DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

October 23 2018

Logistics

- Project proposal is due on Oct 24 (1 page on Gradescope)
  - Project Title
  - Problem Description
  - Dataset
  - Approach
- Final project
  - Presentation: Monday, Dec 3
  - Report: Friday, Dec 7
- Final exam
  - Tuesday, Dec 11

Review

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

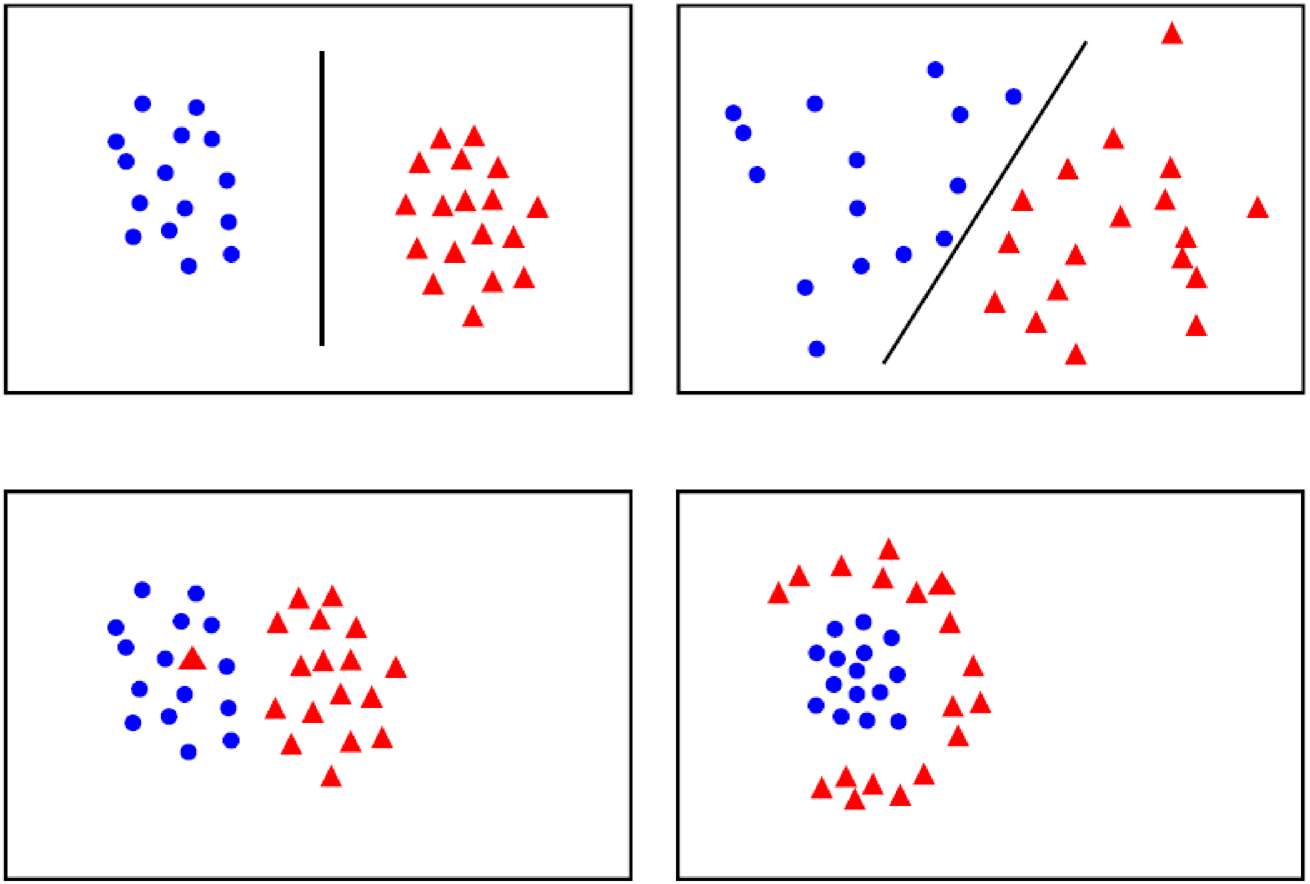
Outline

- Support vector classifier
  - Review
  - Hinge loss
- SVM
  - Non-linear decision boundaries
  - Kernels
  - Polynomial and Radial SVM
- Density estimators

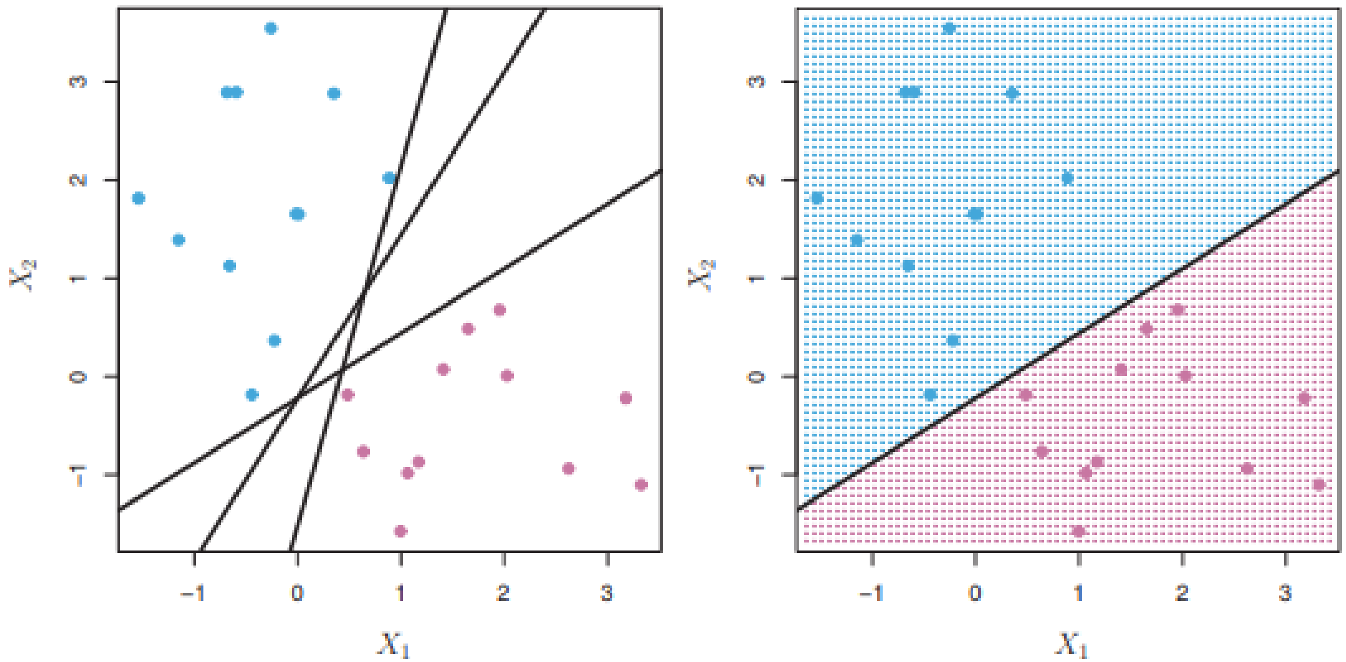
Linear separability

linearly
separable

not
linearly
separable



Separating hyperplane



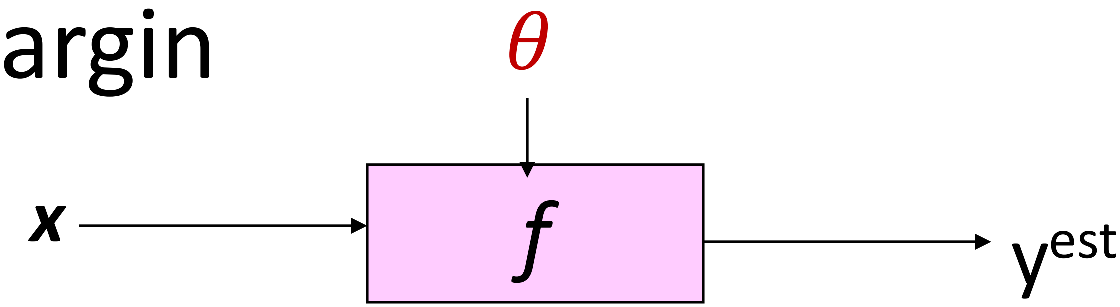
$$y^{(i)}(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}) > 0$$

