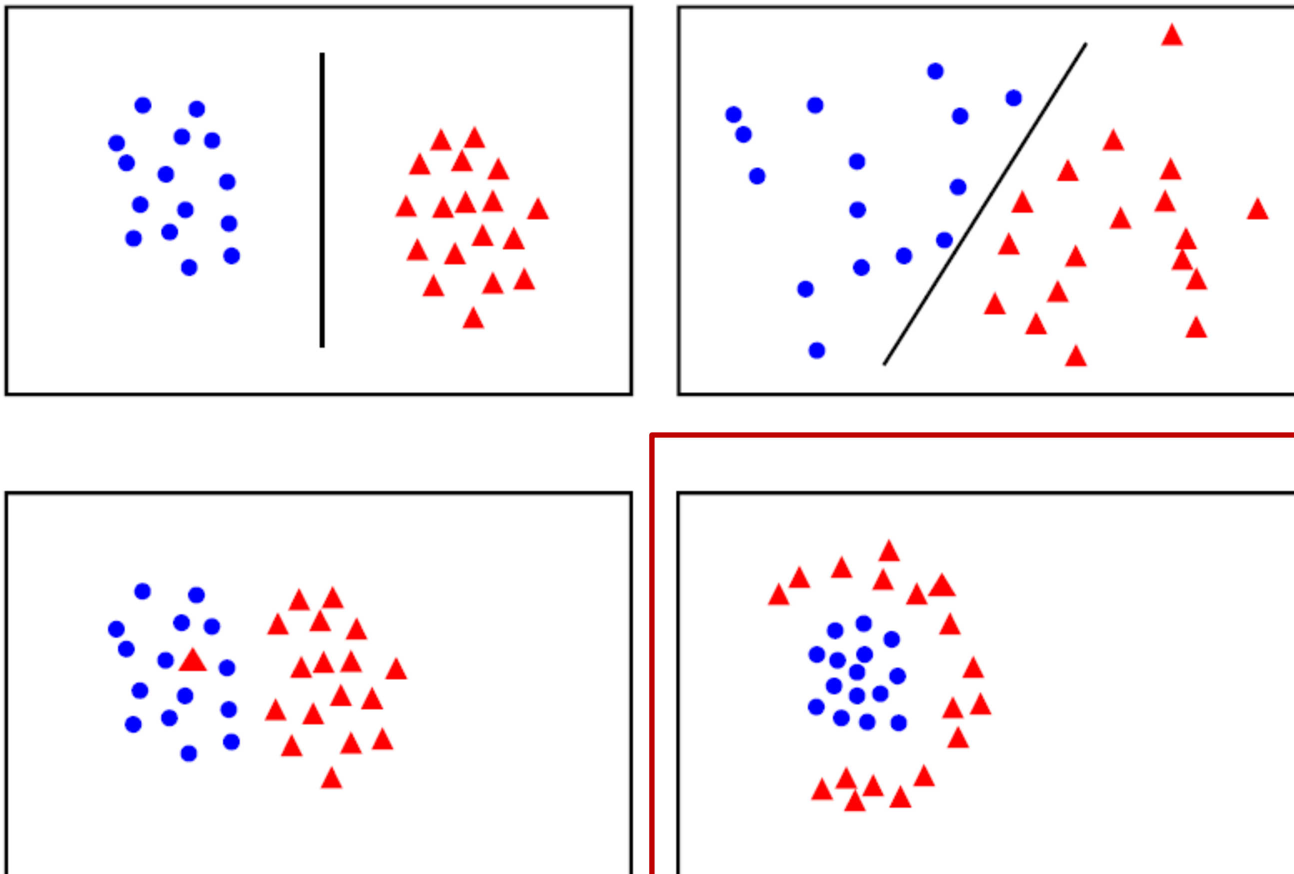
$y_i(\theta^T x_i)$

Linear separability

linearly separable

not linearly separable

(but almost)



(not even close!)

