DS 4400

Machine Learning and Data Mining I

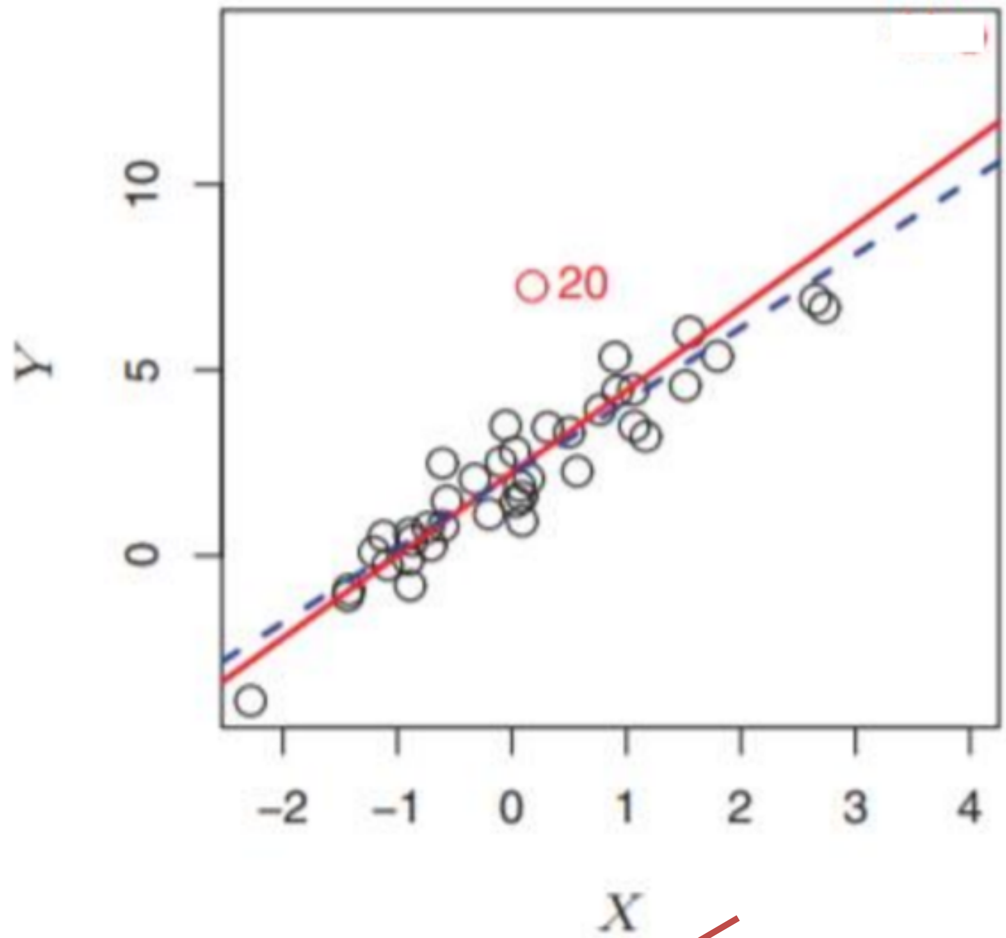
Alina Oprea
Guest lecture by Lisa Friedland
Postdoctoral Researcher, CCIS and NetSI
Northeastern University

February 26 2019

Outline

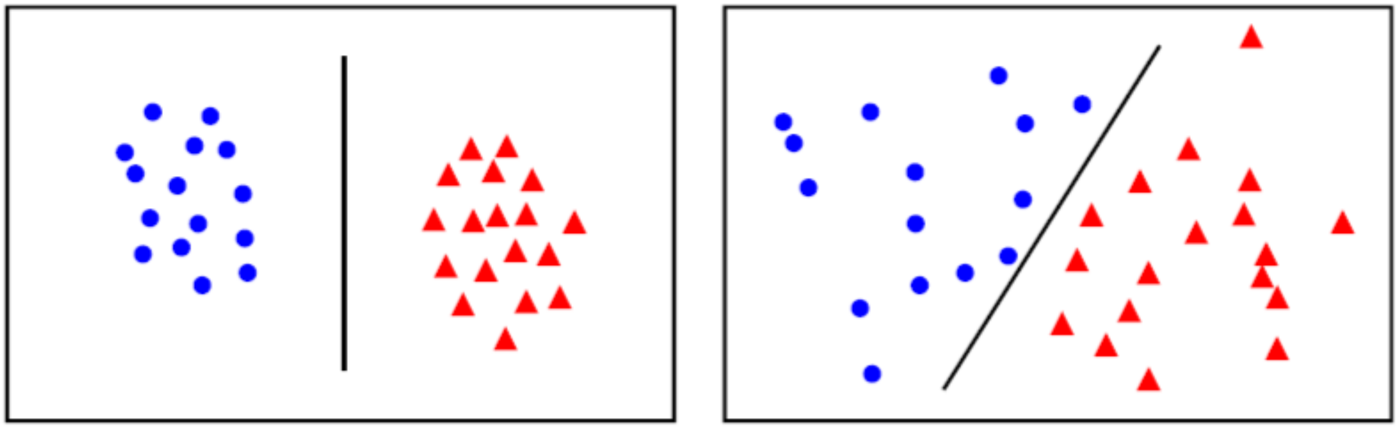
- Review of linear models
  - Separating hyperplanes
- Support Vector Machines
  - Linearly separable data
    - Maximum margin classifier
  - Non-separable data
    - Support vector classifier
  - Non-linear decision boundaries
    - Kernels and Radial SVM

Linear models we've seen



Linear regression

Linear models we've seen



Classifiers with linear decision boundary:
- Perceptron
- Logistic regression
- Linear discriminant analysis
- today: support vector classifier

Hyperplane

- Line (2-dimensions): $\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$
- Hyperplane (d-dimensions): $\theta_0 + \theta_1 x_1 + \cdots \theta_d x_d = 0$

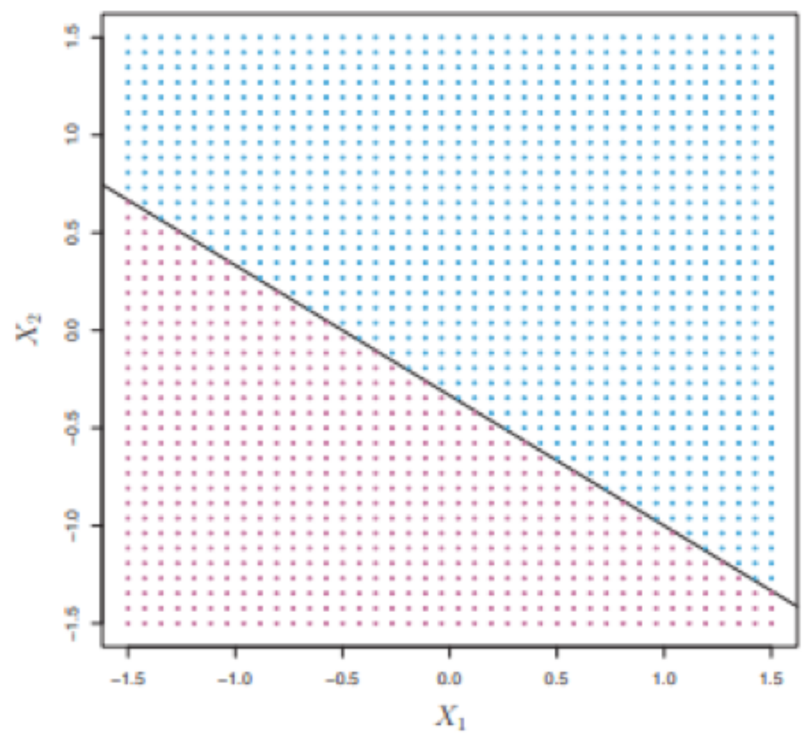


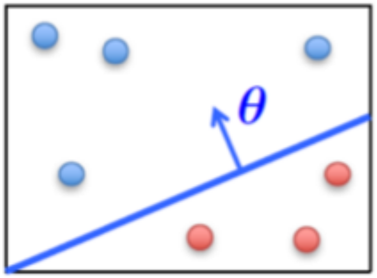
FIGURE 9.1. *The hyperplane $1 + 2X_1 + 3X_2 = 0$ is shown. The blue region is the set of points for which $1 + 2X_1 + 3X_2 > 0$, and the purple region is the set of points for which $1 + 2X_1 + 3X_2 < 0$.*

Recall:

Linear classifiers

- **Linear classifiers**: represent decision boundary by hyperplane

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad \boldsymbol{x}^\intercal = \begin{bmatrix} 1 & x_1 & \dots & x_d \end{bmatrix}$$



All the points x on the hyperplane satisfy: $\theta^T x = 0$

$$h(\boldsymbol{x}) = \text{sign}(\boldsymbol{\theta}^\intercal \boldsymbol{x}) \quad \text{where} \quad \text{sign}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

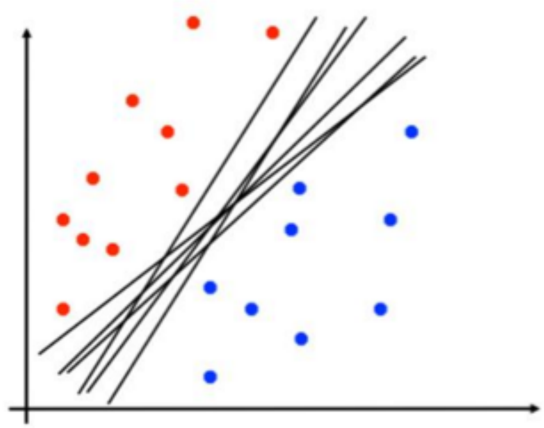
– Note that: $\boldsymbol{\theta}^\intercal \boldsymbol{x} > 0 \implies y = +1$

$$\boldsymbol{\theta}^\intercal \boldsymbol{x} < 0 \implies y = -1$$

Recall:

Perceptron Limitations

- Is dependent on starting point
- It could take many steps for convergence
- Perceptron can overfit
  – Move the decision boundary for every example



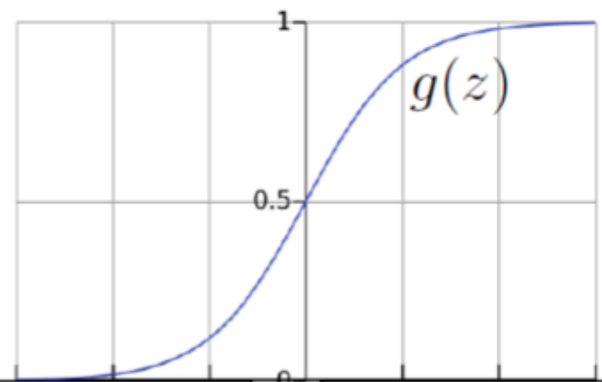
Which of this is optimal?

Recall logistic regression:

- Let $z = \theta^T x$ (a measure of x's distance from the decision boundary)
- $P(y = 1|x) = g(z)$ (Decision boundary tries to maximize probabilities assigned to correct answers)

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}\right)$$

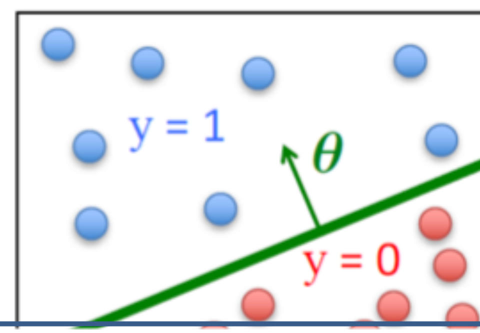
$$g(z) = \frac{1}{1+e^{-z}}$$



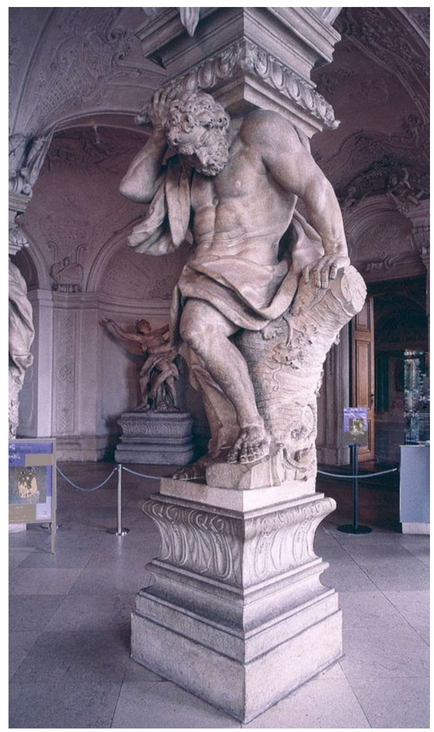
$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ should be large <u>negative</u> values for negative instances

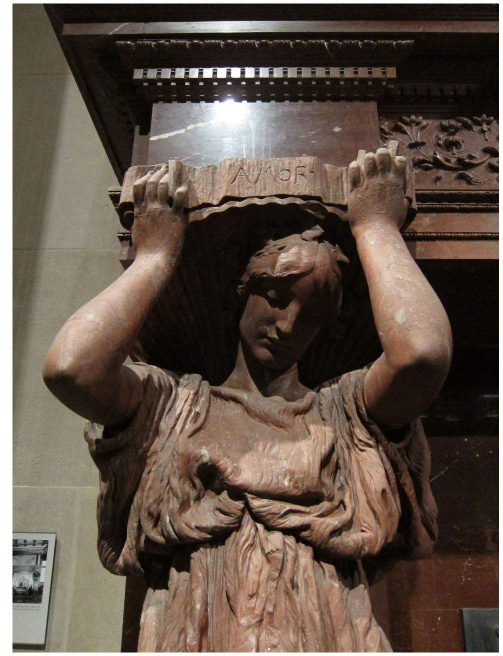
$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ should be large <u>positive</u> values for positive instances

- Assume a threshold and...
  - Predict y = 1 if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 0.5$
  - Predict y = 0 if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0.5$