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

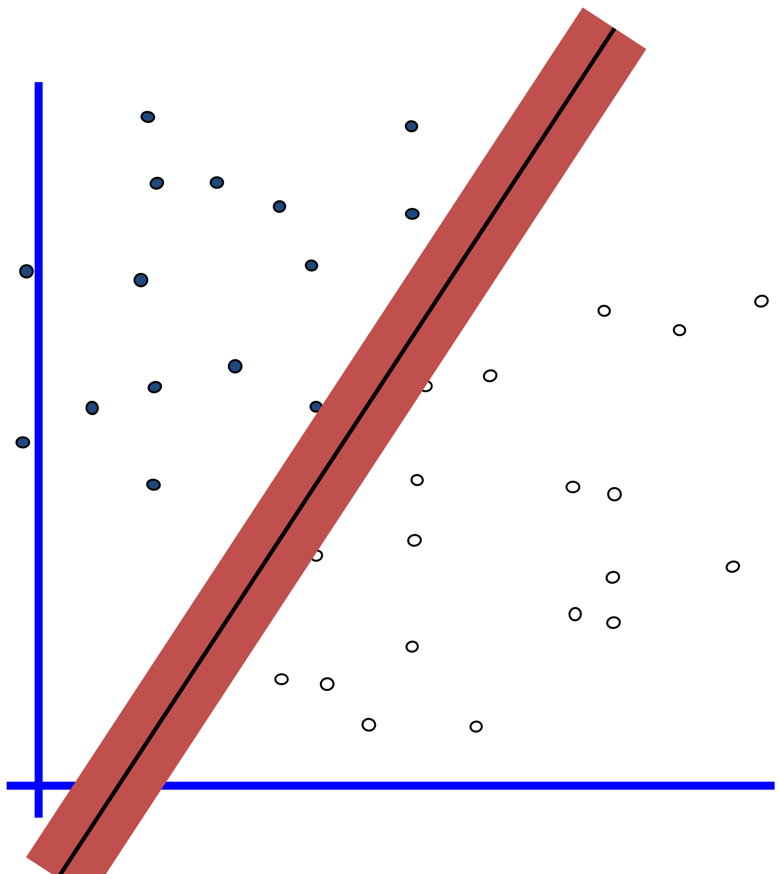
$$\boldsymbol{h(x,} \; \theta) = sign(\theta^T x)$$

Maximum Margin

- Class 1
- Class -1

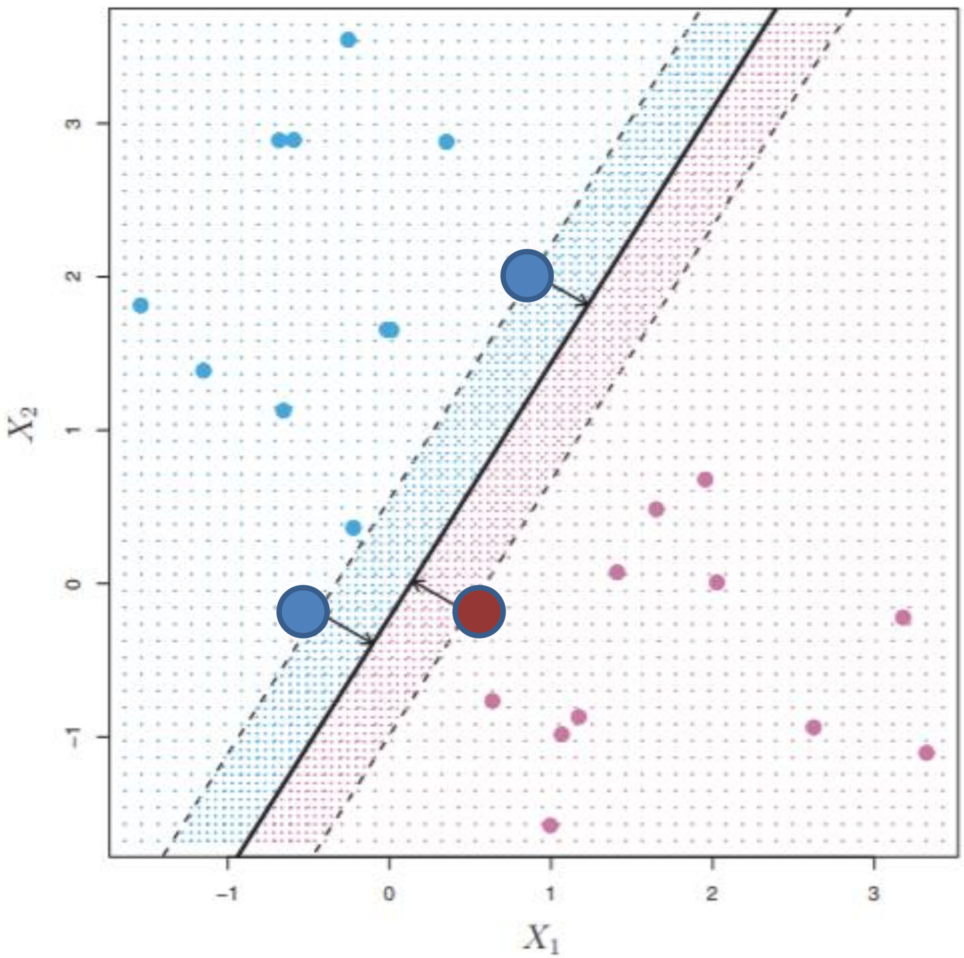


$\boldsymbol{f}(\boldsymbol{x}, \theta) = sign(\theta^T x)$



The maximum margin linear classifier is the linear classifier with the maximum margin!

Classifier margin



- Support vectors are “closest” to hyperplane
- At least 2 support vectors (1 positive, 1 negative)

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: $y_i \in \{-1,1\}$

max 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

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

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)

Equivalent formulation

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

- Min $\left|\left|\theta\right|\right|^2 + C\sum_i \epsilon_i$
- $y^{(i)}\left(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}\right) \geq 1 - \epsilon_i \; \forall i$
- $\epsilon_i \geq 0$
- When i is correctly classified, $y^{(i)} h\left(x^{(i)}\right) \geq 1$
- When i is not correctly classified

  $1 - y^{(i)} h\left(x^{(i)}\right) \leq \epsilon_i$

- $\max\left(0{,}1 - y^{(i)} h\left(x^{(i)}\right)\right) \leq \epsilon_i$

Hinge Loss

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

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

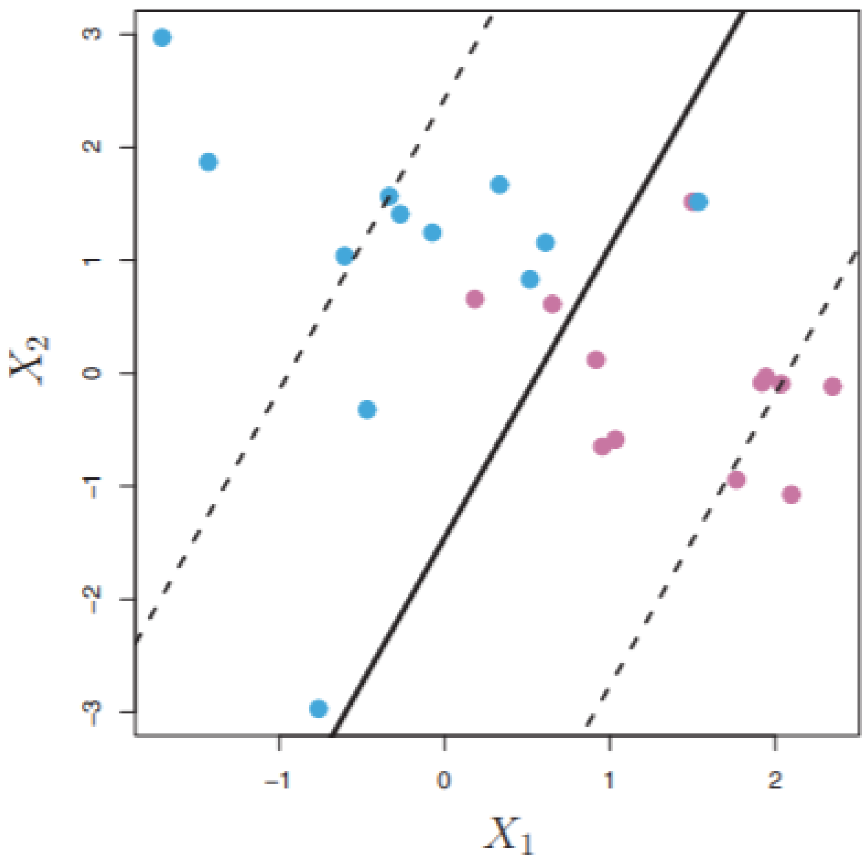
Total Error Budget

Regularization Term

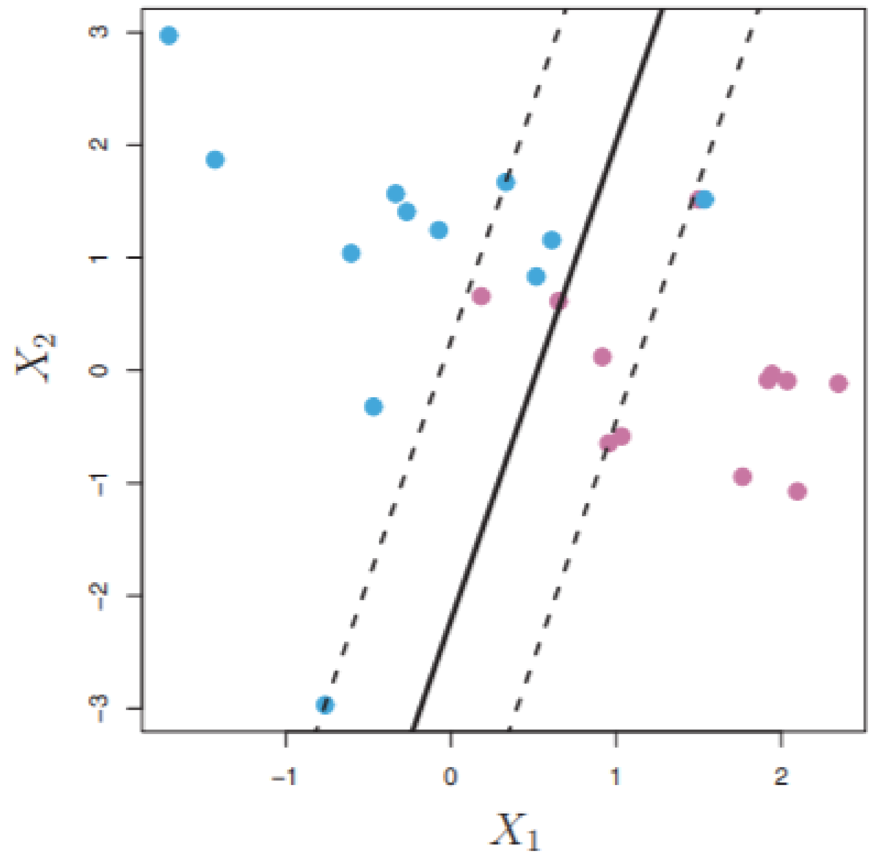
$$J(\theta) = C \sum_{i=0}^{n} \max\left(0{,}1 - y^{(i)} h\left(x^{(i)}\right)\right) + \sum_{j=1}^{d} \theta_j^2$$