Non-linear decision

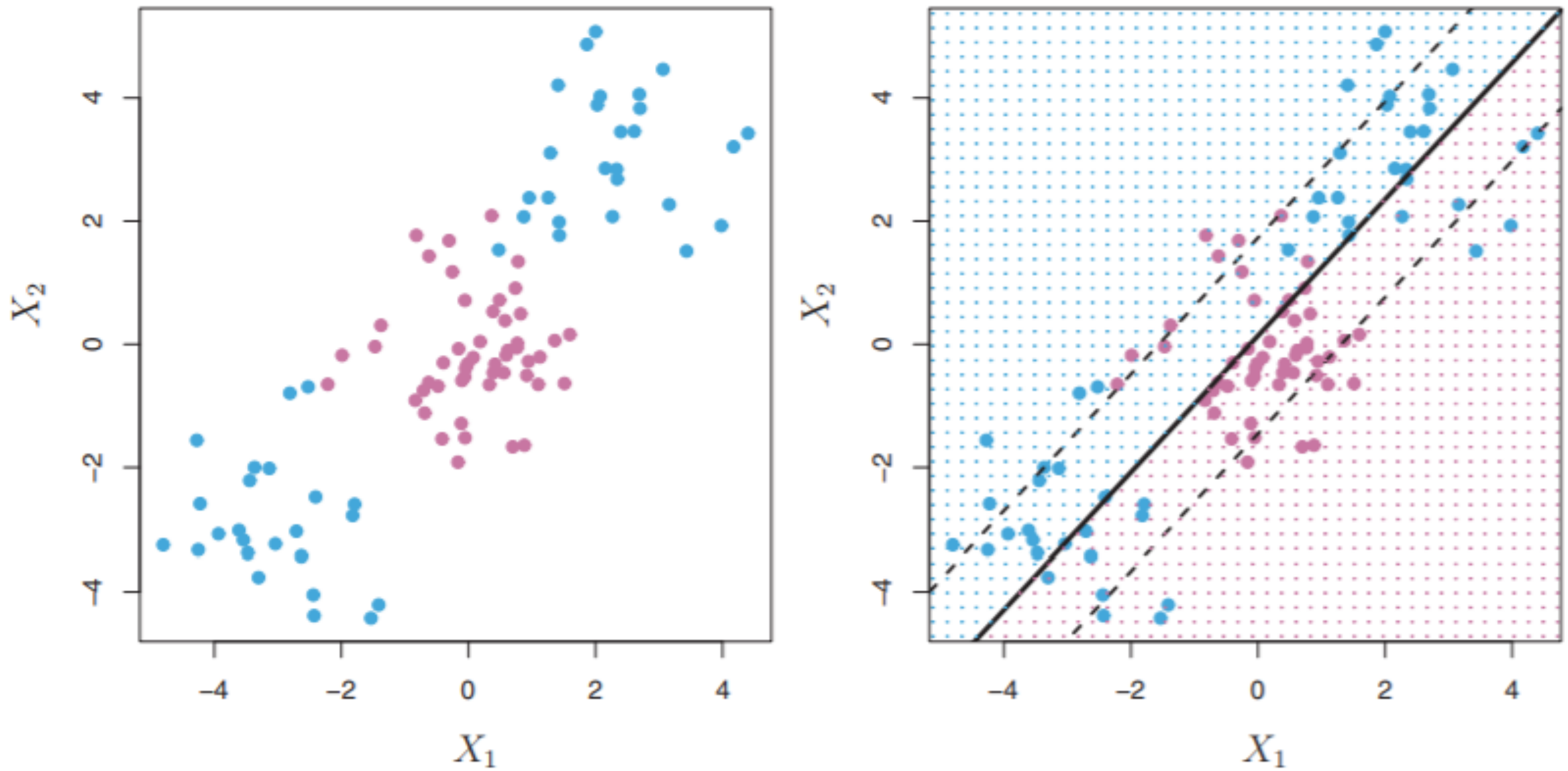
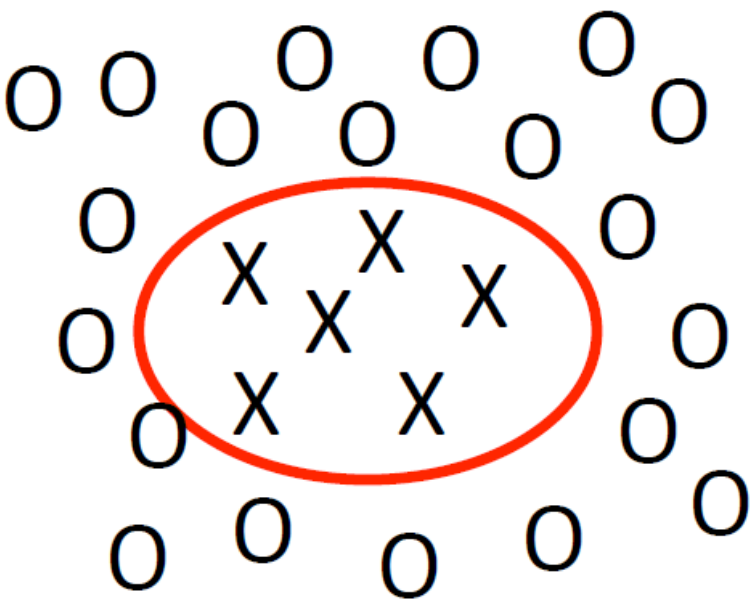