Support vectors

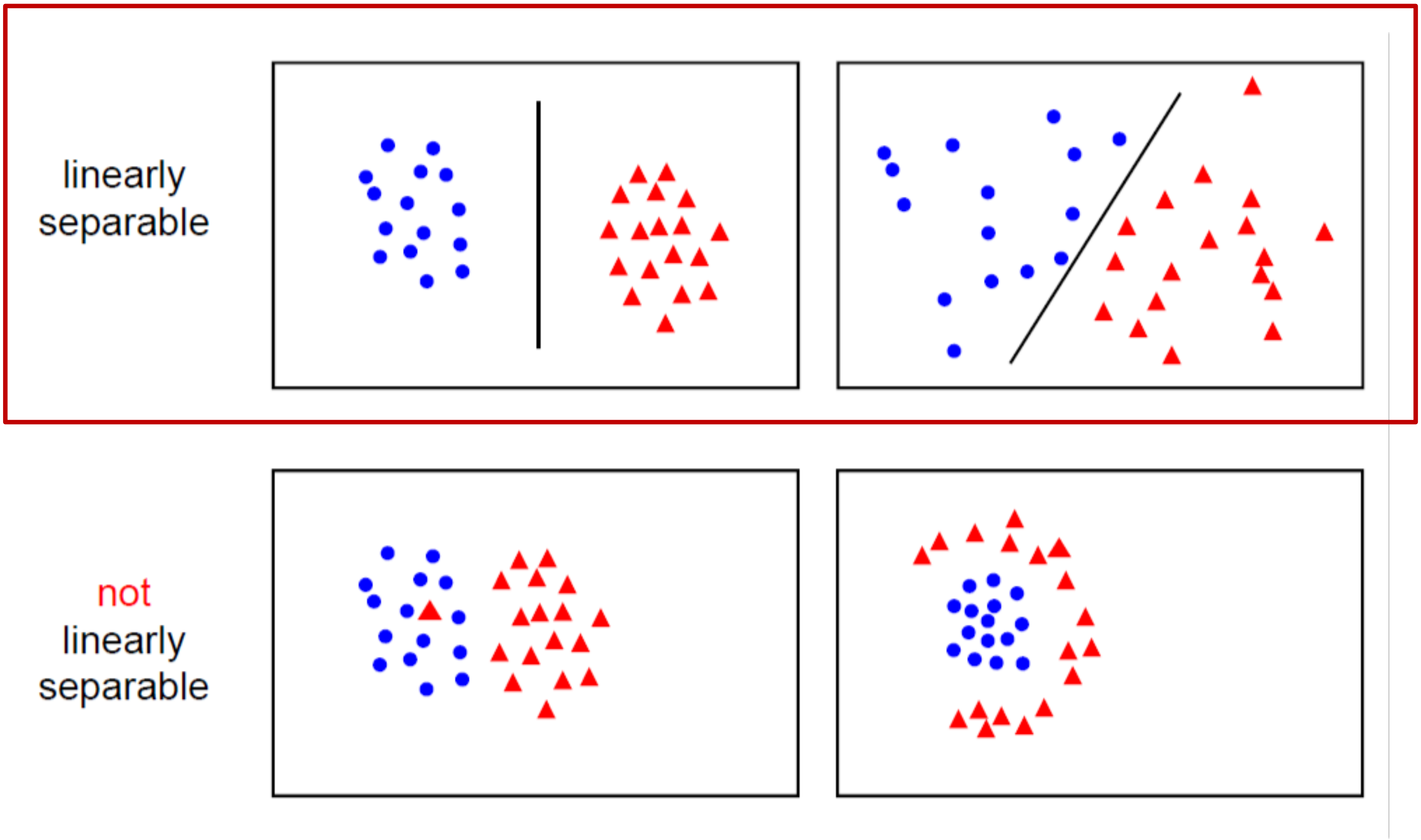




Outline

- Review of linear classifiers
  - Separating hyperplanes
- Support Vector Machines
  - Linearly separable data
    - Maximum margin classifier
  - Non-separable data
    - Support vector classifier
  - Non-linear decision boundaries
    - Kernels and Radial SVM

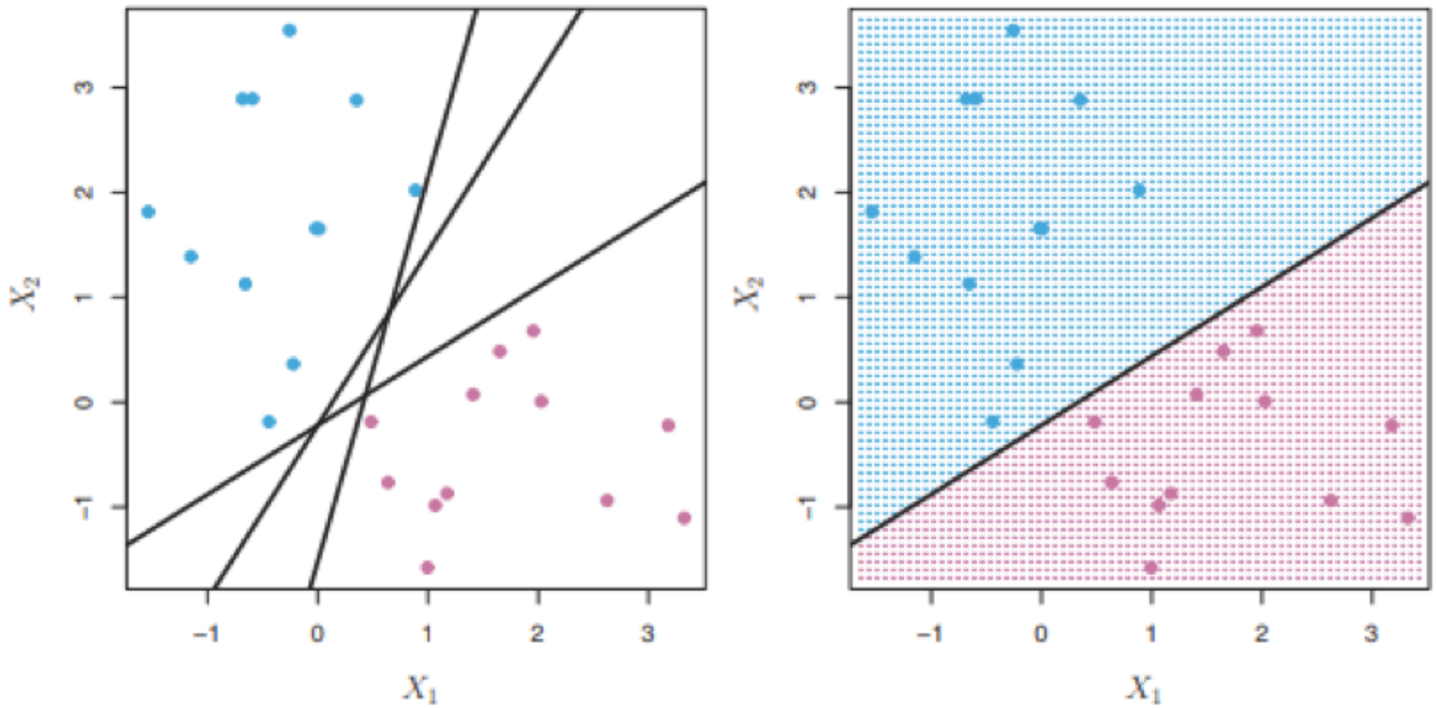
Linear separability



Notation (supervised learning)

- Training data $x^{(1)}, \ldots, x^{(n)}$ with $x^{(i)} = \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)^{\mathrm{T}}$
- Labels are from 2 classes: $y^{(i)} \in \{-1,1\}$
- Goal:
  - Build a model to classify training data
  - Test it on new vector x' = $({x'}_1, \ldots, {x'}_d)^T$ to predict label y'

Separating hyperplane



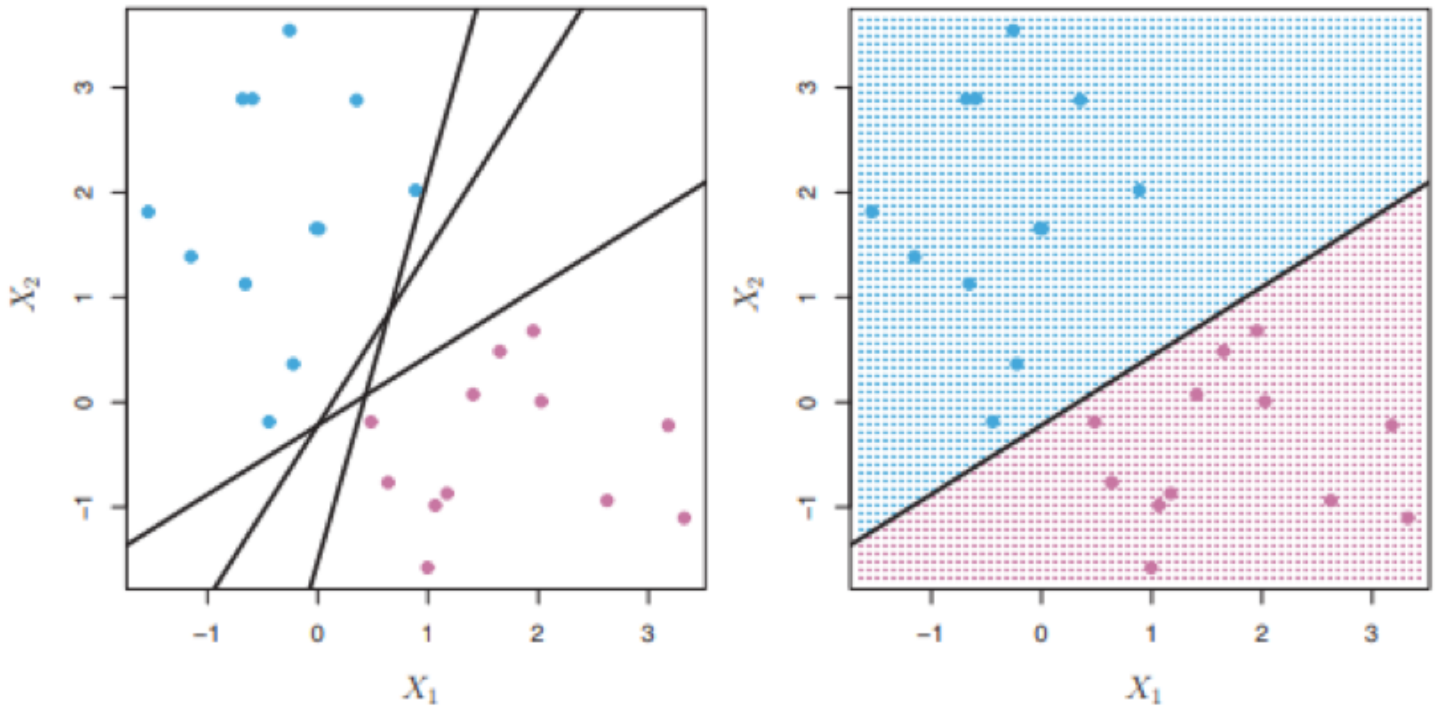
$$\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)} > 0 \;\text{ if } y^{(i)} = 1$$

$$\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)} < 0 \;\text{ if } y^{(i)} = -1$$

For all training data $x^{(i)}, y^{(i)}$, $i \in \{1, \ldots, n\}$

Perfect separation between the 2 classes

Separating hyperplane



$$y^{(i)}(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}) > 0$$

For all training data $x^{(i)}, y^{(i)}$, $i \in \{1, \ldots, n\}$

From separating hyperplane to classifier

- Training data $x^{(1)}, \dots, x^{(n)}$ with $x^{(i)} = \left(x_1^{(i)}, \dots, x_d^{(i)}\right)^{\mathrm{T}}$
- Labels are from 2 classes: $y^{(i)} \in \{-1,1\}$
- Let $\theta_0, \dots, \theta_d$ (will be learned) such that:

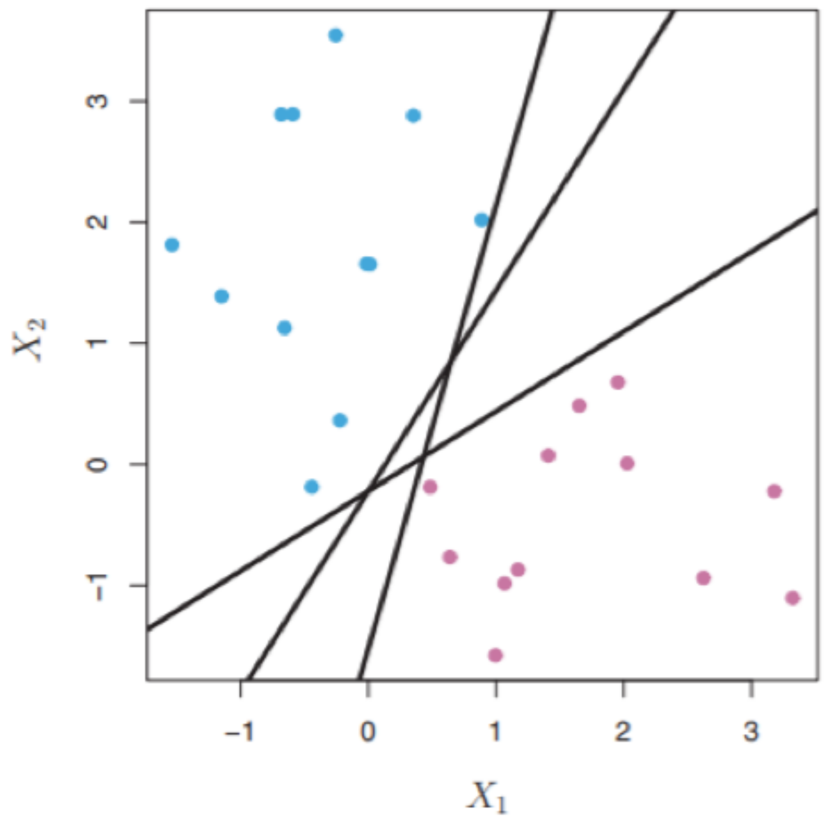
$$y^{(i)}(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}) > 0$$