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

Error Budget and Margin



Larger C
Low variance



Smaller C
Over-fitting

Find best hyper-parameter C by cross-validation

Non-linear decision

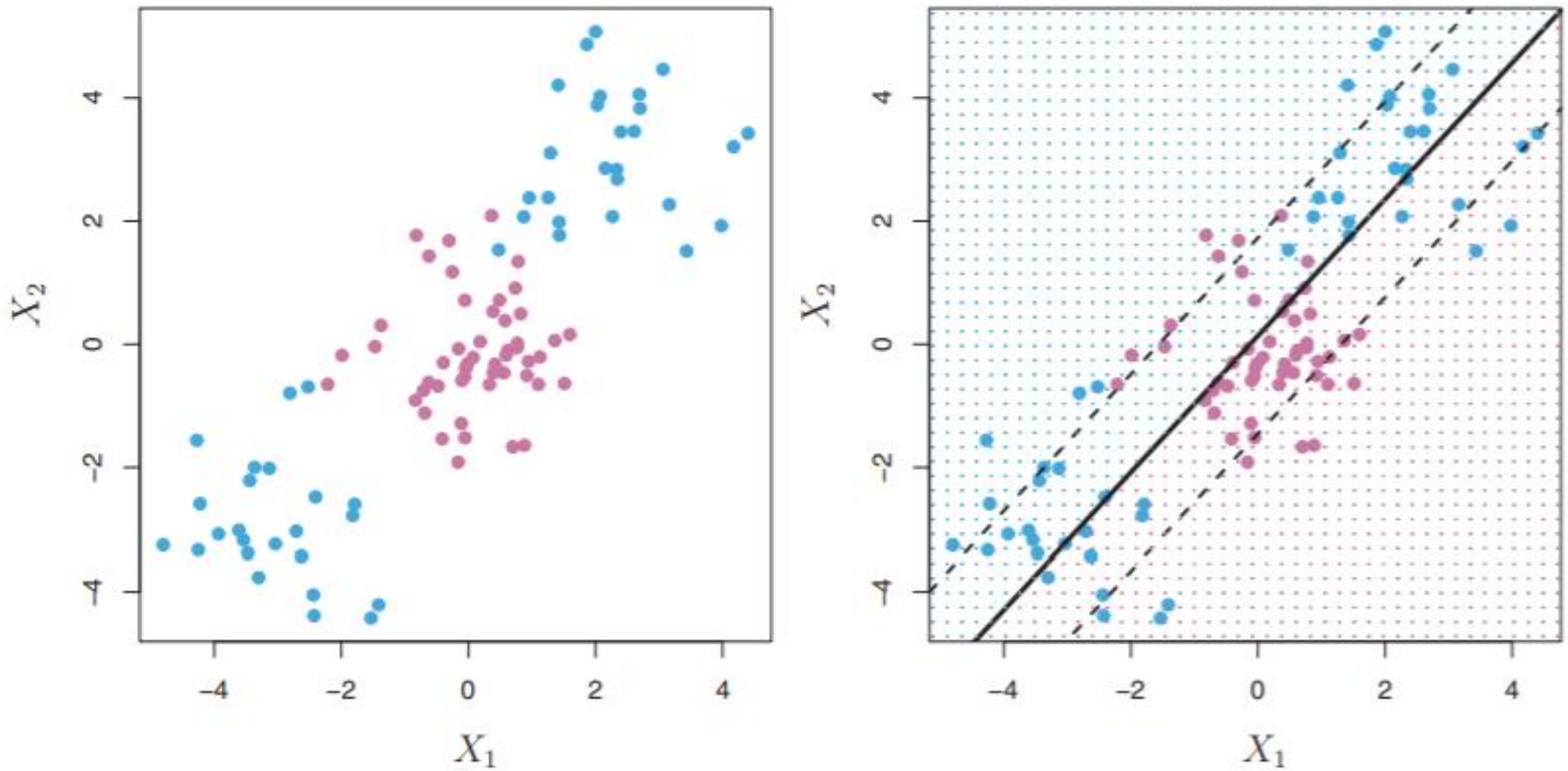
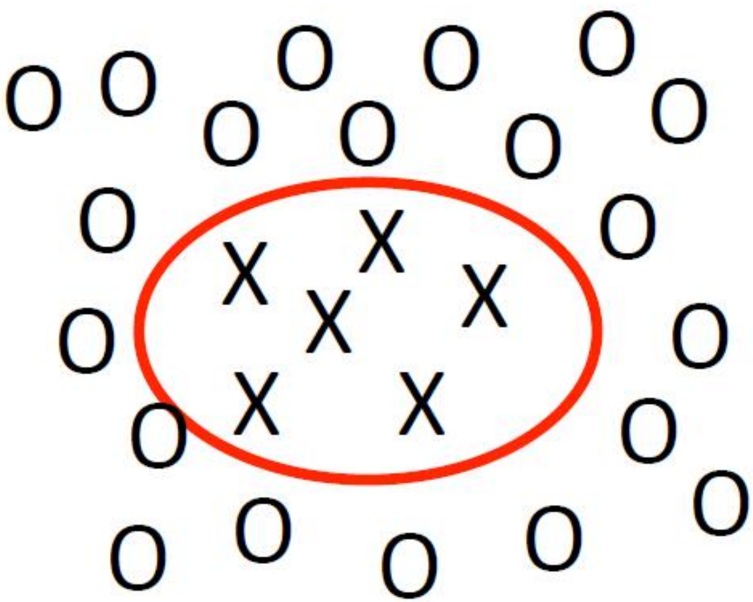


FIGURE 9.8. Left: *The observations fall into two classes, with a non-linear boundary between them.* Right: *The support vector classifier seeks a linear boundary, and consequently performs very poorly.*