FIGURE 9.8. Left: *The observations fall into two classes, with a non-linear boundary between them.* Right: *The support vector classifier seeks a linear boundary, and consequently performs very poorly.*

More examples



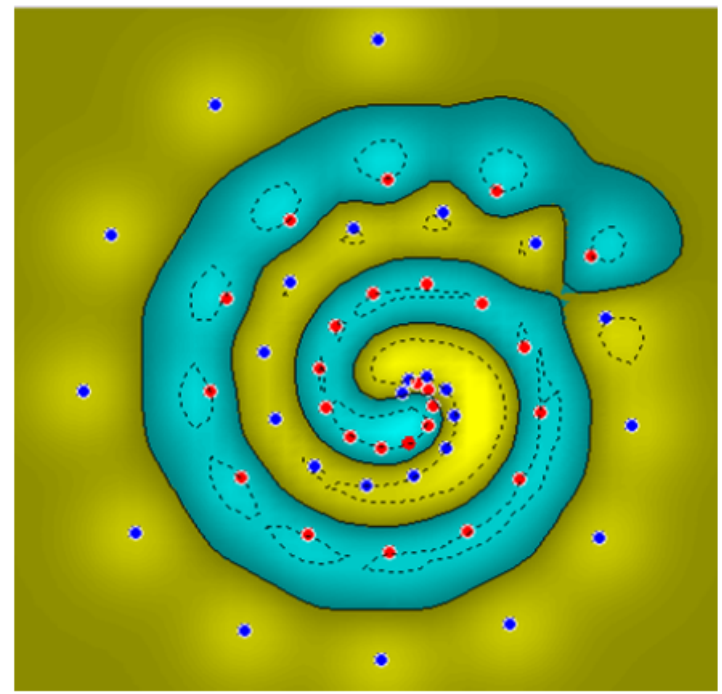


Image from http://www.atrandomresearch.com/iclass/

Kernels

- Support vector classifier
  - $\mathrm{h}(z) = \theta_0 + \sum_{i \in S} \alpha_i < z, x_i >$
    $= \theta_0 + \sum_{i \in S} \alpha_i \sum_{j=1} z_j x_{ij}$
  - S is set of support vectors
  - Replace with $\mathrm{h}(z) = \theta_0 + \sum_{i \in S} \alpha_i K(z, x_i)$

Any kernel function!