- Classifier

$$f(z) = \mathrm{sign}(\theta_0 + \theta_1 z_1 + \cdots \theta_d z_d) = \mathrm{sign}(\theta^T \mathrm{z})$$

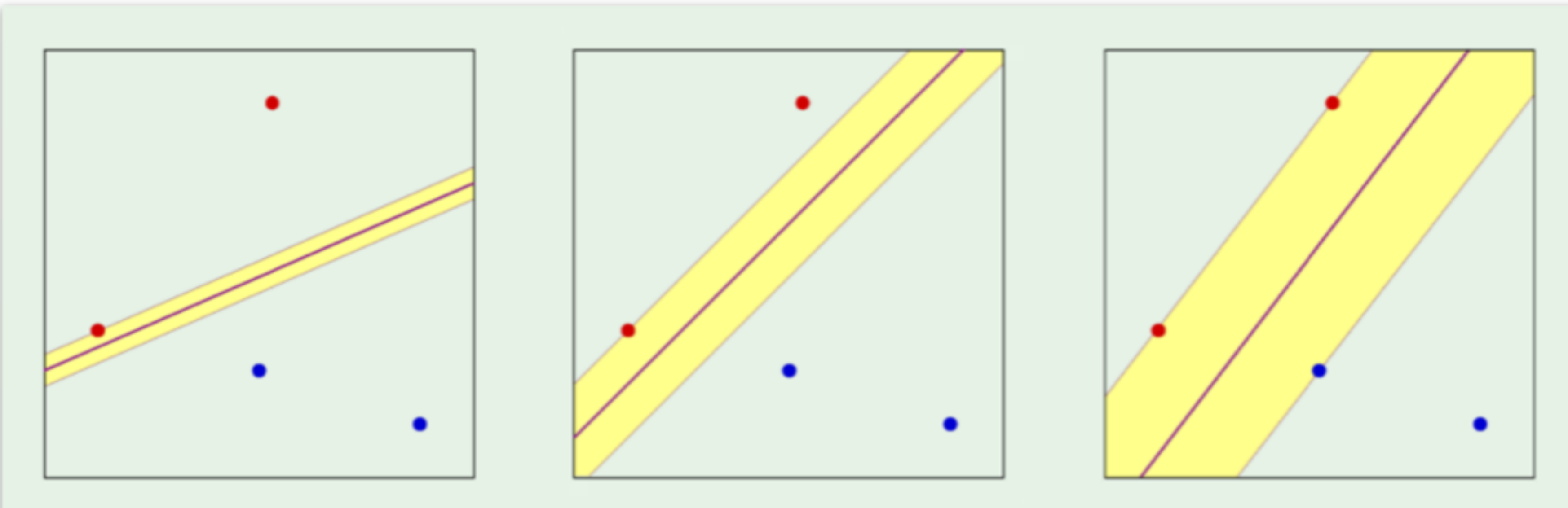
- Classify new test point x'
  - If $f(x') > 0$ predict y'$= 1$
  - Otherwise predict y'$= -1$

Separating hyperplane



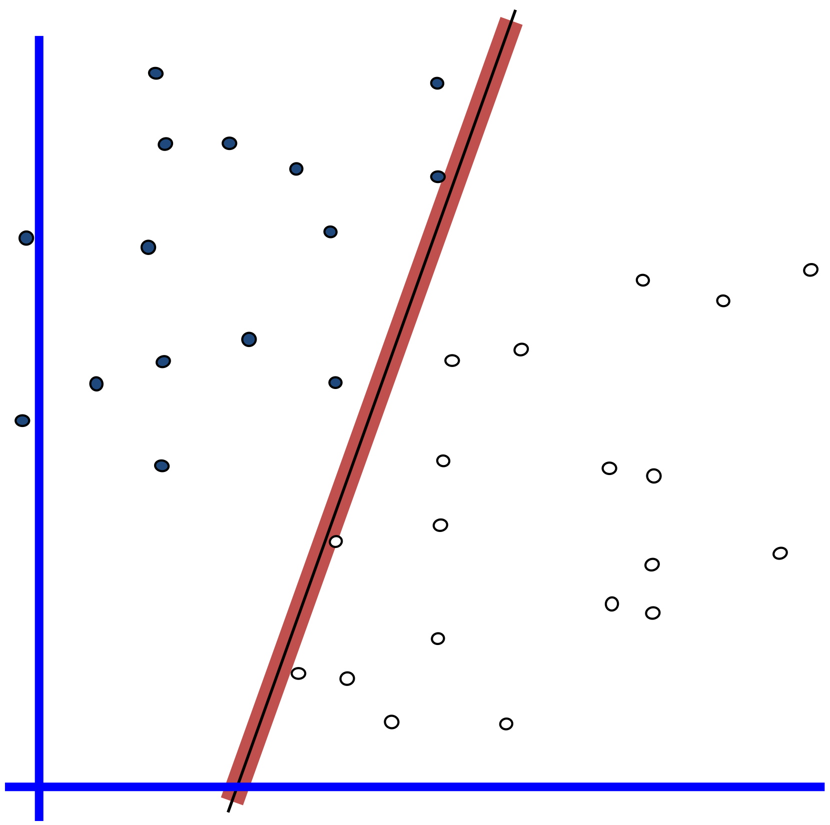
- If a separating hyperplane exists, there are infinitely many
- Which one should we choose?

Intuition



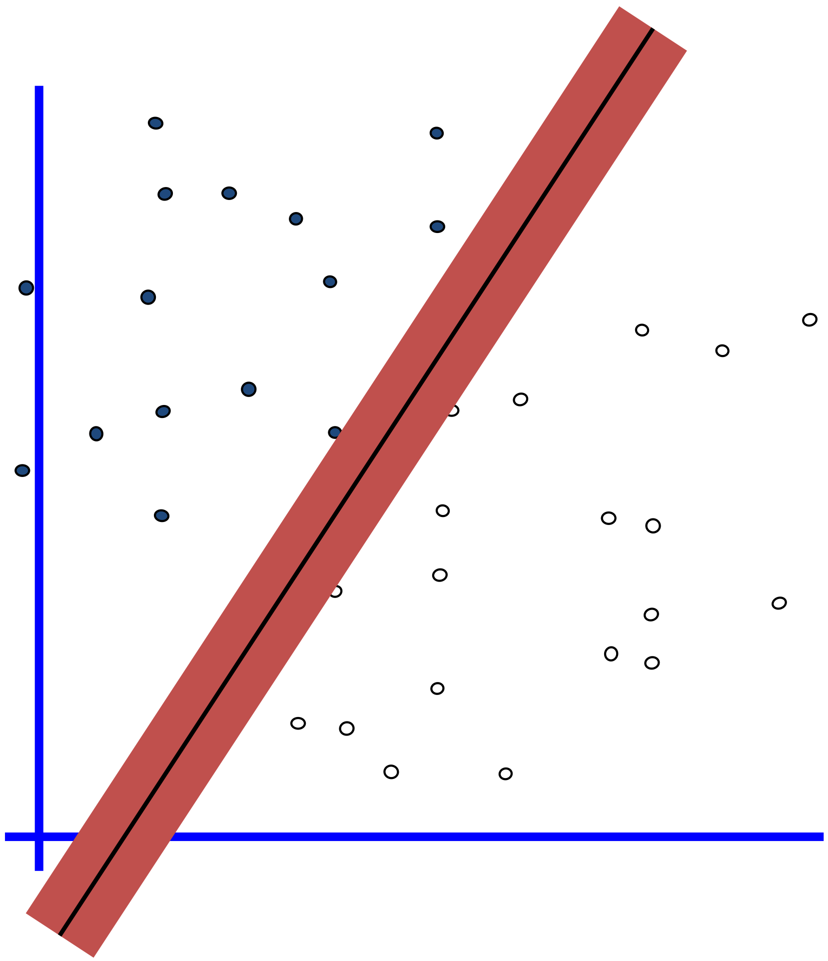
Which of these linear classifiers is the best?

Classifier Margin



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

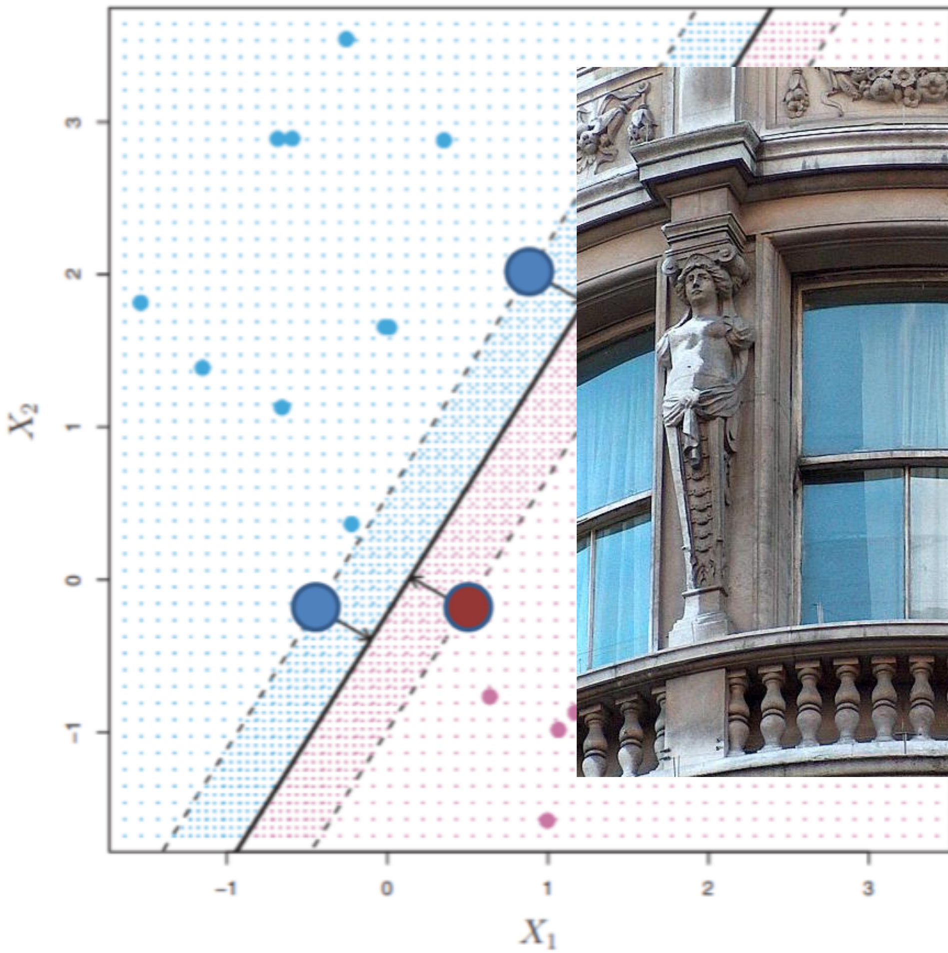
Maximum Margin

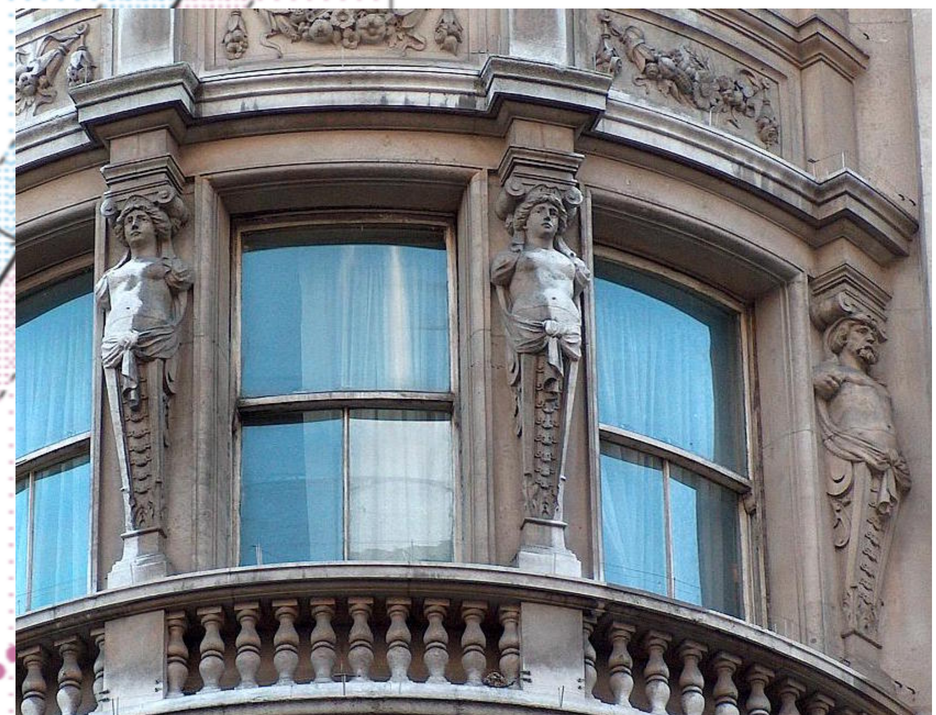


Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a data point.

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

Support Vectors (informally)





- Support vectors = points "closest" to hyperplane
- If support vectors change, classifier changes
- If other points change, no effect on classifier

Finding the maximum margin classifier

- Training data $x^{(1)}, \ldots, x^{(n)}$ with $x^{(i)} = \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)^{\mathrm{T}}$
- Labels are from 2 classes: $y_i \in \{-1,1\}$

maximize M

$$y^{(i)}\left(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}\right) \geq M \ \forall i$$

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

Normalization constraint (ok because if $\theta^T x = 0$, then also $k\theta^T x = 0$)

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

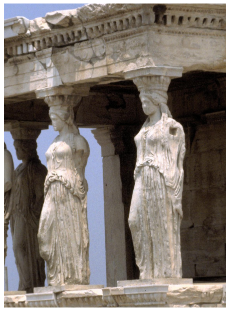
Equivalent formulation

- Min $||\theta||^2$
- $y^{(i)}\left(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}\right) \geq 1\ \forall i$

- Maximum margin classifier – given by solution θ to this optimization problem
- Can be solved with quadratic optimization techniques. Easier to solve via its dual problem.