More examples



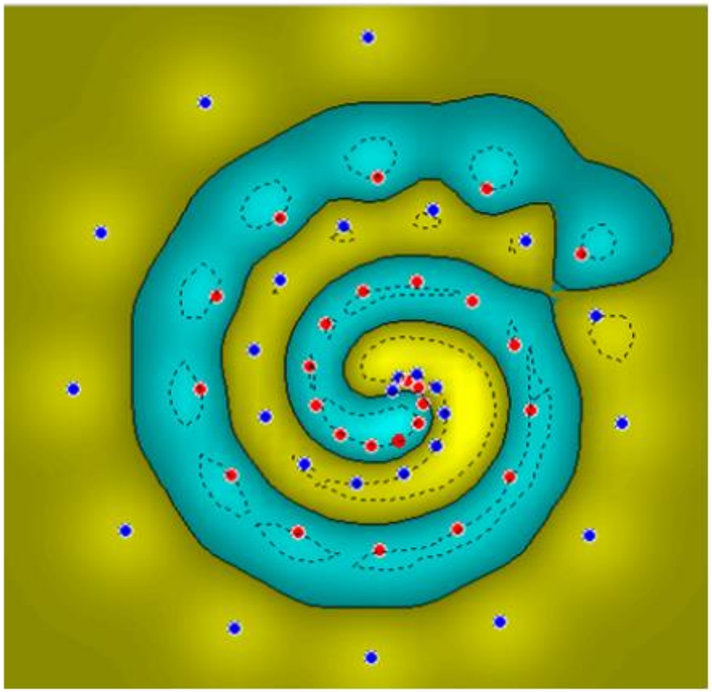


Image from http://www.atrandomresearch.com/iclass/

Kernels

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

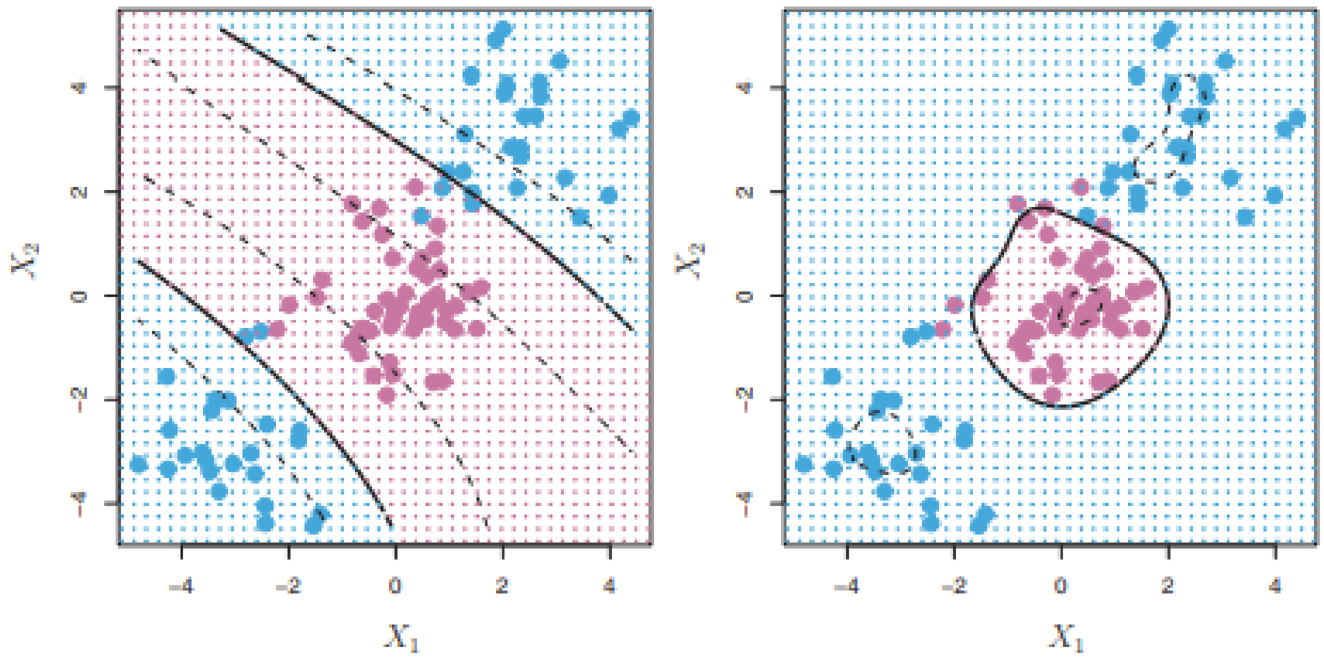


FIGURE 9.9. Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

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

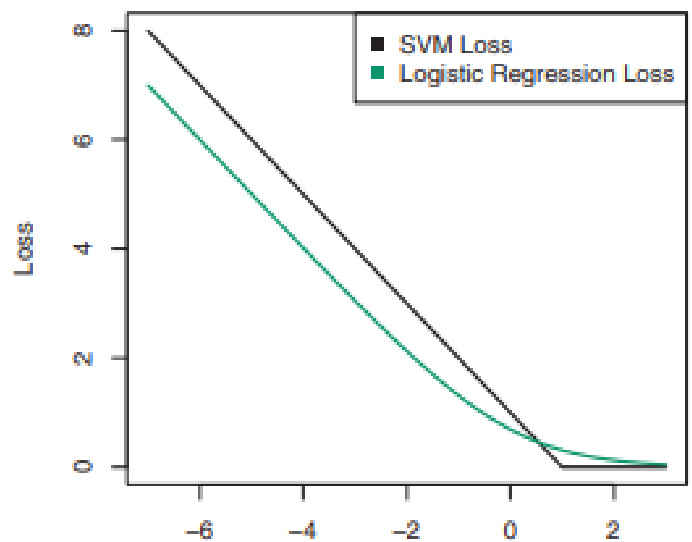
Comparing SVM with other classifiers

- 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

Connection to Logistic Regression

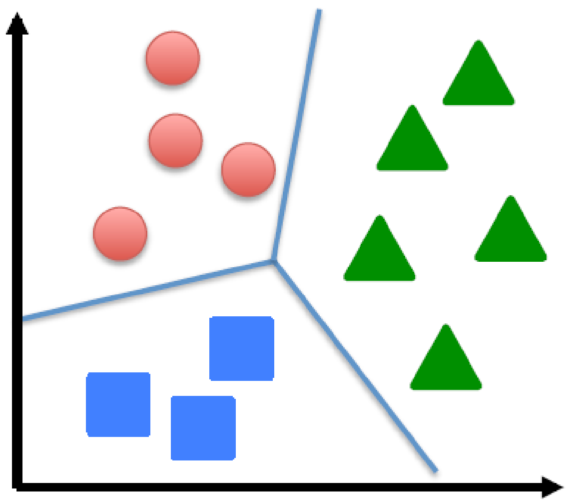
- $J(\theta) = \sum_{i=0}^{n} \underbrace{\max\left(0, 1 - y^{(i)} h\left(x^{(i)}\right)\right)}_{\text{Hinge loss}} + \lambda \sum_{j=1}^{d} \theta_j^2$

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



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

SVM for Multiple Classes

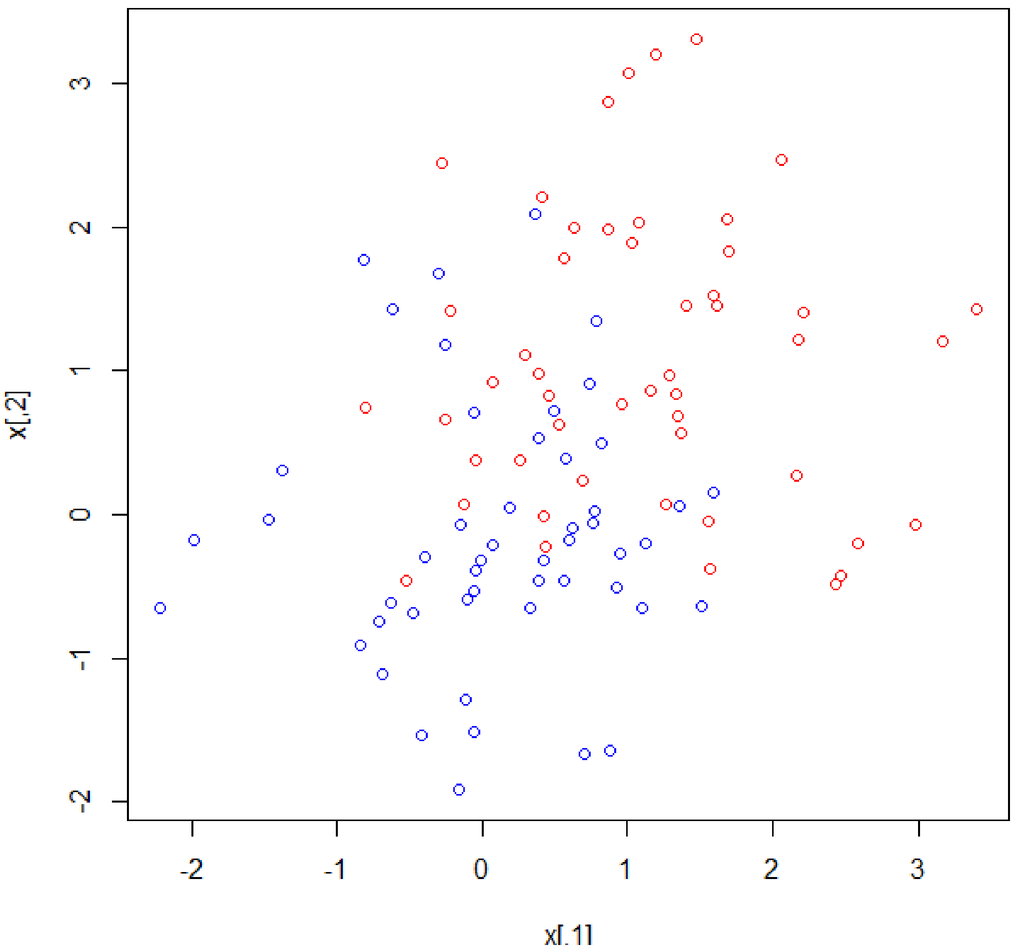


$y \in \{1, \ldots, K\}$

- 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 $(\boldsymbol{\theta}^{(i)})^\intercal \mathbf{x}$