- What is a kernel?
  - Function that characterizes similarity between 2 observations
  - $K(x, y) =< x, y > = \sum_j x_i y_j$ linear kernel!
  - The closer the points, the larger the kernel
- Intuition
  - The closest support vectors to the point play larger role in classification

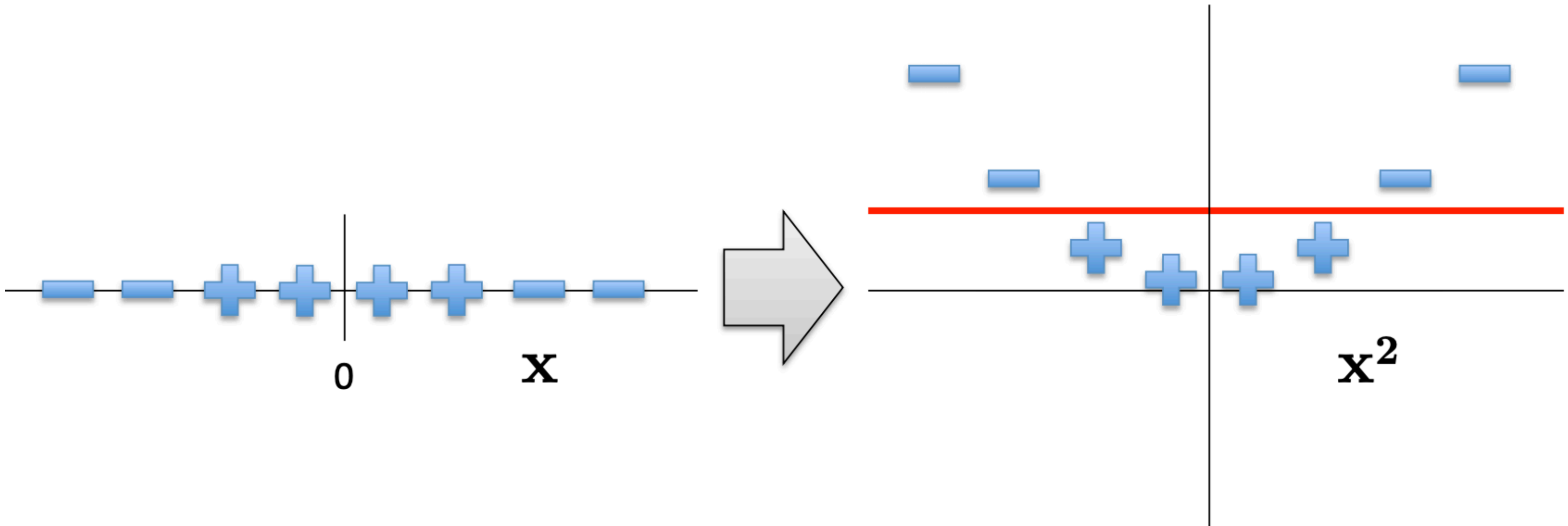
The Kernel Trick

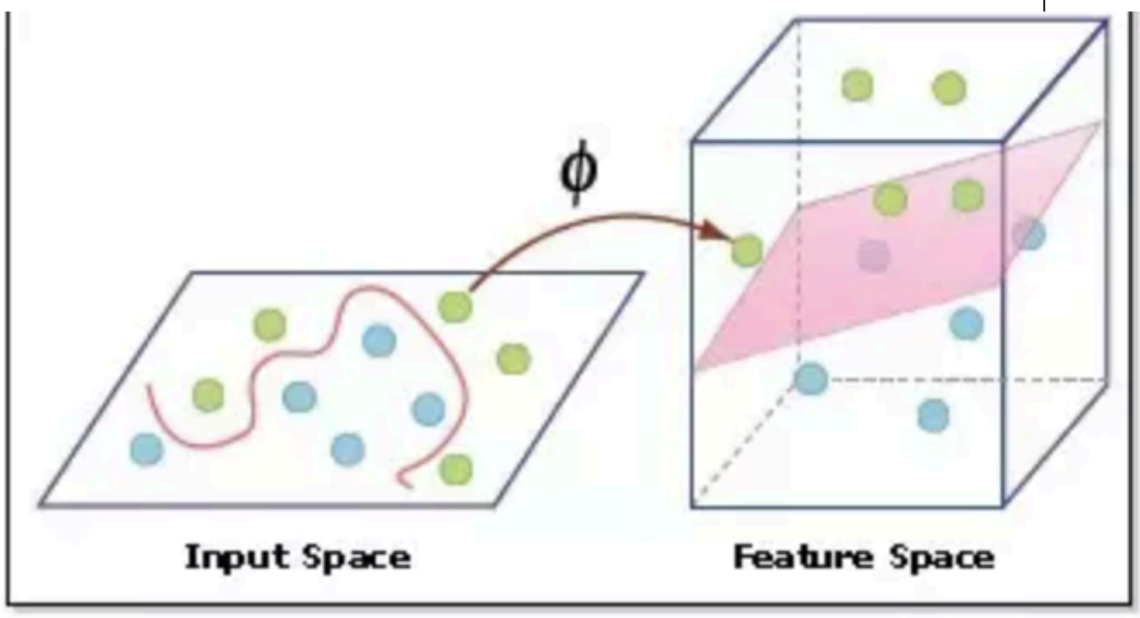
"Given an algorithm which is formulated in terms of a positive definite kernel K_1, one can construct an alternative algorithm by replacing K_1 with another positive definite kernel K_2"