Properties of solution

- The solution to the (dual) optimization happens to provide a convenient way to rewrite the decision function using new variables α_i
  - Originally: $f(z) = \text{sign}(\theta_0 + \theta_1 z_1 + \cdots \theta_d z_d) = \text{sign}(\theta^T \text{z})$
  - Equivalent to: $f(z) = \theta_0 + \sum_i \alpha_i < z, x^{(i)} >$
    - For test point *z*, the inner product $< z, x^{(i)} >$ = $z^T x^{(i)}$ with each training instance $x^{(i)}$ in turn.
    - And $\alpha_i \neq 0$ only for support vectors! For all other training points $\alpha_i = 0$.

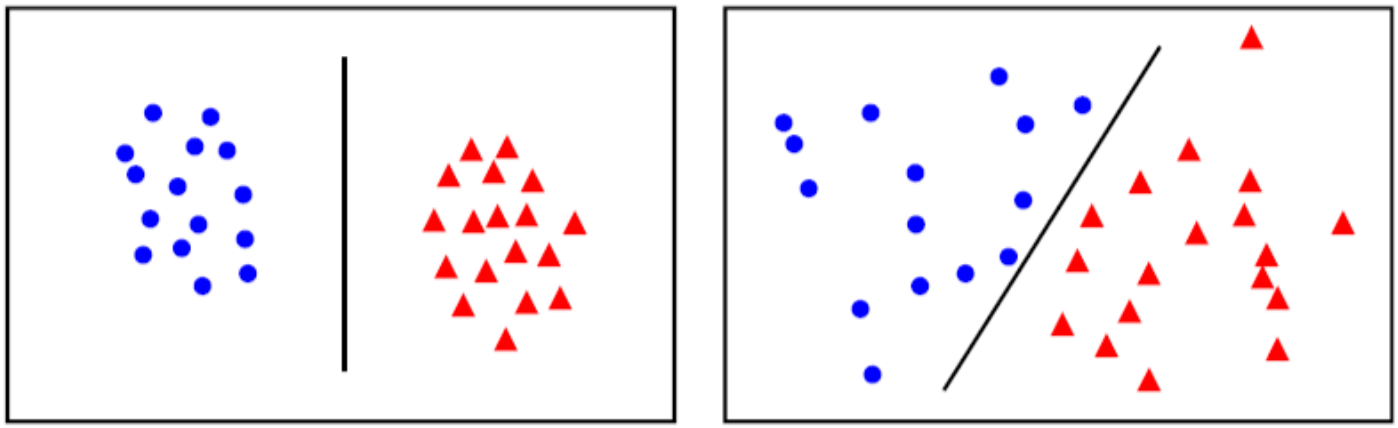


Outline

- Review of linear models
  - Separating hyperplanes
- Support Vector Machines
  - Linearly separable data
    - Maximum margin classifier
  - Non-separable data
    - Support vector classifier
  - Non-linear decision boundaries
    - Kernels and Radial SVM

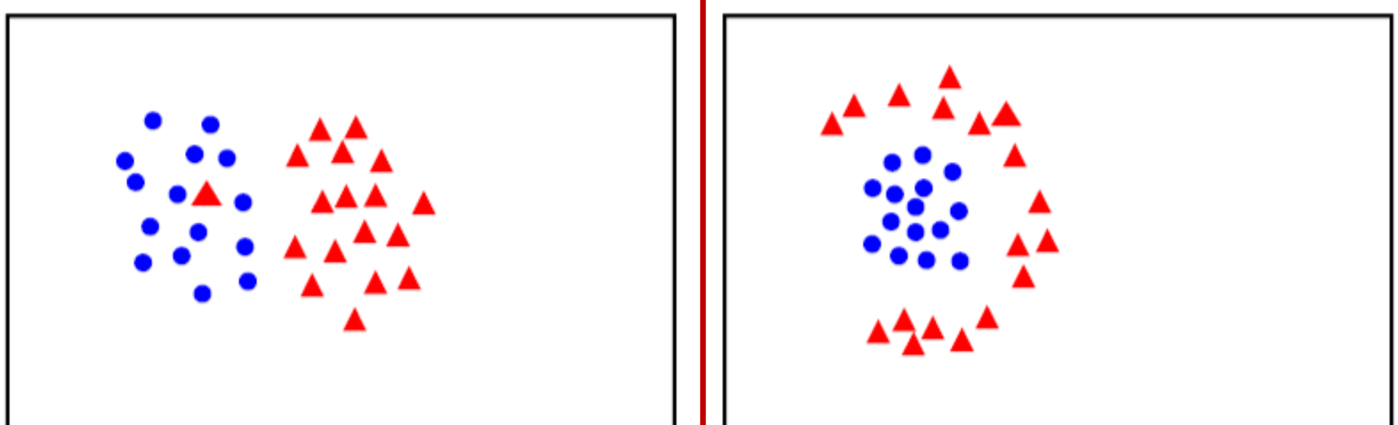
Linear separability

linearly
separable

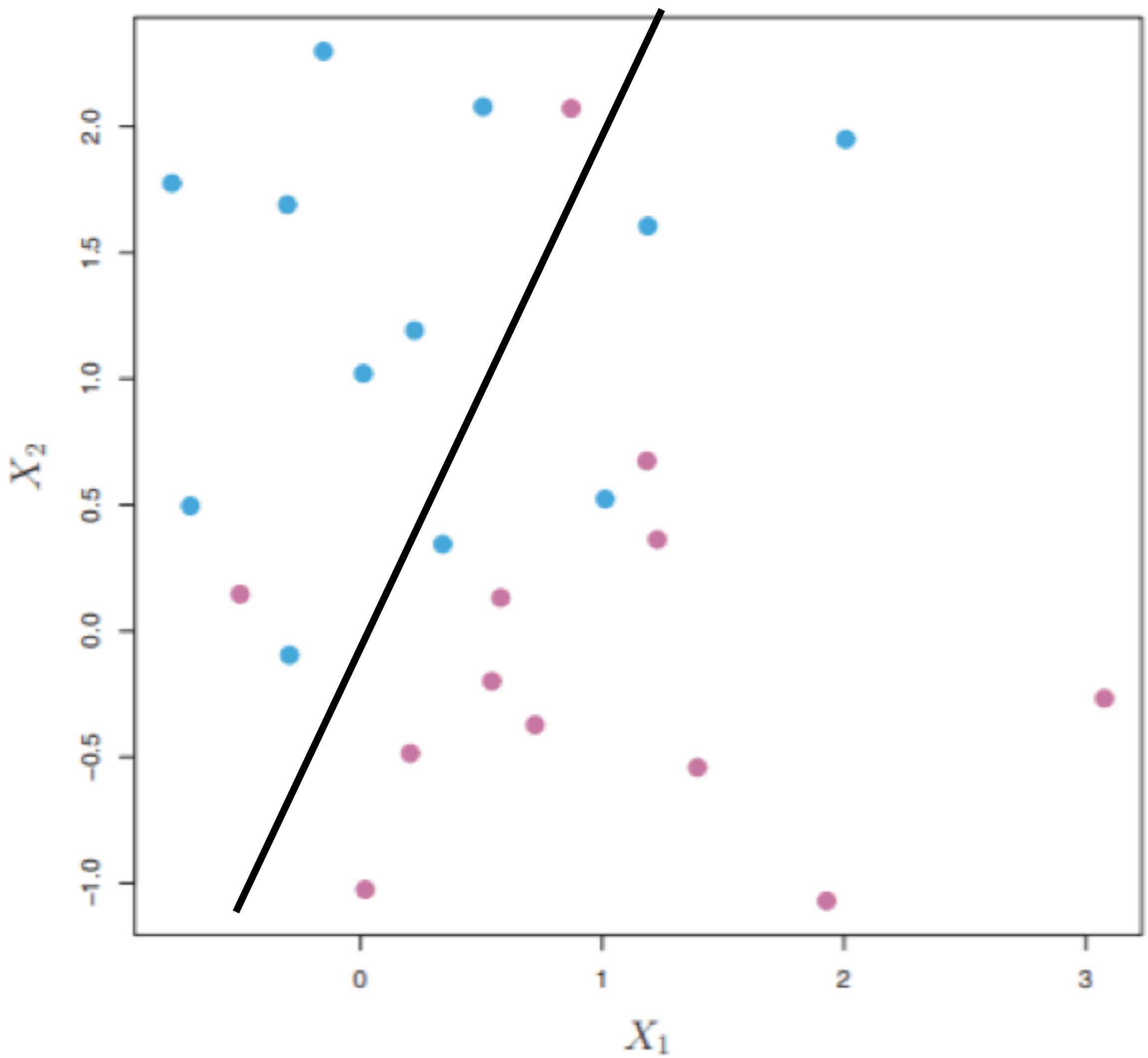


not
linearly
separable

(but almost)

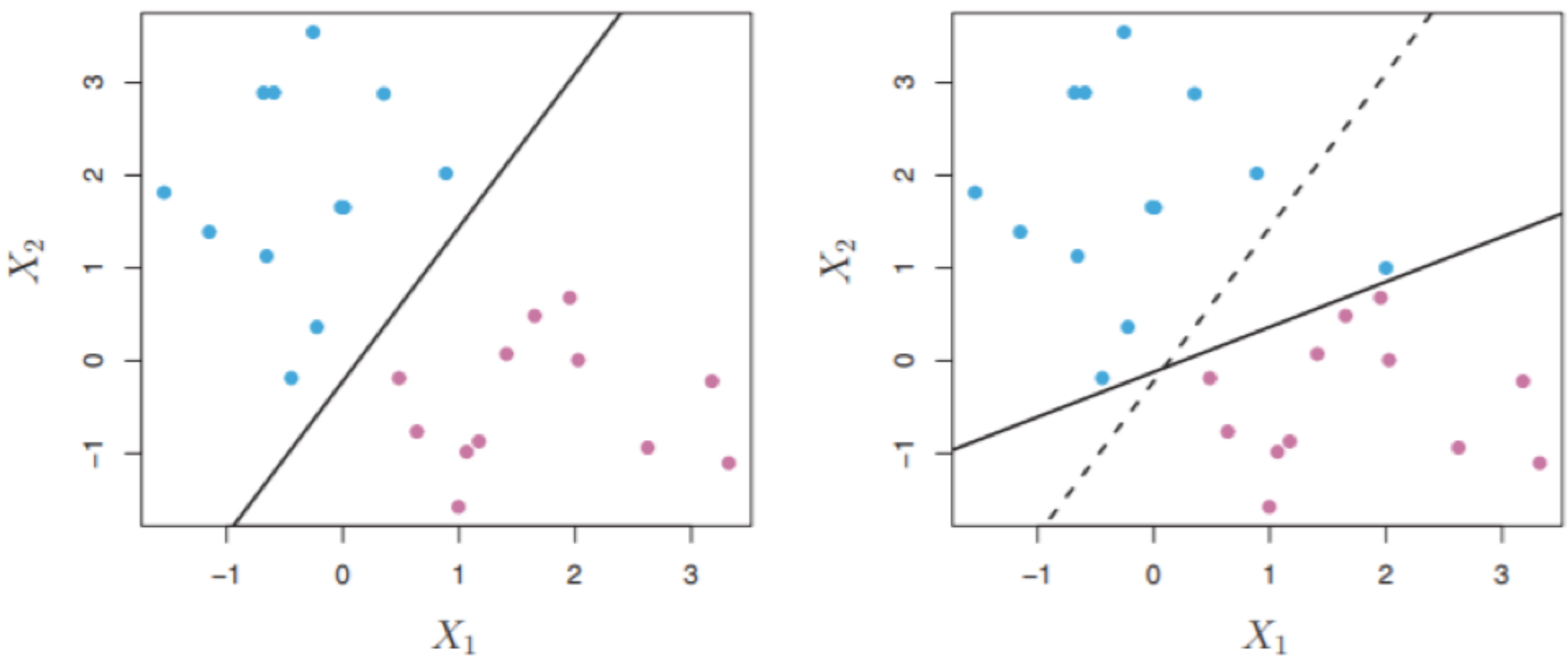


Non-separable case



Optimization problem has no solution!

Maximum margin is not always the best!