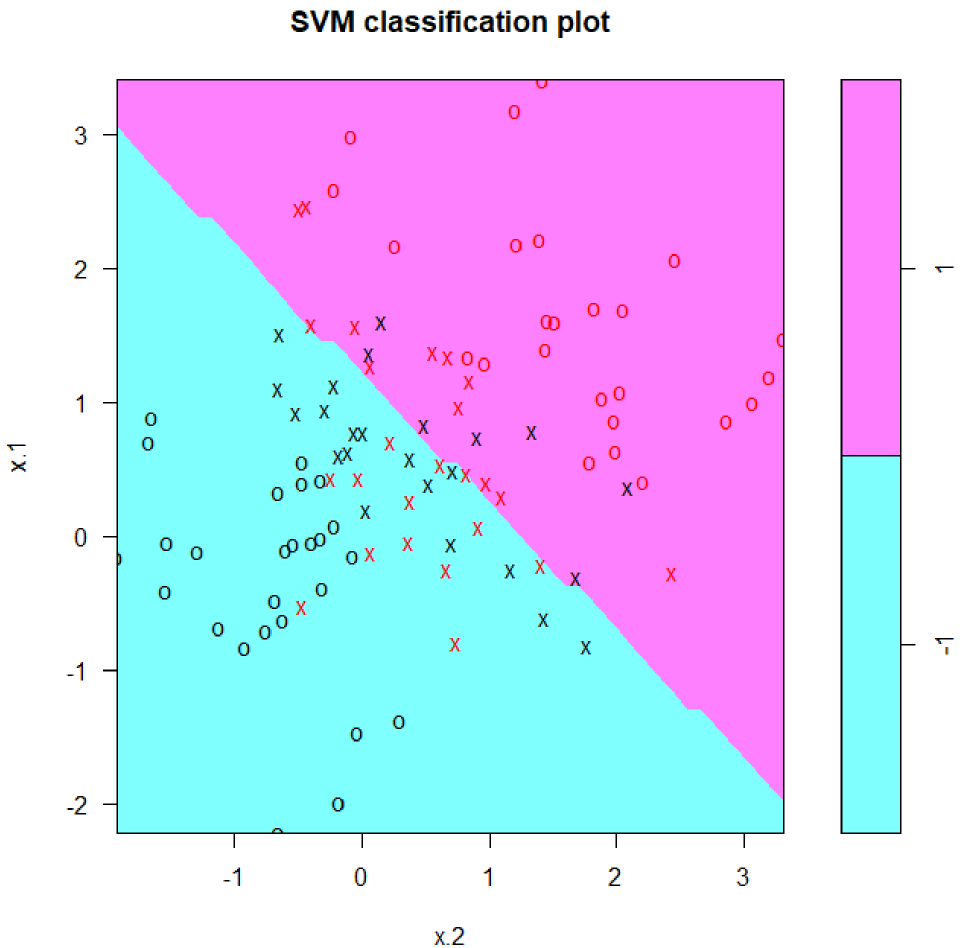
Lab – Linear SVM

```
> 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)
```



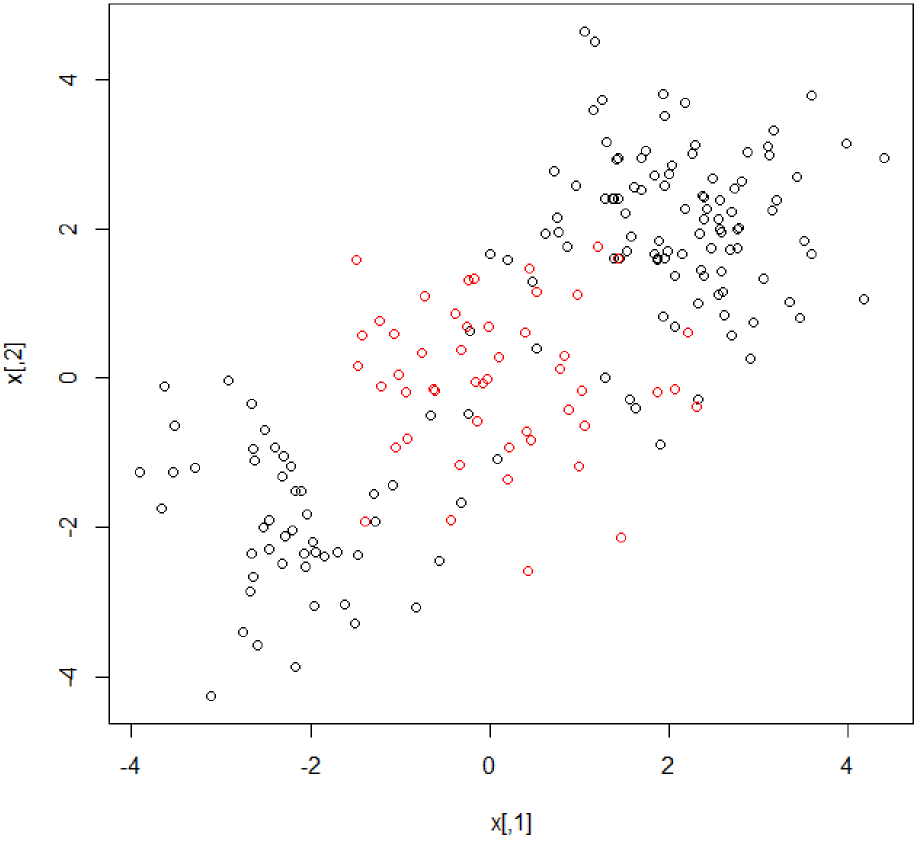
Lab – Linear SVM

```
> 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
```

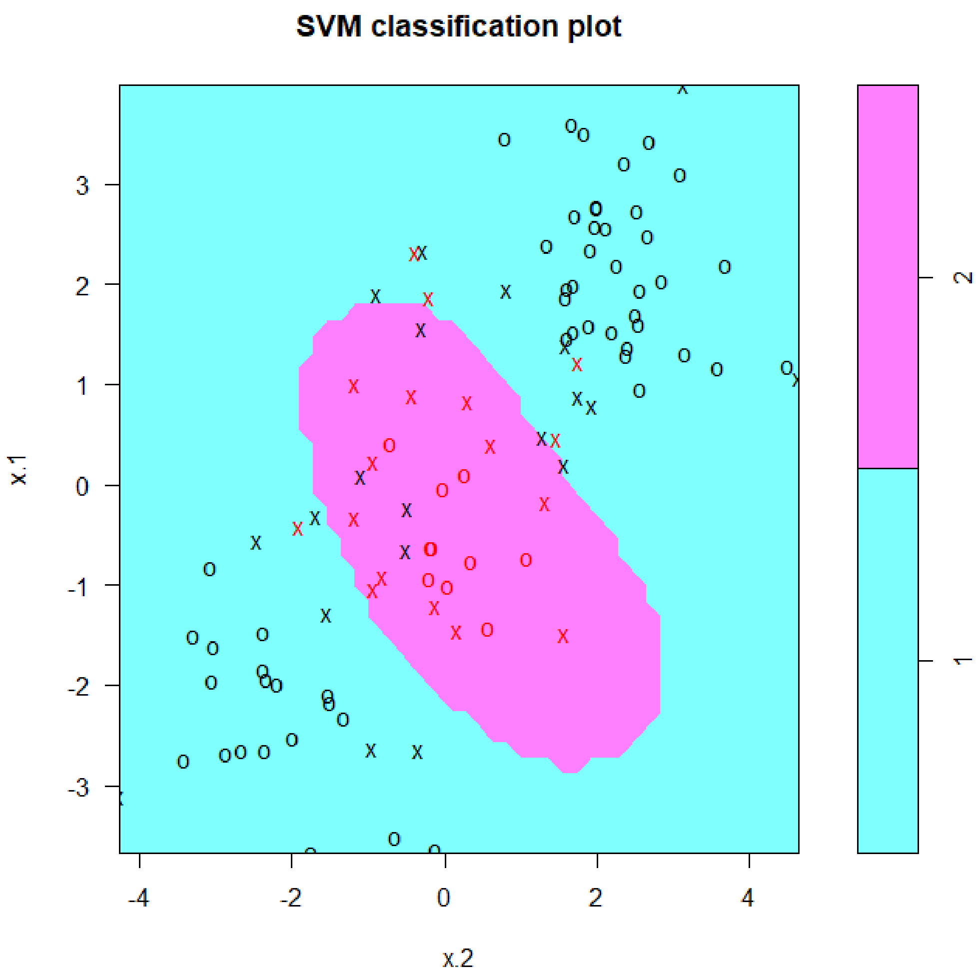
Lab – Radial SVM

```
>
> 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)
```