- SVMs can use the kernel trick

- Enlarge feature space
- Shape of the kernel changes the decision boundary

Why Use Kernels





SVM Classifier

- Select a kernel function
  - The Gram matrix $G_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$
    - Symmetric matrix
    - Positive semi-definite matrix:

      $\mathbf{z}^{\mathsf{T}}\mathrm{G}\mathbf{z} \geq 0$ for every non-zero vector $\mathbf{z} \in \mathbb{R}^n$

- Final SVM classifier is linear combination of kernel between testing point and support vectors
  - $h(z) = \theta_0 + \sum_{i \in S} \alpha_i K(z, x_i)$

Kernels

- Linear kernels
  - $K(x, y) = < x, y > = \sum_i x_i y_i$
- Polynomial kernel of degree *p*
  - $K(x, y) = \left(1 + \sum_{i=0}^{d} x_i y_i\right)^p$
- Radial Basis Function (RBF) kernel (or Gaussian)
  - $K(x, y) = \exp\left(-\sum_{i=0}^{d} (x_i - y_i)^2 / 2\sigma^2\right)$

Examples of SVM classifiers

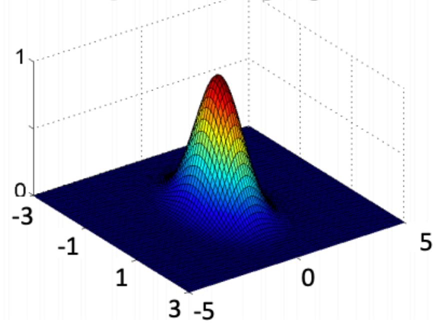
- Notation
  - S = index of support vectors
  - $\{x_i\}, i \in S$ = set of support vectors
- SVM with polynomial kernel
  - $h(z) = \theta_0 + \sum_{i \in S} \alpha_i \left(1 + \sum_{j=0}^{d} z_j x_{ij}\right)^p$
  - Hyper-parameter p (degree of polynomial)
- SVM with Gaussian / radial kernel
  - $h(z) = \theta_0 + \sum_{i \in S} \alpha_i e^{-\sum_{j=0}^{d} (z_j - x_{ij})^2 / 2\sigma^2}$
  - Hyper-parameter σ

Gaussian / Radial Kernel

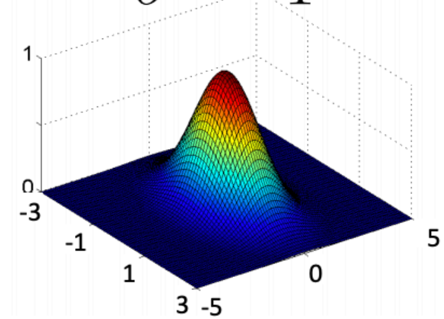
$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right)$$

- Has value 1 when $\mathbf{x}_i = \mathbf{x}_j$
- Value falls off to 0 with increasing distance
- Note: Need to do feature scaling <u>before</u> using Gaussian Kernel