- Overfits to training data
- Sensitive to small modification (high variance)

Support vector classifier

- Allow for small number of mistakes on training data
- Obtain a more robust model

max M

$$y^{(i)}\left(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}\right) \geq M(1 - \epsilon_i\ )\forall i$$

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

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

Slack

Error Budget (Hyper-parameter)

max M

$$y^{(i)}\left(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}\right) \geq M(1 - \epsilon_i\ )\ \forall i$$

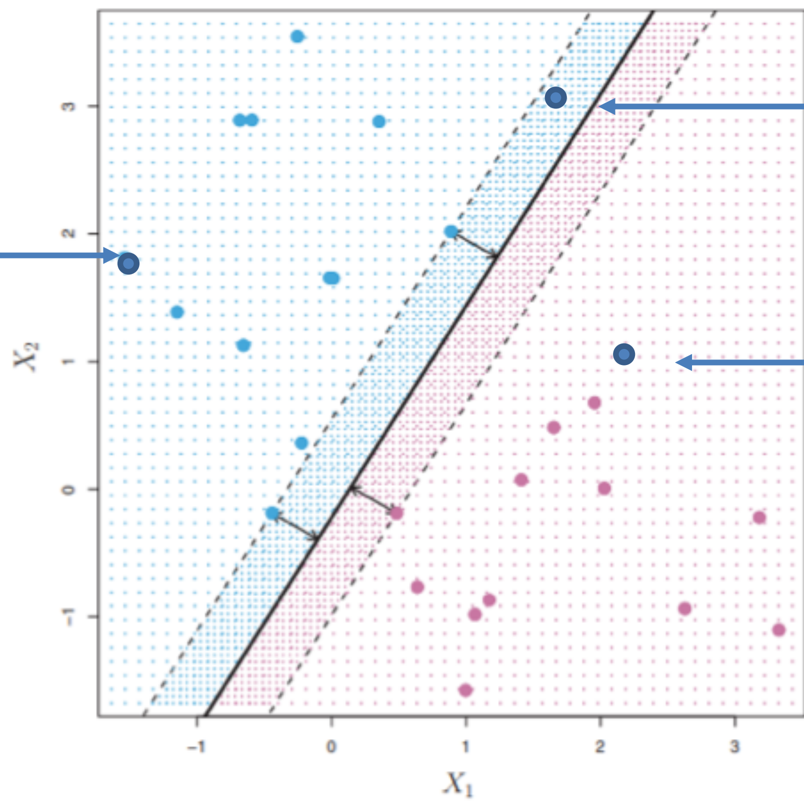
$$||\theta||_2 = 1$$

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

Error Budget

Slack

$\epsilon_i = 0$
Correct side of margin



$0 < \ \epsilon_i < 1$
Violates margin
Correct label

$\epsilon_i > 1$
Incorrect label
At most C data points

Equivalent formulation

- Min $||\theta||^2 + C\sum_i \epsilon_i$
- $y^{(i)}\left(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}\right) \geq 1 - \epsilon_i \ \forall i$
- $\epsilon_i \geq 0$

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

$$f(z) = \theta_0 + \sum_i \alpha_i < z, x^{(i)} >$$

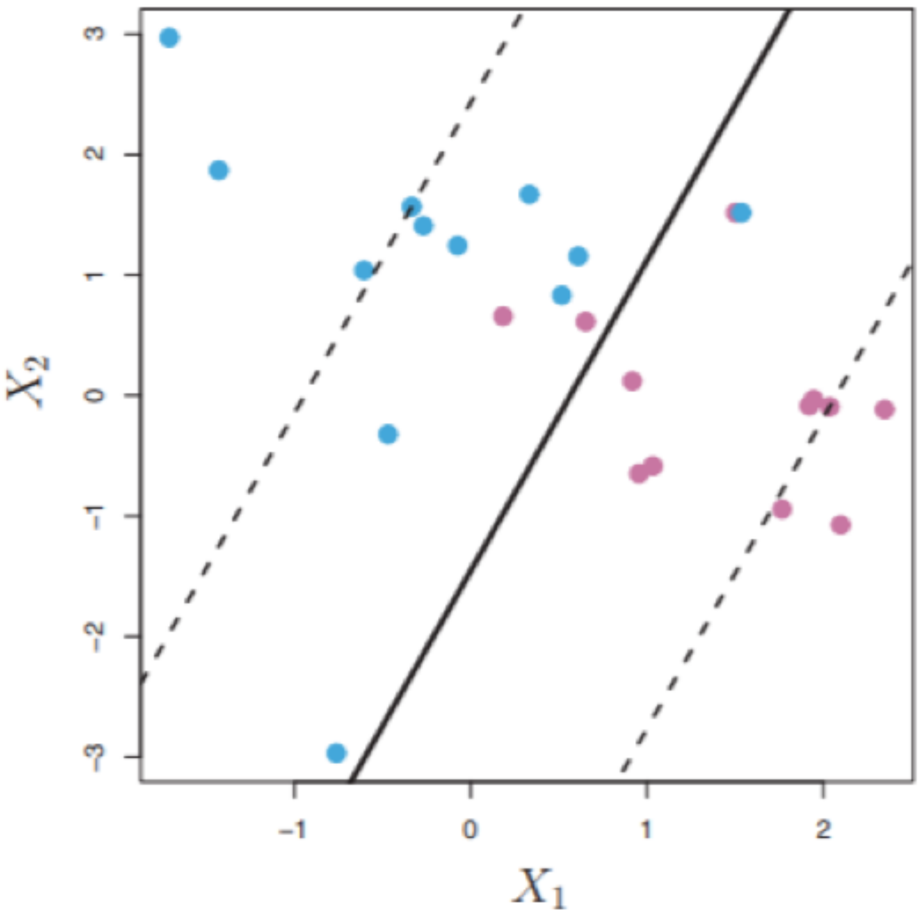
Where $\alpha_i \neq 0$ for support vectors (and $\alpha_i = 0$ for all other training points)

- This model is called Support Vector Classifier, also Linear SVM, also soft-margin classifier

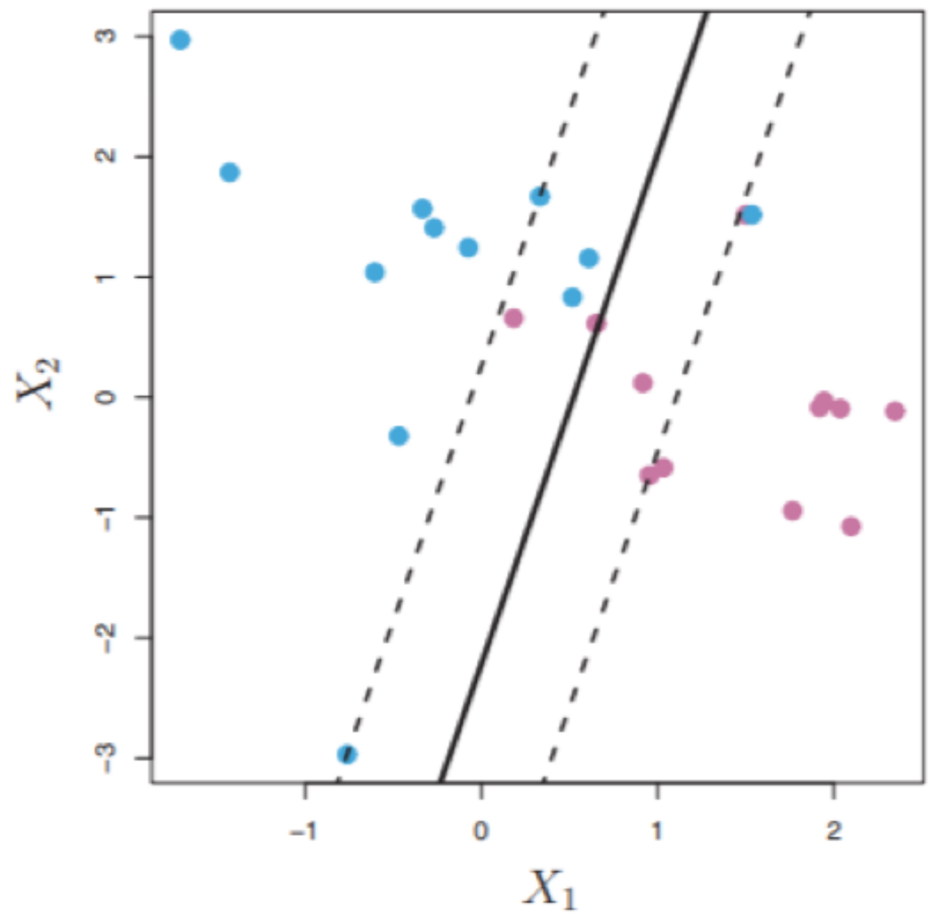
Properties

- Maximum margin classifier
  - Classifier of maximum margin
  - For linearly separable data
- Support vector classifier
  - Allows some slack and sets a total error budget (hyper-parameter)
- For both, final classifier on a point is a linear combination of inner product of point with support vectors
  - Efficient to evaluate

Error Budget and Margin



Larger C
Low variance



Smaller C
Over-fitting

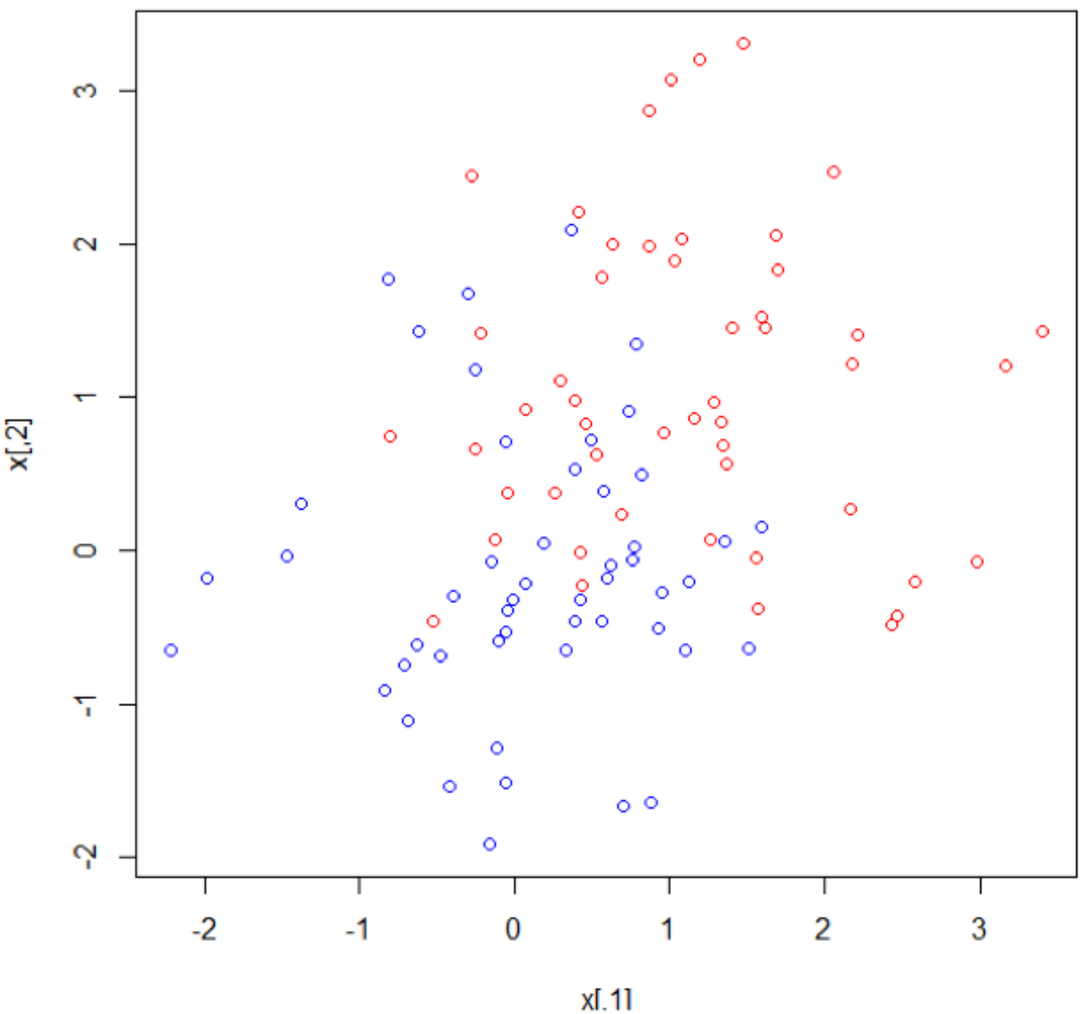
Find best hyper-parameter C by cross-validation

Resilience to outliers

- LDA is very sensitive to outliers
  - Estimates mean and co-variance using all training data
- SVM is resilient to outliers
  - Decision hyper-plane mainly depends on support vectors
- Logistic regression is also resilient to points far from decision boundary
  - Cross-entropy uses logs in the loss function

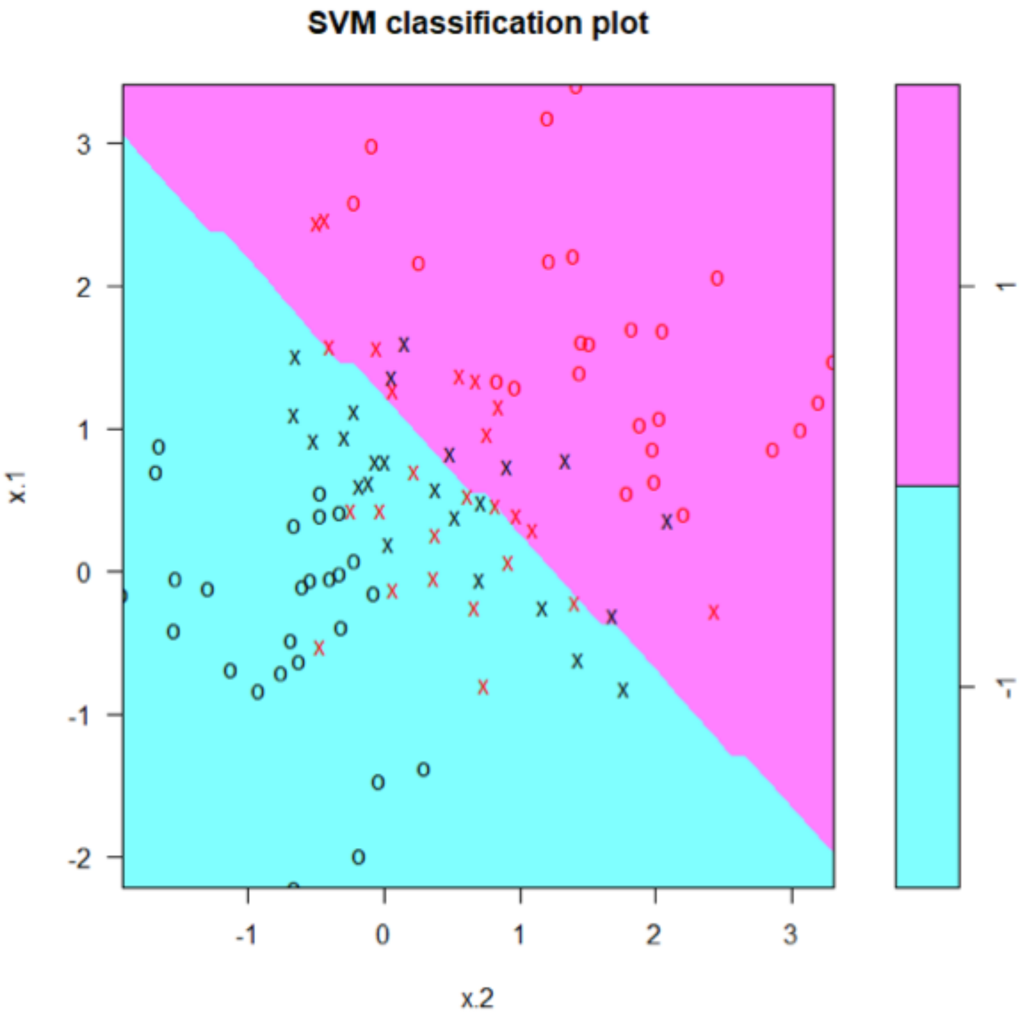
Lab – Linear SVM

```
> set.seed(1)
> x=matrix(rnorm(100*2), ncol=2)
> y=c(rep(-1,50), rep(1,50))
> x[y==1,]=x[y==1,] + 1
> plot(x, col=(3-y))
> dat=data.frame(x=x, y=as.factor(y))
>
```