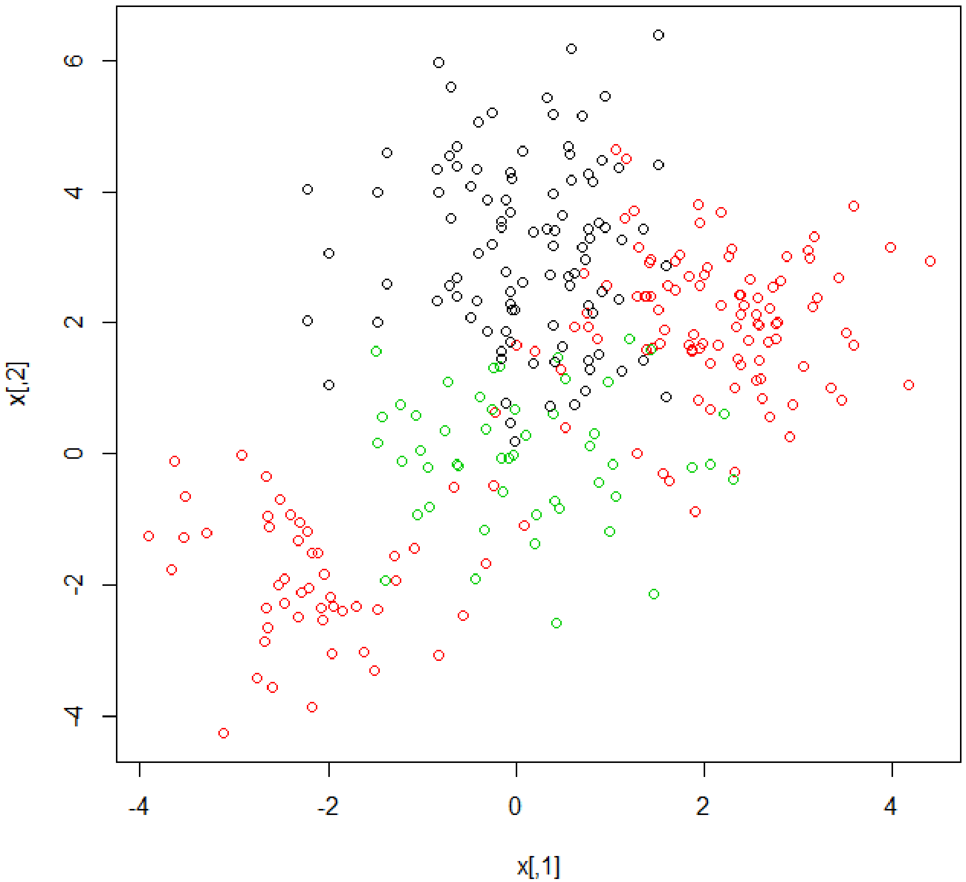
Lab – Radial SVM

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



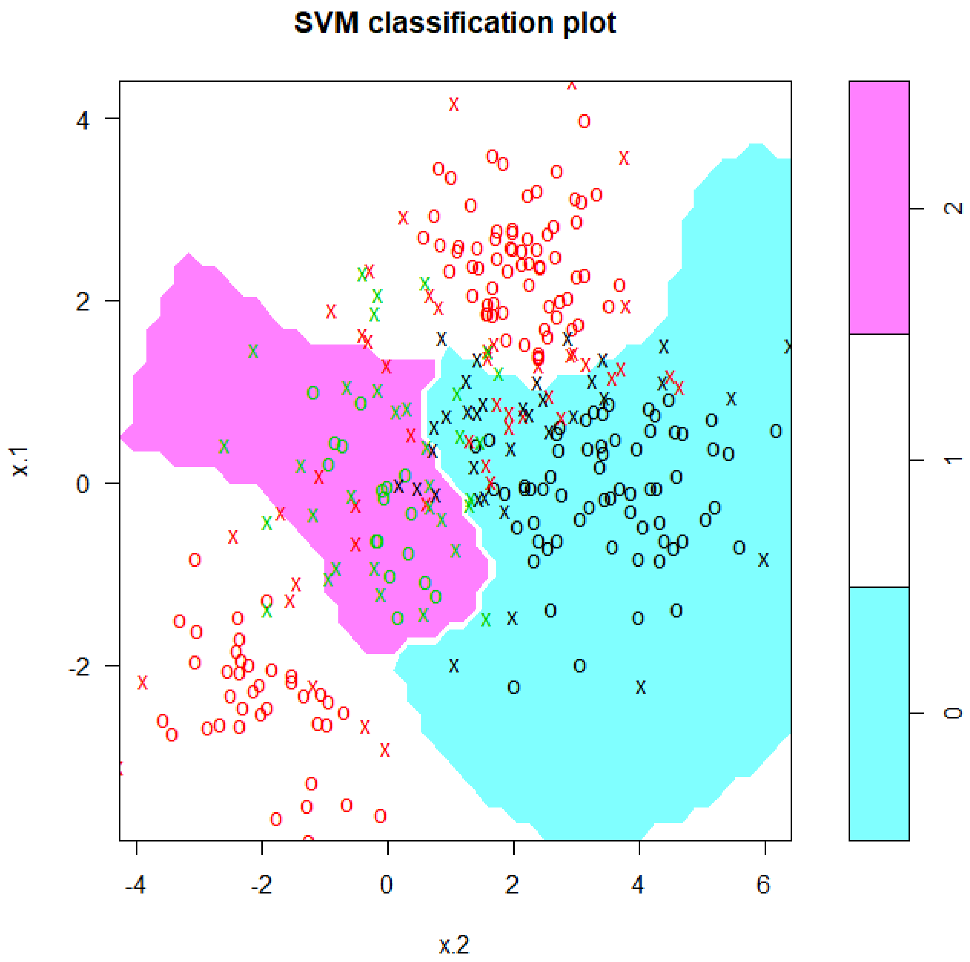
Lab – Multiple Classes

```
>
> 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)
>
```



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)

Outline

- Support vector classifier
  - Review
  - Hinge loss
- SVM
  - Non-linear decision boundaries
  - Kernels
  - Polynomial and Radial SVM
- Density estimators

Essential probability concepts

- Marginalization: $P(B) = \sum_{v \in \text{values}(A)} P(B \wedge A = v)$

- Conditional Probability: $P(A \mid B) = \dfrac{P(A \wedge B)}{P(B)}$

- Bayes' Rule: $P(A \mid B) = \dfrac{P(B \mid A) \times P(A)}{P(B)}$

- Independence:

$$
\begin{aligned}
A \perp\!\!\!\perp B \quad &\leftrightarrow \quad P(A \wedge B) = P(A) \times P(B) \\
&\leftrightarrow \quad P(A \mid B) = P(A) \\
A \perp\!\!\!\perp B \mid C \quad &\leftrightarrow \quad P(A \wedge B \mid C) = P(A \mid C) \times P(B \mid C)
\end{aligned}
$$

Prior and Joint Probabilities

- **Prior probability**: degree of belief without any other evidence
- **Joint probability**: matrix of combined probabilities of a set of variables

Russell & Norvig's Alarm Domain: (boolean RVs)

- A world has a specific instantiation of variables:
  $(\text{alarm} \wedge \text{burglary} \wedge \neg\text{earthquake})$
- The joint probability is given by:

$P(\text{Alarm}, \text{Burglary}) =$

	alarm	¬alarm
burglary	0.09	0.01
¬burglary	0.1	0.8

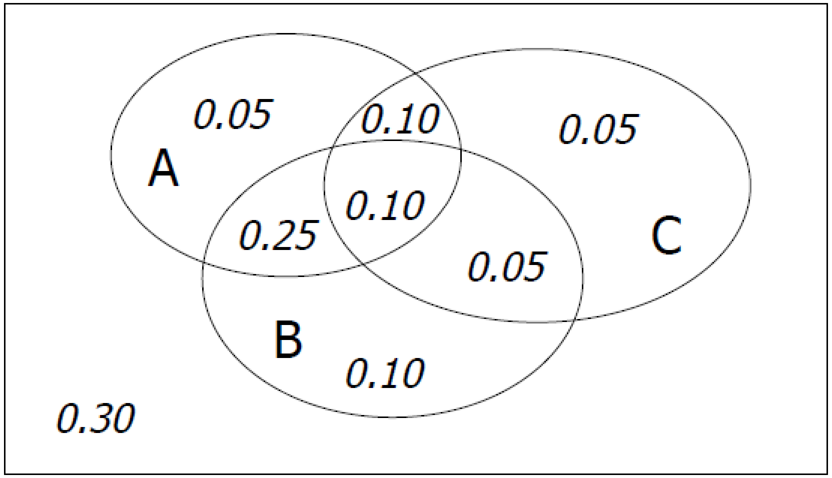
The Joint Distribution

Recipe for making a joint distribution of d variables:

1. Make a truth table listing all combinations of values of your variables (if there are d Boolean variables then the table will have 2^d rows).
2. For each combination of values, say how probable it is.
3. If you subscribe to the axioms of probability, those numbers must sum to 1.

e.g., Boolean variables A, B, C

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10



Computing Prior Probabilities

	alarm		¬alarm	
	earthquake	¬earthquake	earthquake	¬earthquake
burglary	0.01	0.08	0.001	0.009
¬burglary	0.01	0.09	0.01	0.79

Learning Joint Distributions

Step 1:

Build a JD table for your attributes in which the probabilities are unspecified

A	B	C	Prob
0	0	0	?
0	0	1	?
0	1	0	?
0	1	1	?
1	0	0	?
1	0	1	?
1	1	0	?
1	1	1	?