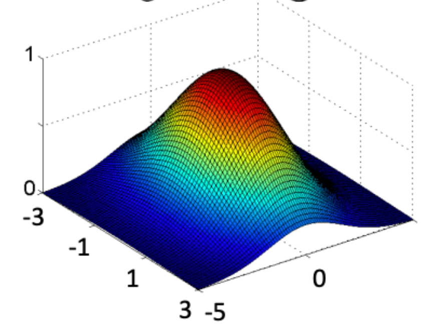
$\sigma^2 = 0.5$



$\sigma^2 = 1$



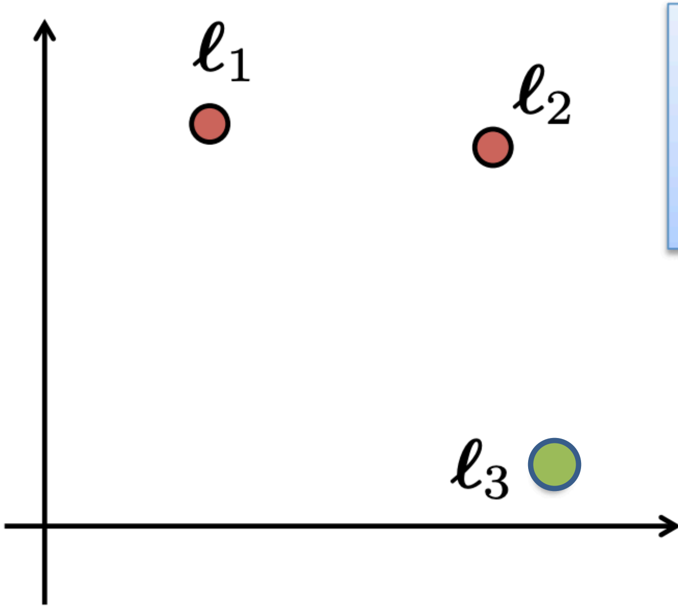
$\sigma^2 = 3$



lower bias,
higher variance

higher bias,
lower variance

Gaussian / Radial Kernel Example



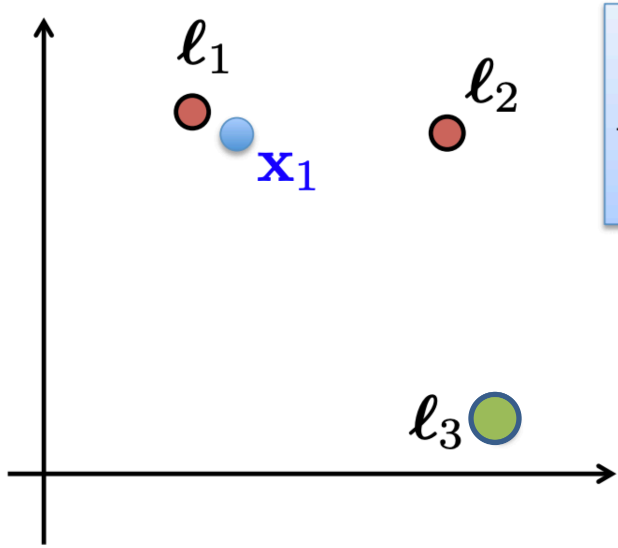
$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right)$$

Imagine we've learned that:

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

Predict +1 if $\theta_0 + \theta_1 K(\mathbf{x}, \boldsymbol{\ell}_1) + \theta_2 K(\mathbf{x}, \boldsymbol{\ell}_2) + \theta_3 K(\mathbf{x}, \boldsymbol{\ell}_3) \geq 0$

Gaussian / Radial Kernel Example



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

Imagine we've learned that:

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

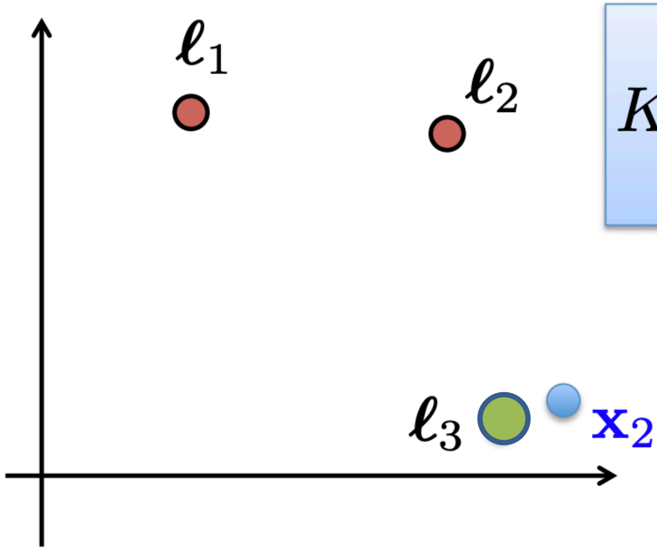
Predict +1 if $\theta_0 + \theta_1 K(\mathbf{x}, \boldsymbol{\ell}_1) + \theta_2 K(\mathbf{x}, \boldsymbol{\ell}_2) + \theta_3 K(\mathbf{x}, \boldsymbol{\ell}_3) \geq 0$

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

$$\begin{aligned} &\theta_0 + \theta_1(1) + \theta_2(0) + \theta_3(0) \\ &= -0.5 + 1(1) + 1(0) + 0(0) \\ &= 0.5 \geq 0 \end{aligned}$$, so predict +1

Based on example by Andrew Ng

Gaussian / Radial Kernel Example



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

Imagine we've learned that:

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

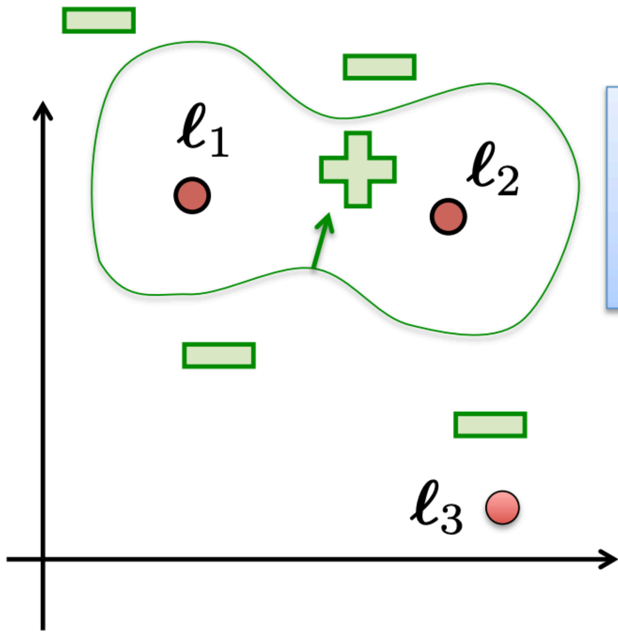
Predict +1 if $\theta_0 + \theta_1 K(\mathbf{x}, \boldsymbol{\ell}_1) + \theta_2 K(\mathbf{x}, \boldsymbol{\ell}_2) + \theta_3 K(\mathbf{x}, \boldsymbol{\ell}_3) \geq 0$

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

$$\theta_0 + \theta_1(0) + \theta_2(0) + \theta_3(1)$$
$$= -0.5 + 1(0) + 1(0) + 0(1)$$
$$= -0.5 < 0$$, so predict -1

Based on example by Andrew Ng

Gaussian / Radial Kernel Example



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

Imagine we've learned that:

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

Predict +1 if $\theta_0 + \theta_1 K(\mathbf{x}, \boldsymbol{\ell}_1) + \theta_2 K(\mathbf{x}, \boldsymbol{\ell}_2) + \theta_3 K(\mathbf{x}, \boldsymbol{\ell}_3) \geq 0$