Lab – Linear SVM

```
> library(e1071)
> svmfit=svm(y~., data=dat, kernel="linear", cost=10,scale=FALSE)
> plot(svmfit, dat)
```



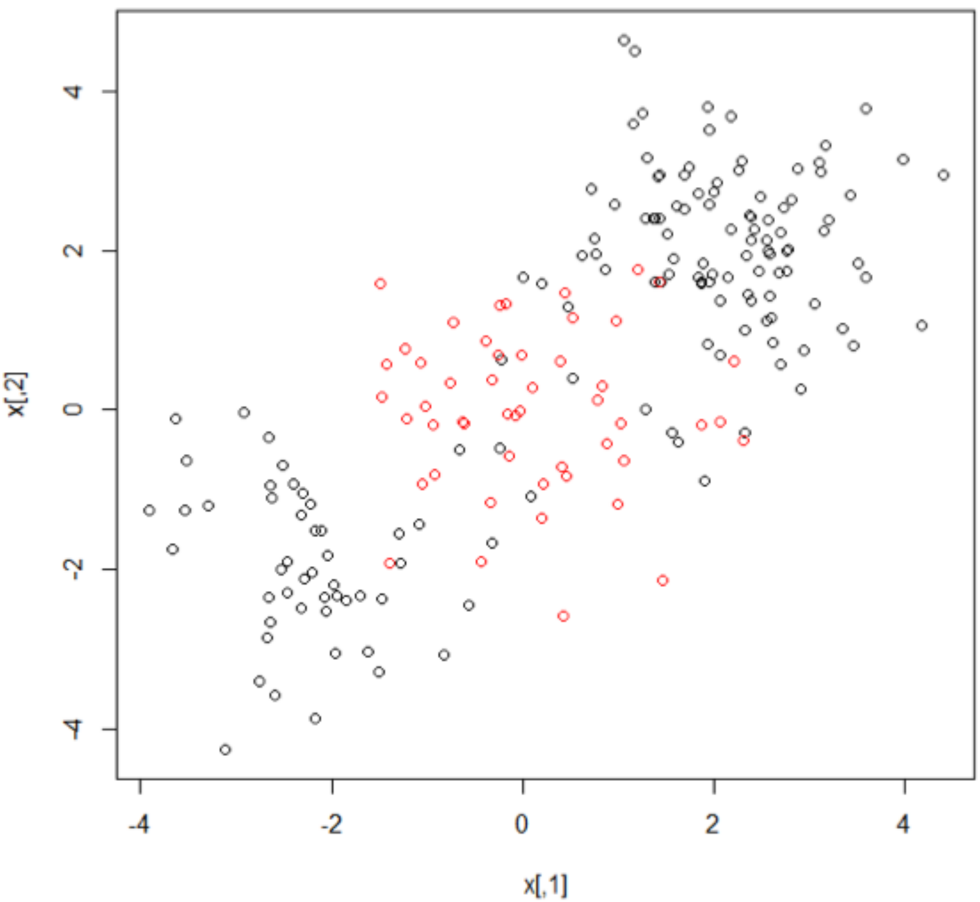
Lab – Linear SVM

```
> summary(svmfit)

Call:
svm(formula = y ~ ., data = dat, kernel = "linear", cost = 10, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10
      gamma:  0.5

Number of Support Vectors:  49

 ( 24 25 )


Number of Classes:  2

Levels:
 -1 1
```

```
> svmfit=svm(y~., data=dat, kernel="linear", cost=0.01,scale=FALSE)
>
>
>
> summary(svmfit)

Call:
svm(formula = y ~ ., data = dat, kernel = "linear", cost = 0.01, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.01
      gamma:  0.5

Number of Support Vectors:  88

 ( 44 44 )


Number of Classes:  2

Levels:
 -1 1
```

Lab – Radial SVM

```
>
> set.seed(1)
> x=matrix(rnorm(200*2), ncol=2)
> x[1:100,]=x[1:100,]+2
> x[101:150,]=x[101:150,]-2
> y=c(rep(1,150),rep(2,50))
> dat=data.frame(x=x,y=as.factor(y))
> plot(x, col=y)
```



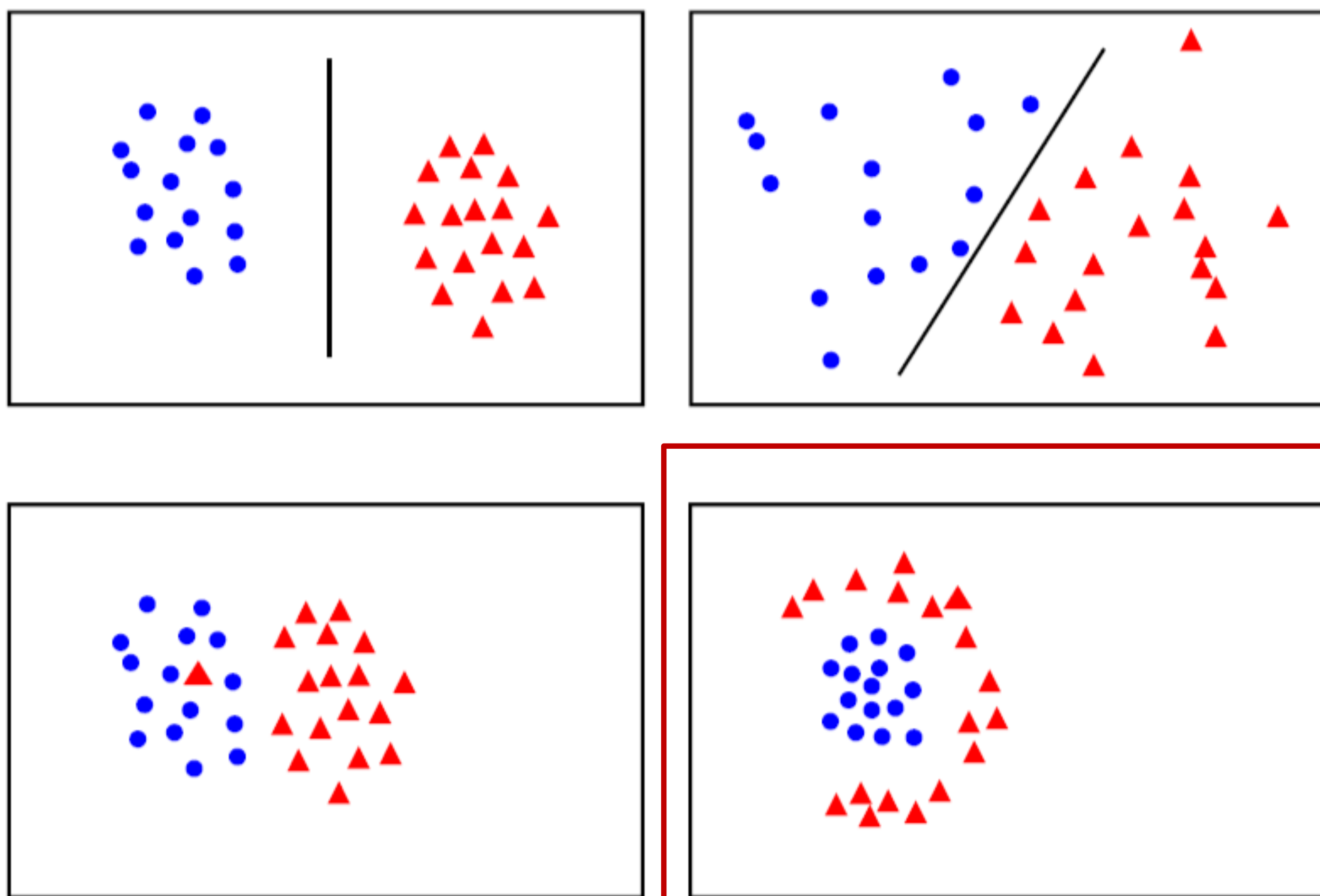
Outline

- Review of linear models
  - Separating hyperplanes
- Support Vector Machines
  - Linearly separable data
    - Maximum margin classifier
  - Non-separable data
    - Support vector classifier
  - Non-linear decision boundaries
    - Kernels and Radial SVM

Linear separability

linearly separable

not linearly separable (but almost)



(not even close!)

Non-linear decision

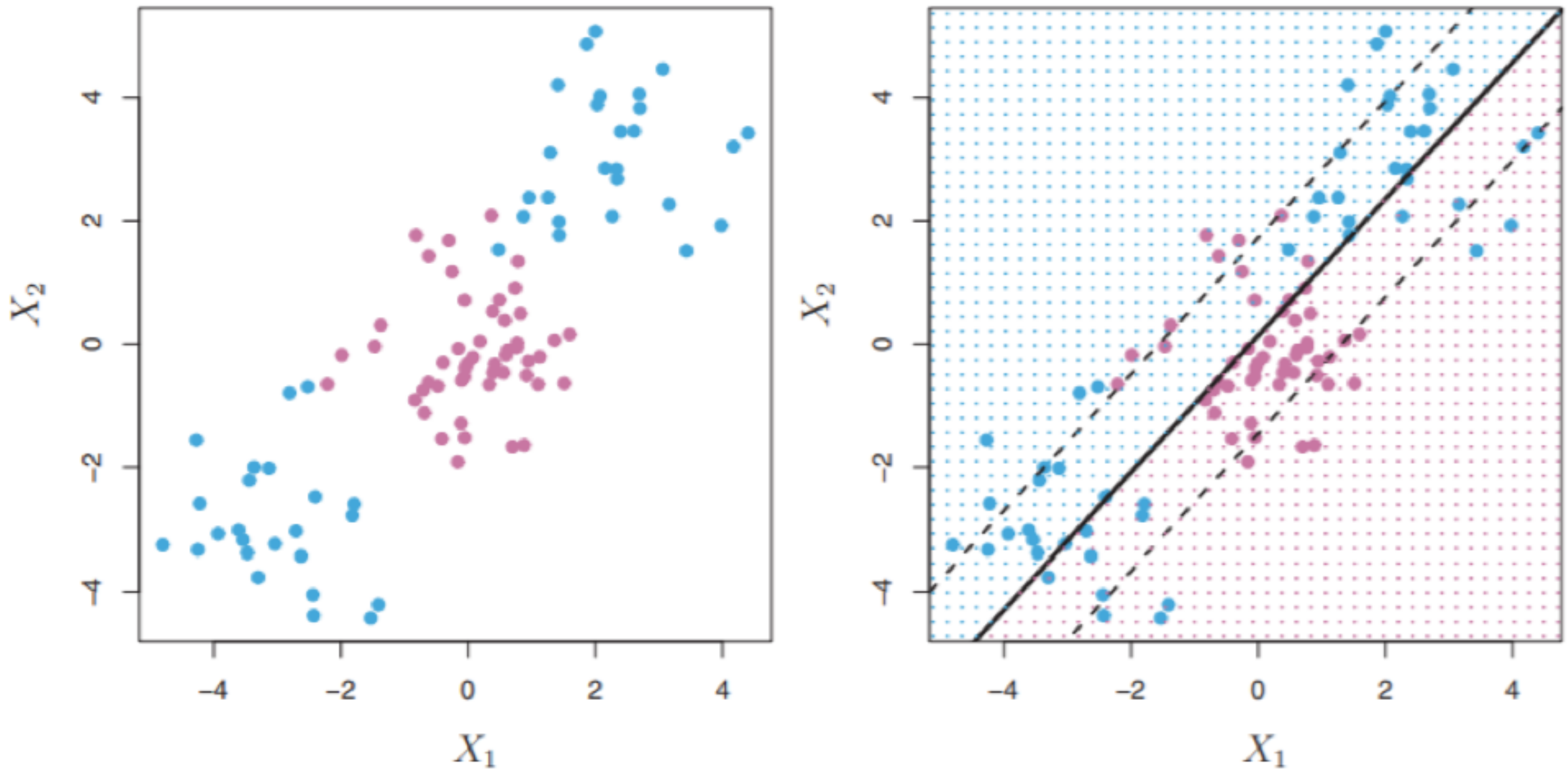
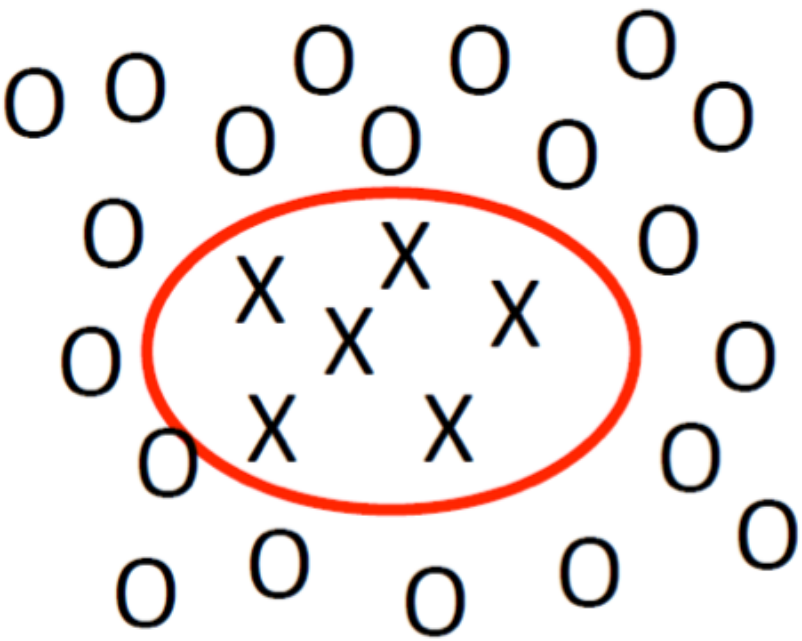