Step 2:

Then, fill in each row with:

$$\hat{P}(\text{row}) = \frac{\text{records matching row}}{\text{total number of records}}$$

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	**0.25**
1	1	1	0.10

Fraction of all records in which A and B are true but C is false

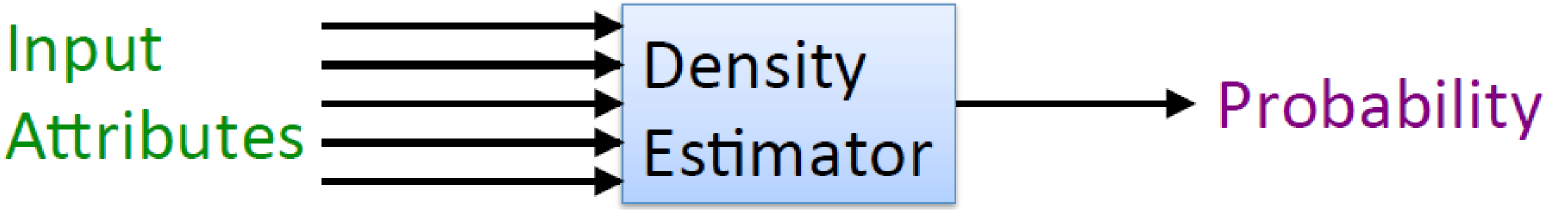
Example – Learning Joint Probability Distribution

This Joint PD was obtained by learning from three attributes in the UCI "Adult" Census Database [Kohavi 1995]

gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

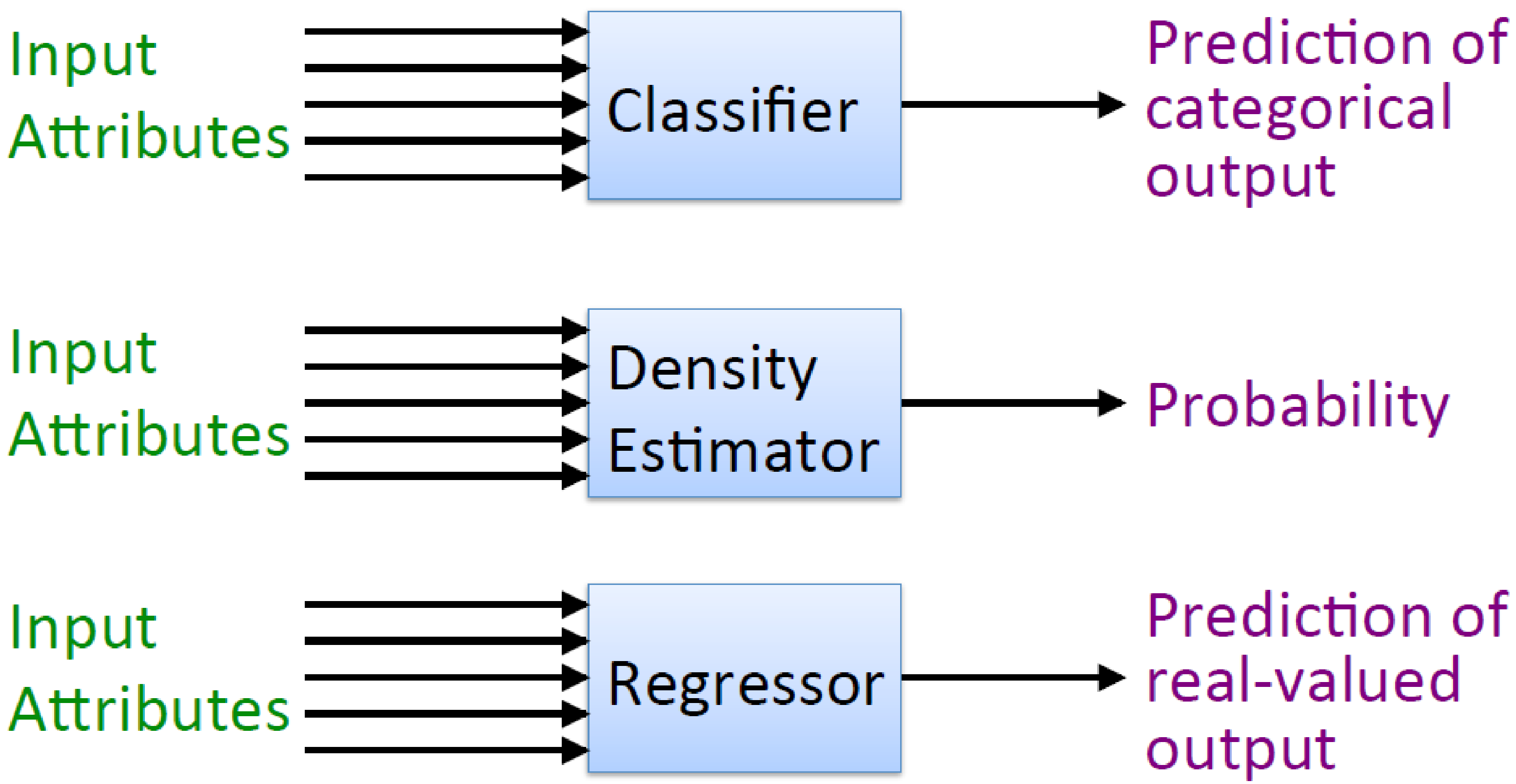
Density Estimation

- Our joint distribution learner is an example of something called **Density Estimation**
- A Density Estimator learns a mapping from a set of attributes to a probability

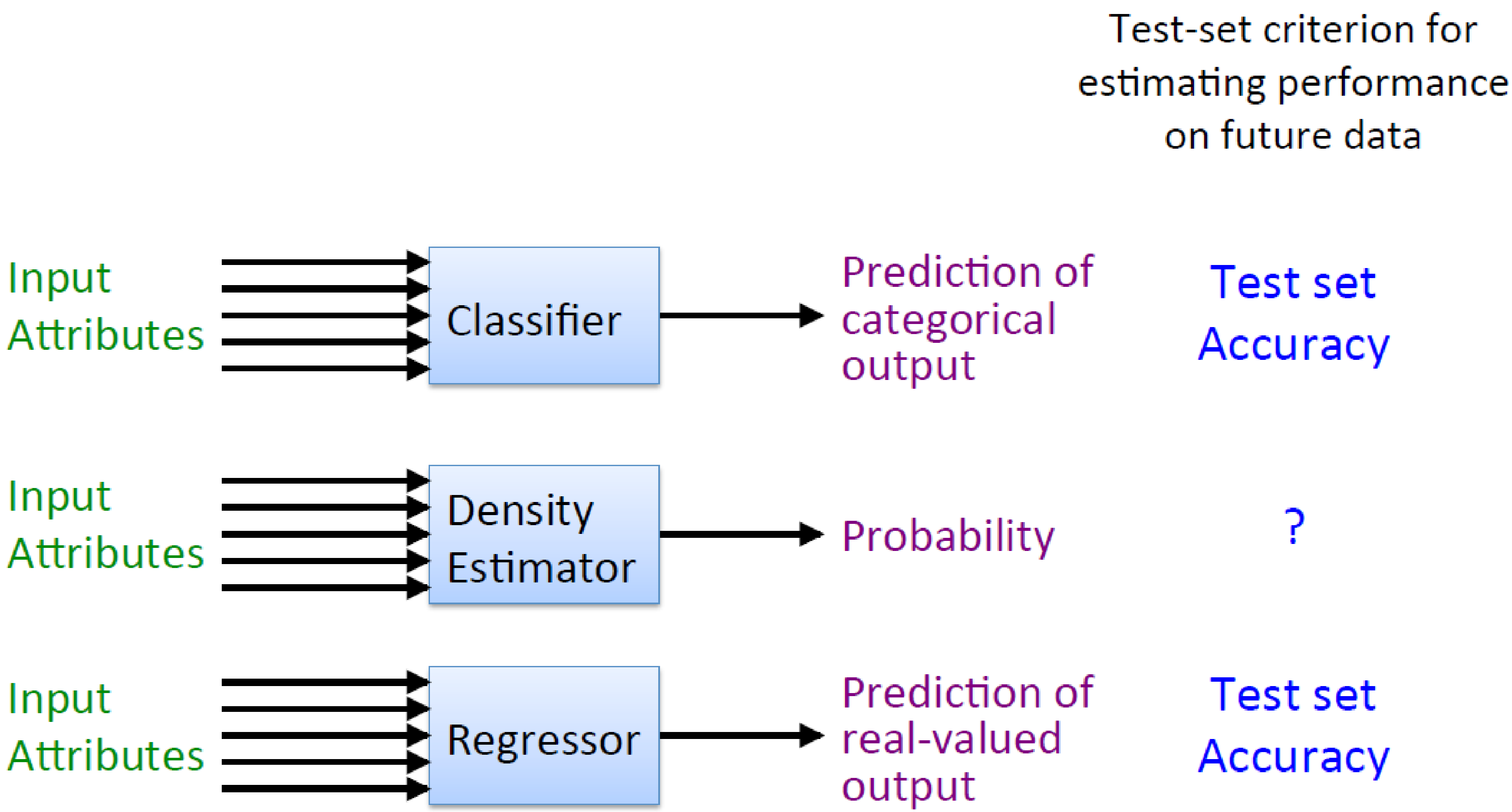


Density Estimation

Compare it against the two other major kinds of models:



Evaluating Density Estimators



Evaluating Density Estimators

- Given a record $\mathbf{x}$, a density estimator *M* can tell you how likely the record is:

$$\hat{P}(\mathbf{x} \mid M)$$

- The density estimator can also tell you how likely the dataset is:
  - Under the assumption that all records were independently generated from the Density Estimator's JD (that is, i.i.d.)

$$\hat{P}(\underbrace{\mathbf{x}_1 \wedge \mathbf{x}_2 \wedge \ldots \wedge \mathbf{x}_n}_{\text{dataset}} \mid M) = \prod_{i=1}^{n} \hat{P}(\mathbf{x}_i \mid M)$$

Example

From the UCI repository (thanks to Ross Quinlan)