Rough sketch of decision surface

Kernel Example

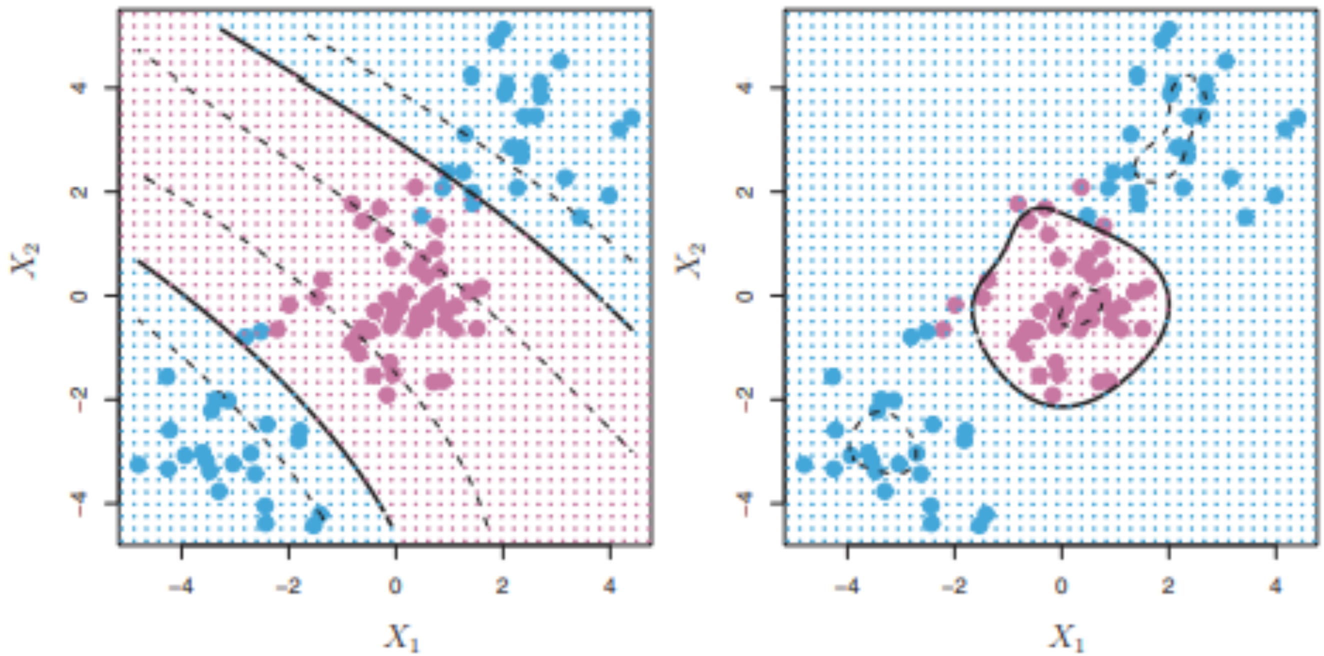


FIGURE 9.9. Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

Advantages of Kernels

- Generate non-linear features
- More flexibility in decision boundary
- Generate a family of SVM classifiers
- Testing is computationally efficient
  - Cost depends only on support vectors and kernel operation

- Disadvantages
  - Kernels need to be tuned (additional hyper-parameters)

When to use different kernels?

- If data is (close to) linearly separable, use linear SVM
- Radial or polynomial kernels preferred for non-linear data
- Training radial or polynomial kernels takes longer than linear SVM
- Other kernels
  - Sigmoid
  - Hyperbolic Tangent

Kernels in ML

- Kernel ridge regression
  - Non-linear regression method
- Kernel Density Estimate (KDE)
  - Unsupervised method for learning density estimation using kernel functions
- Kernel PCA
  - Unsupervised learning for dimensionality reduction
- Kernel clustering
  - Create clusters of similar points
  - Similarity between points computed with kernel function

Review SVM

- SVMs find optimal linear separator
- The kernel trick makes SVMs learn non-linear decision surfaces

- Strength of SVMs:
  - Good theoretical and empirical performance
  - Supports many types of kernels

- Disadvantages of SVMs:
  - "Slow" to train/predict for huge data sets (but relatively fast!)
  - Need to choose the kernel (and tune its parameters)

Comparing SVM with other classifiers

- SVM is resilient to outliers
  - Similar to Logistic Regression
  - LDA or kNN are not
- SVM can be trained with Gradient Descent
  - Hinge loss cost function
- Supports regularization
  - Can add penalty term (ridge or Lasso) to cost function
- Linear SVM is most similar to Logistic Regression

Acknowledgements

- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag
  - Andrew Moore
  - Yann LeCun
- Thanks!