FIGURE 9.8. Left: *The observations fall into two classes, with a non-linear boundary between them.* Right: *The support vector classifier seeks a linear boundary, and consequently performs very poorly.*

More examples



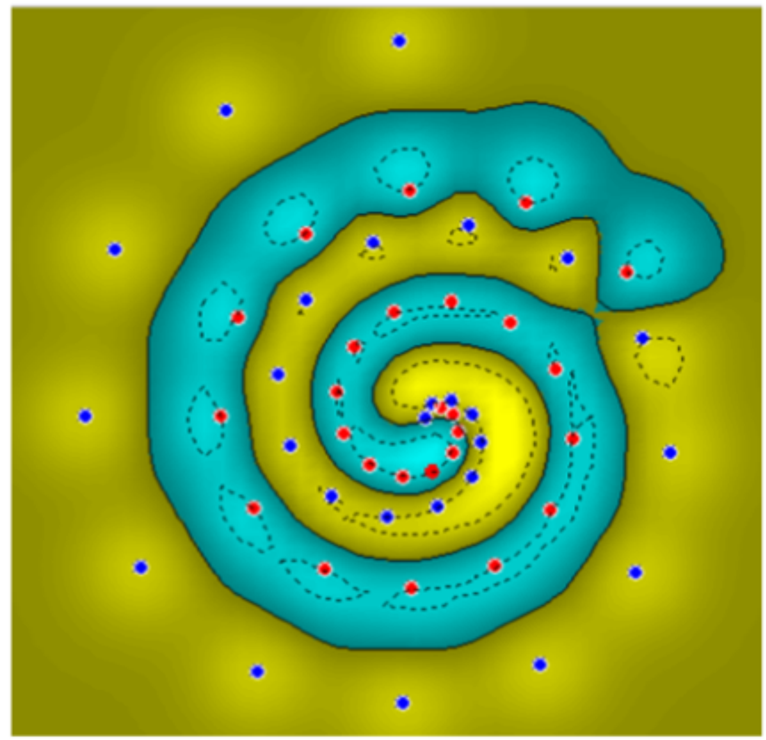


Image from http://www.atrandomresearch.com/iclass/

Kernels

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

Any kernel function!

- What is a kernel?
  - Function that characterizes similarity between 2 observations
  - $K(a,b) =< a, b > = \sum_j a_j b_j$ 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

Kernels

- Linear kernels
  - $K(a,b) = < a,b > = \sum_i a_i b_i$
- Polynomial kernel of degree *m*
  - $K(a,b) = \left(1 + \sum_{i=0}^{d} a_i b_i \right)^m$
- Radial Basis Function (RBF) kernel (or Gaussian)
  - $K(a,b) = \exp\left(-\gamma \sum_{i=0}^{d} (a_i - b_i)^2\right)$
- Support vector machine classifier
  - $\mathrm{h}(z) = \theta_0 + \sum_{i \in S} \alpha_i K(z, x^{(i)})$

General SVM classifier

- S = set of support vectors
- SVM with polynomial kernel
  - $h(z) = \theta_0 + \sum_{i \in S} \alpha_i \left(1 + \sum_{j=0}^{d} z_j x_j^{(i)}\right)^m$
  - Hyper-parameter m (degree of polynomial)
- SVM with radial kernel
  - $h(z) = \theta_0 + \sum_{i \in S} \alpha_i \exp\left(-\gamma \sum_{j=0}^{d} (z_j - x_j^{(i)})^2\right)$
  - Hyper-parameter γ (increase for non-linear data)
  - As testing point z is closer to support vector, kernel is close to 1
  - Local behavior: points far away have negligible impact on prediction

Kernel Example

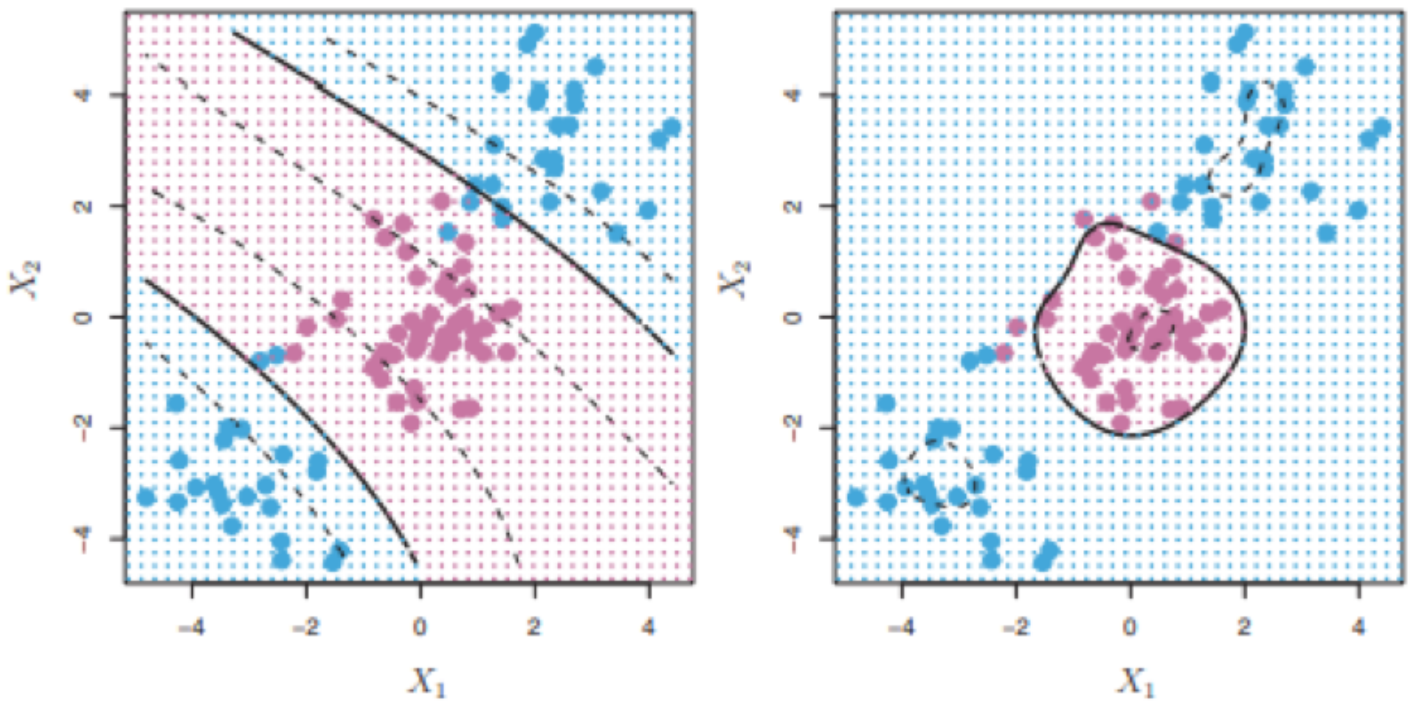


FIGURE 9.9. Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

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

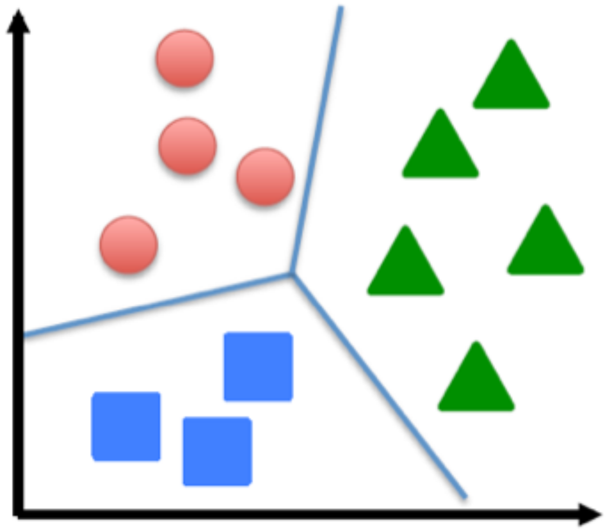
Review SVM

- 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)

SVM for Multiple Classes



$y \in \{1, \ldots, K\}$

- Many SVM packages already have multi-class classification built in
- Otherwise, use one-vs-rest
  - Train K SVMs, each picks out one class from rest, yielding $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(K)}$
  - Predict class i with largest $(\boldsymbol{\theta}^{(i)})^\intercal \mathbf{x}$

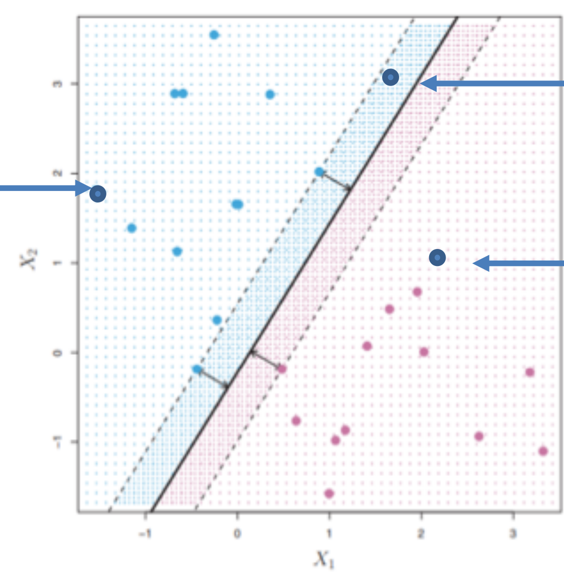
Comparing SVM with other classifiers

- 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

Support vector classifier

$$h(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}$$

- Min $||\theta||^2 + C \sum_i \epsilon_i$
- $y^{(i)} \left(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}\right) \geq 1 - \epsilon_i \; \forall i$
- $\epsilon_i \geq 0$
- Rearranging: $1 - y^{(i)} h(x^{(i)}) \leq \epsilon_i$



$\epsilon_i = 0$
Correct side of margin

$0 < \epsilon_i < 1$
Violates margin
Correct label

$\epsilon_i > 1$
Incorrect label

Support vector classifier

$$\mathrm{h}\left(x^{(i)}\right) = \theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}$$

- Min $\left|\left|\theta\right|\right|^2 + C \sum_i \epsilon_i$
- $y^{(i)}\left(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}\right) \geq 1 - \epsilon_i \; \forall i$
- $\epsilon_i \geq 0$
- Rearranging: $1 \; - y^{(i)} h\left(x^{(i)}\right) \leq \epsilon_i$
- Define cost($h\left(x^{(i)}\right), y^{(i)}$) $= \max\left(0{,}1 - y^{(i)} h\left(x^{(i)}\right)\right)$
  – When $\epsilon_i > 0$, this is just ϵ_i
  – When *i* is correctly classified (and outside the margin), $1 \; - y^{(i)} h\left(x^{(i)}\right) \leq 0$, so cost = 0

Hinge Loss

$$\text{h}\left(x^{(i)}\right) = \theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}$$

- $J(\theta) = \sum_{i=1}^{n} \max\left(0, 1 - y^{(i)} h\left(x^{(i)}\right)\right) + \lambda \sum_{j=1}^{d} \theta_j^2$

Hinge loss

Total Error Budget

Regularization Term

$$J(\theta) = C \sum_{i=0}^{n} \max\left(0, 1 - y^{(i)} h\left(x^{(i)}\right)\right) + \sum_{j=1}^{d} \theta_j^2$$

$C = \frac{1}{\lambda}$

Objective for Logistic Regression

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n} \left[y^{(i)} \log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \left(1 - y^{(i)}\right) \log \left(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right) \right]$$

- Cost of a single instance:

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

- Can re-write objective function as

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

Regularized Logistic Regression

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y^{(i)}\log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \left(1 - y^{(i)}\right)\log\left(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right)\right]$$

- 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

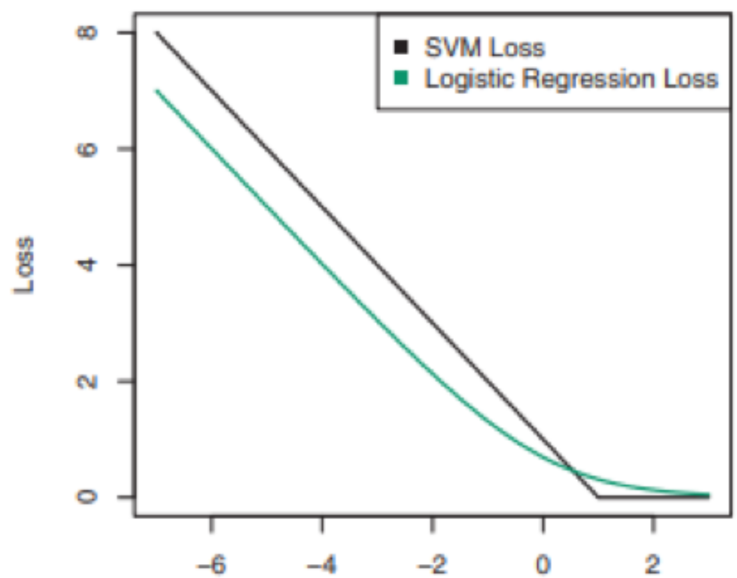
Connection to Logistic Regression

- $J(\theta) = \sum_{i=0}^{n} \underbrace{\max\left(0, 1 - y^{(i)} f\left(x^{(i)}\right)\right)}_{\text{Hinge loss}} + \lambda \sum_{j=1}^{d} \theta_j^2$

$f\left(x^{(i)}\right) = \theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}$

- $J(\theta) = C \sum_{i=0}^{n} \max\left(0, 1 - y^{(i)} f\left(x^{(i)}\right)\right) + \sum_{j=1}^{d} \theta_j^2$