- 192 records in the training set

mpg	modelyear	maker
good	75to78	asia
bad	70to74	america
bad	75to78	europe
bad	70to74	america
bad	70to74	america
bad	70to74	asia
bad	70to74	asia
bad	75to78	america
:	:	:
:	:	:
:	:	:
bad	70to74	america
good	79to83	america
bad	75to78	america
good	79to83	america
bad	75to78	america
good	79to83	america
good	79to83	america
bad	70to74	america
good	75to78	europe
bad	75to78	europe

mpg	modelyear	maker	
bad	70to74	america	0.27551
		asia	0.0255102
		europe	0.0153061
	75to77	america	0.153061
		asia	0.0255102
		europe	0.0357143
	78to83	america	0.0561224
		asia	Never
		europe	Never
good	70to74	america	0.0102041
		asia	0.0306122
		europe	0.0459184
	75to77	america	0.0306122
		asia	0.0408163
		europe	0.0357143
	78to83	america	0.112245
		asia	0.0714286
		europe	0.0357143

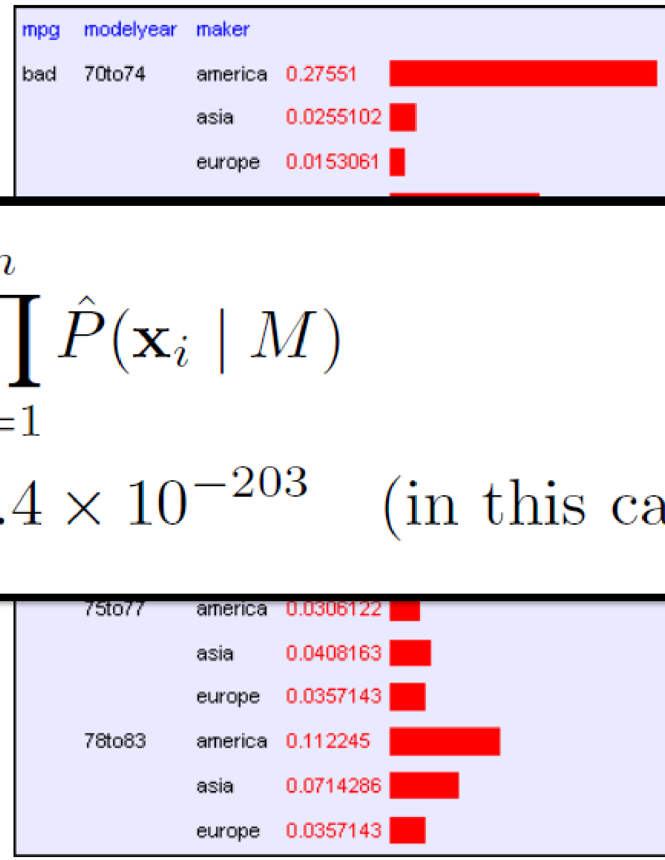
e by Andrew Moore

Example

From the UCI repository (thanks to Ross Quinlan)

- 192 records in the training set

mpg	modelyear	maker
good	75to78	asia
bad	70to74	america
bad	75to78	america
good	79to83	america
good	79to83	america
bad	70to74	america
good	75to78	europe
bad	75to78	europe



$$\hat{P}(\text{dataset} \mid M) = \prod_{i=1}^{n} \hat{P}(\mathbf{x}_i \mid M)$$

$$= 3.4 \times 10^{-203} \quad \text{(in this case)}$$

Slide by Andrew Moore

Log Probabilities

- For decent sized data sets, this product will underflow

$$\hat{P}(\text{dataset} \mid M) = \boxed{\prod_{i=1}^{n}} \hat{P}(\mathbf{x}_i \mid M)$$

- Therefore, since probabilities of datasets get so small, we usually use log probabilities

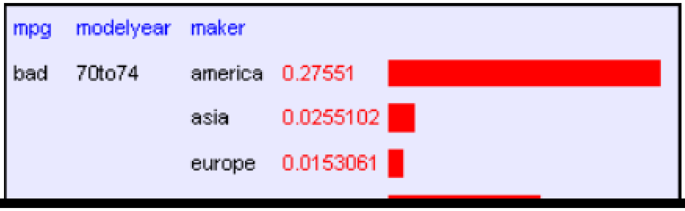
$$\log \hat{P}(\text{dataset} \mid M) = \log \prod_{i=1}^{n} \hat{P}(\mathbf{x}_i \mid M) = \sum_{i=1}^{n} \log \hat{P}(\mathbf{x}_i \mid M)$$

Example

From the UCI repository (thanks to Ross Quinlan)

- 192 records in the training set

mpg	modelyear	maker
good	75to78	asia
bad	70to74	america
bad	75to78	america
good	79to83	america
good	79to83	america
bad	70to74	america
good	75to78	europe
bad	75to78	europe



$$\log \hat{P}(\text{dataset} \mid M) = \sum_{i=1}^{n} \log \hat{P}(\mathbf{x}_i \mid M)$$

$$= -466.19 \quad \text{(in this case)}$$

Slide by Andrew Moore

Evaluation on Test Set

	Set Size	Log likelihood
Training Set	196	-466.1905
Test Set	196	-614.6157

- An independent test set with 196 cars has a much worse log-likelihood
  - Actually it's a billion quintillion quintillion quintillion quintillion times less likely
- Density estimators can overfit...

  ...and the full joint density estimator is the overfittiest of them all!

Overfitting

If this ever happens, the joint PDE learns there are certain combinations that are impossible

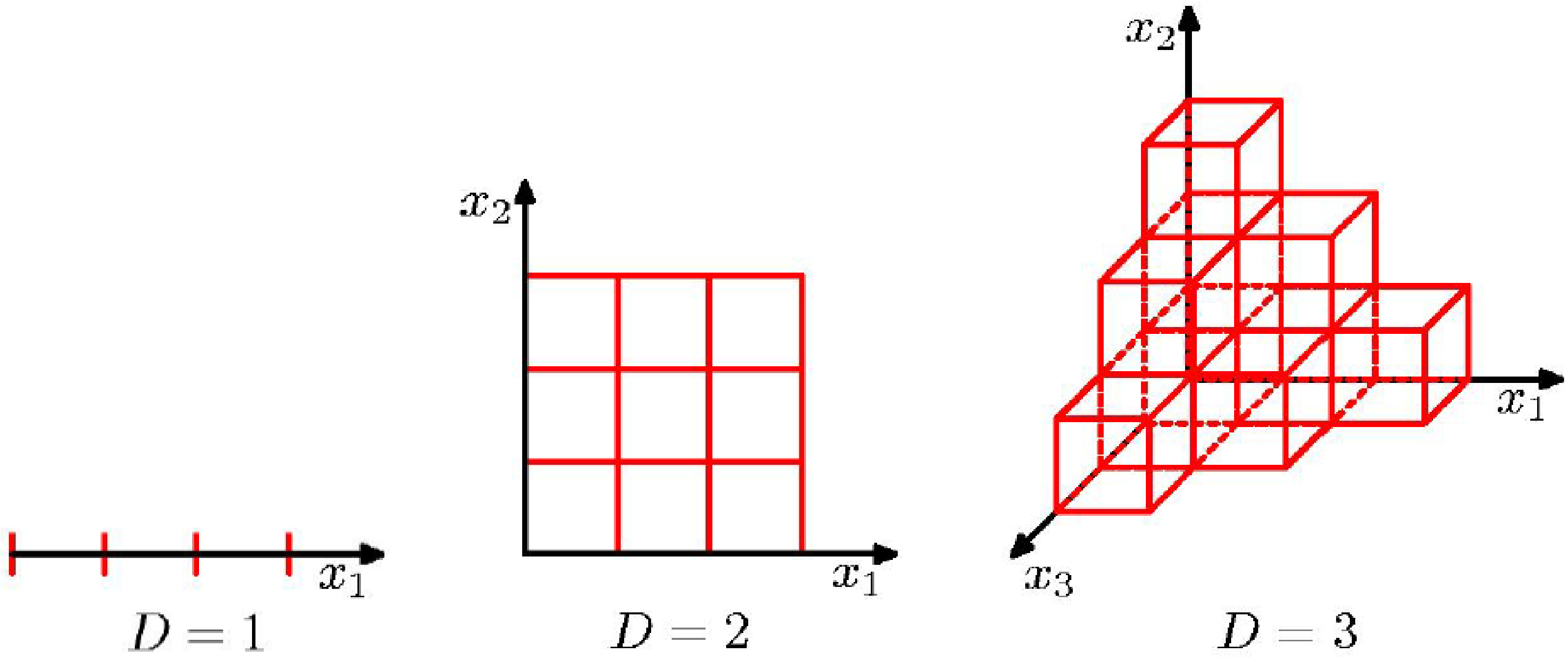
mpg	modelyear	maker	
bad	70to74	america	0.27551
		asia	0.0255102
		europe	0.0153061
	75to77	america	0.153061
		asia	0.0255102
		europe	0.0357143
	78to83	america	0.0561224
		asia	Never
		europe	Never
good	70to74	america	0.0102041
		asia	0.0306122

$$\log \hat{P}(\text{dataset} \mid M) = \sum_{i=1}^{n} \log \hat{P}(\mathbf{x}_i \mid M)$$

$$= -\infty \quad \text{if for any } i,\ \hat{P}(\mathbf{x}_i \mid M) = 0$$

de by Andrew Moore

Curse of Dimensionality



Pros and Cons of Density Estimators

- Pros
  - Density Estimators can learn distribution of training data
  - Can compute probability for a record
  - Can do inference (predict likelihood of record)
- Cons
  - Can overfit to the training data and not generalize to test data
  - Curse of dimensionality

Naïve Bayes classifier fixes these cons!

Acknowledgements

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