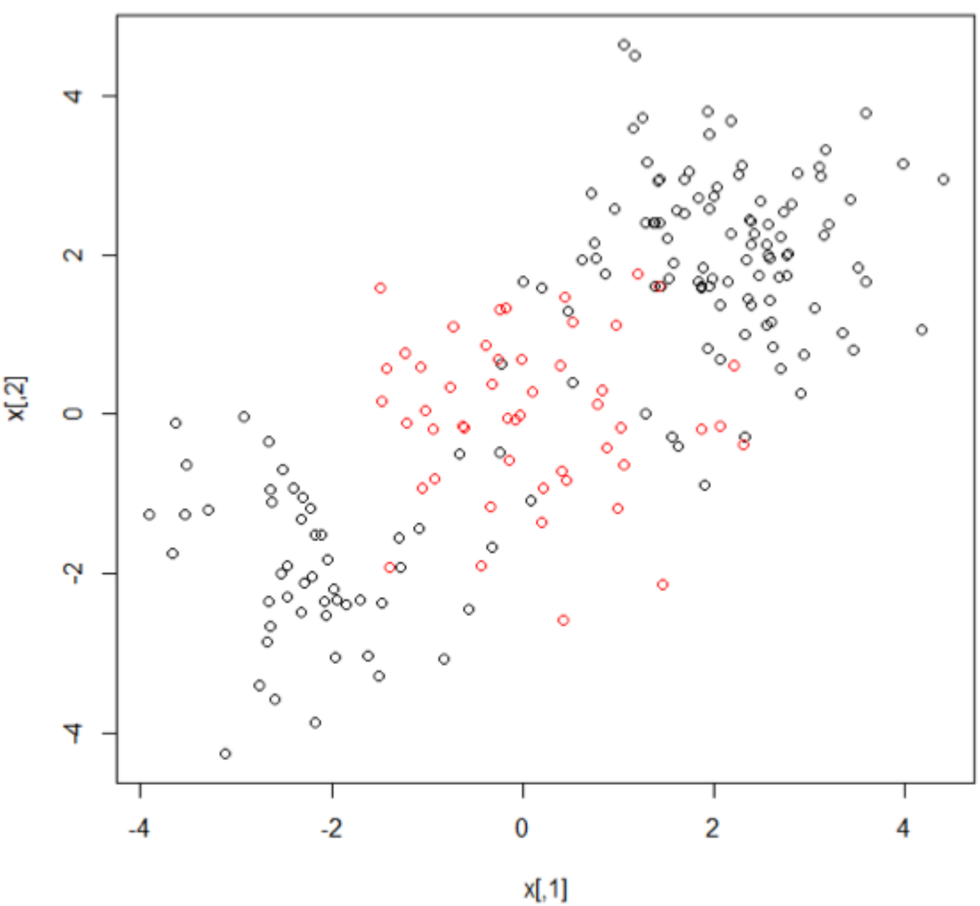
$C = \text{regularization cost}$



$y^{(i)}\left(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}\right)$

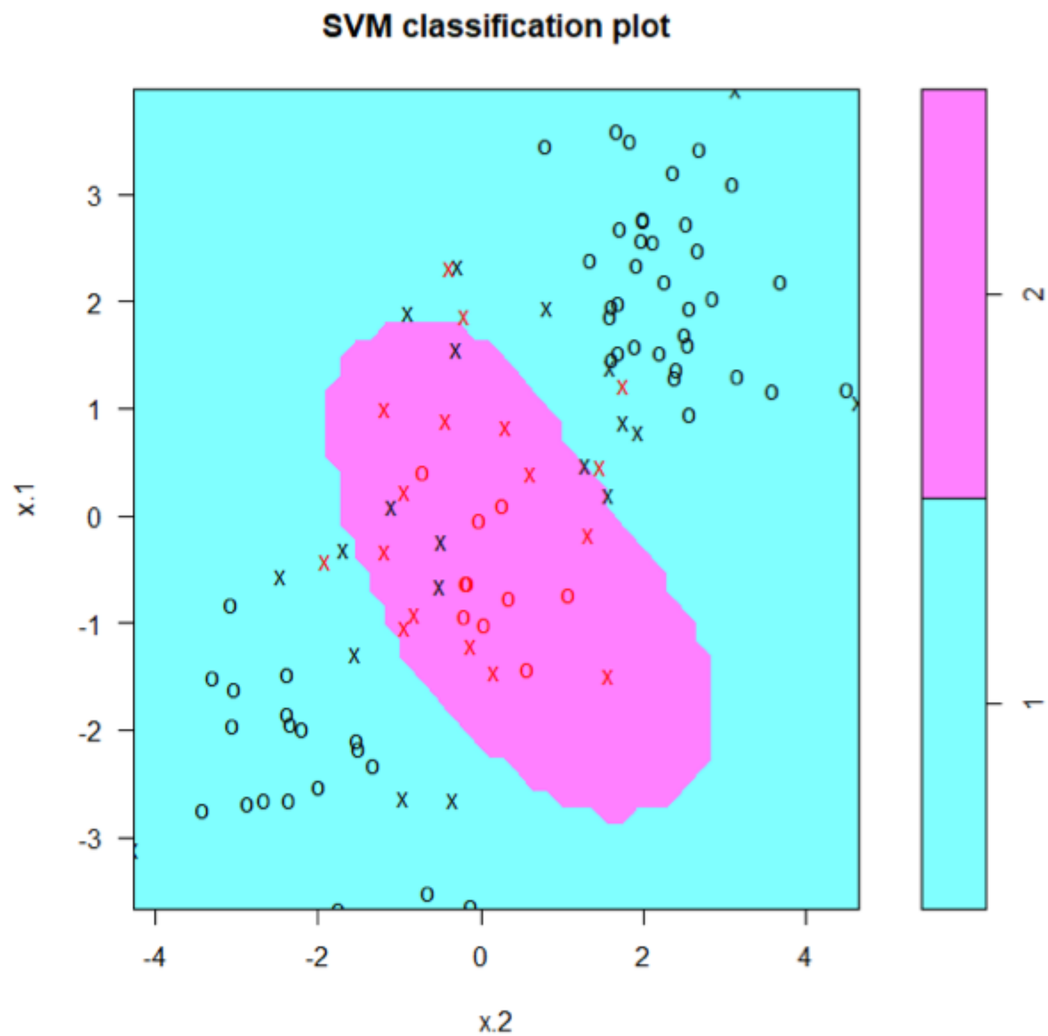
Lab – Radial SVM

```
>
> set.seed(1)
> x=matrix(rnorm(200*2), ncol=2)
> x[1:100,]=x[1:100,]+2
> x[101:150,]=x[101:150,]-2
> y=c(rep(1,150),rep(2,50))
> dat=data.frame(x=x,y=as.factor(y))
> plot(x, col=y)
```



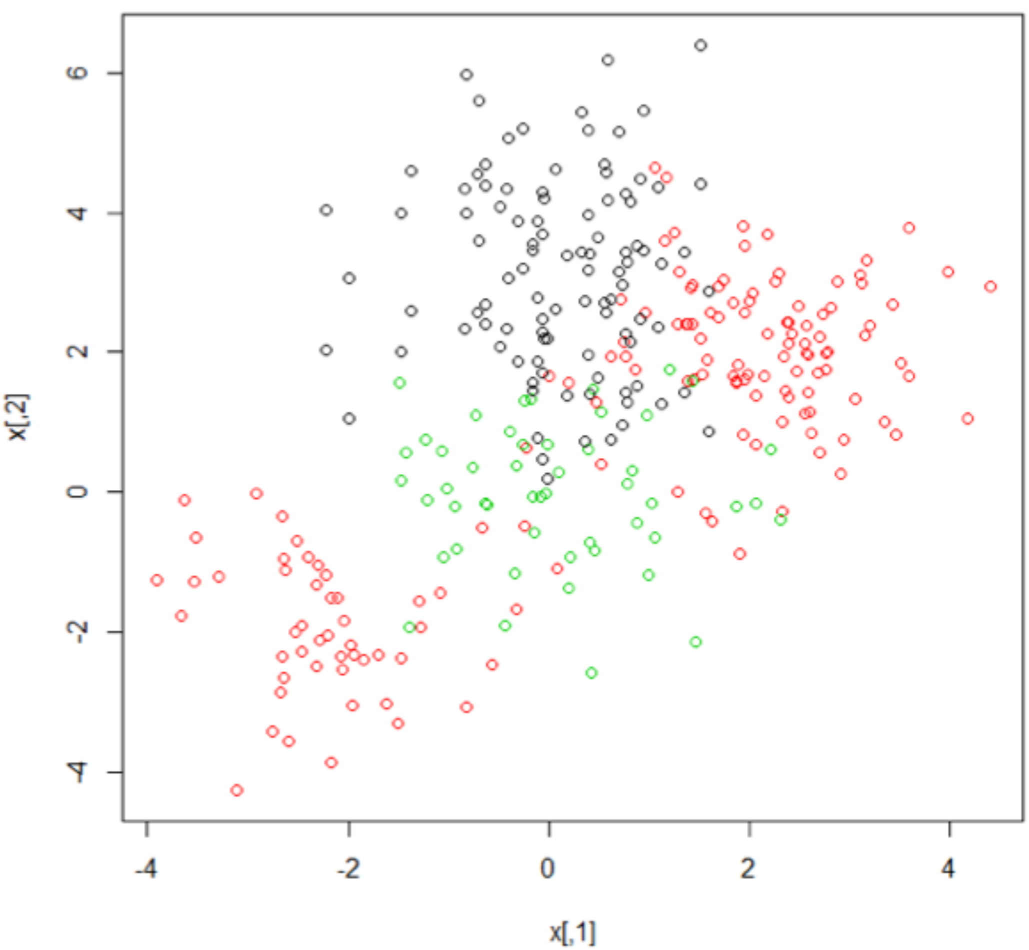
Lab – Radial SVM

```
> train=sample(200,100)
> svmfit=svm(y~., data=dat[train,], kernel="radial",  gamma=1, cost=1)
> plot(svmfit, dat[train,])
> 
```



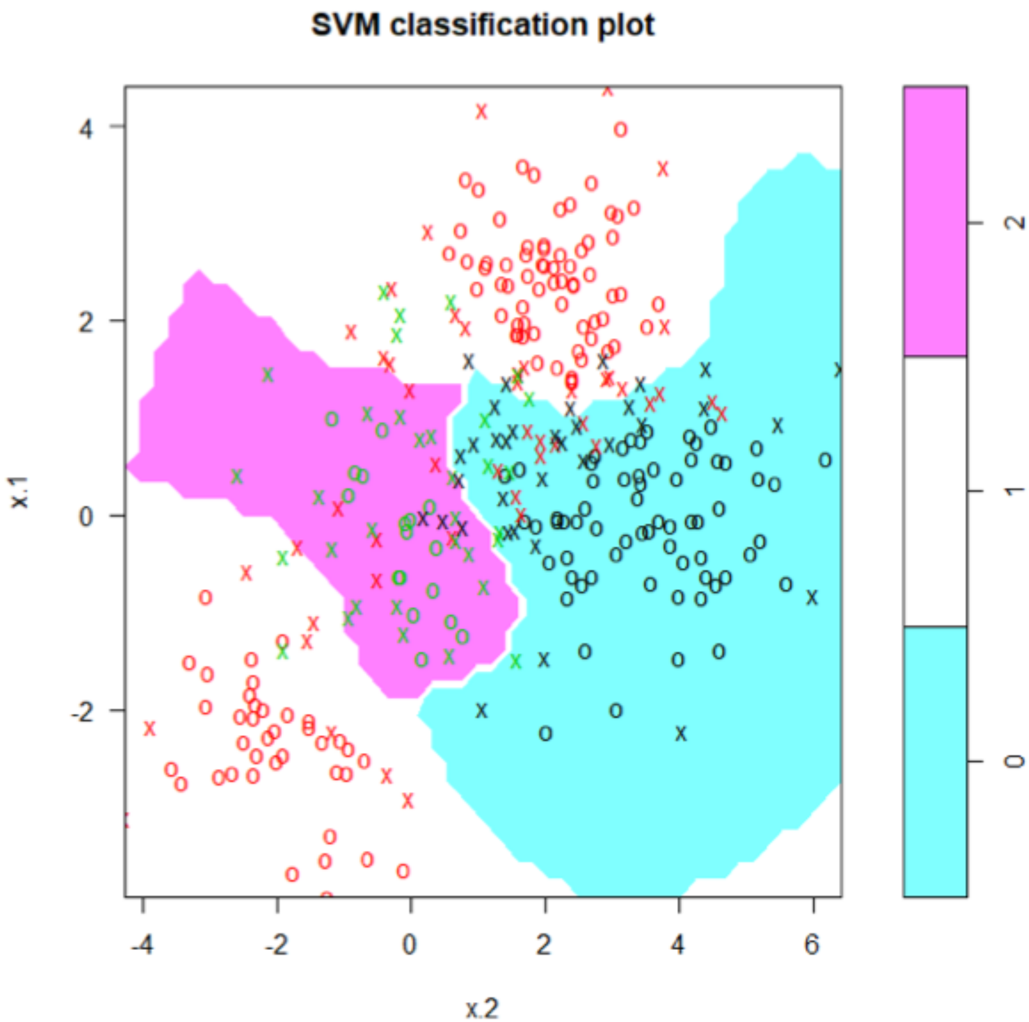
Lab – Multiple Classes

```
>
> set.seed(1)
> x=rbind(x, matrix(rnorm(50*2), ncol=2))
> y=c(y, rep(0,50))
> x[y==0,2]=x[y==0,2]+2
> dat=data.frame(x=x, y=as.factor(y))
> par(mfrow=c(1,1))
> plot(x,col=(y+1))
>
```



Lab – Multiple Classes

```
>
> svmfit=svm(y~., data=dat, kernel="radial", cost=10, gamma=1)
> plot(svmfit, dat)
>
```



Acknowledgements

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