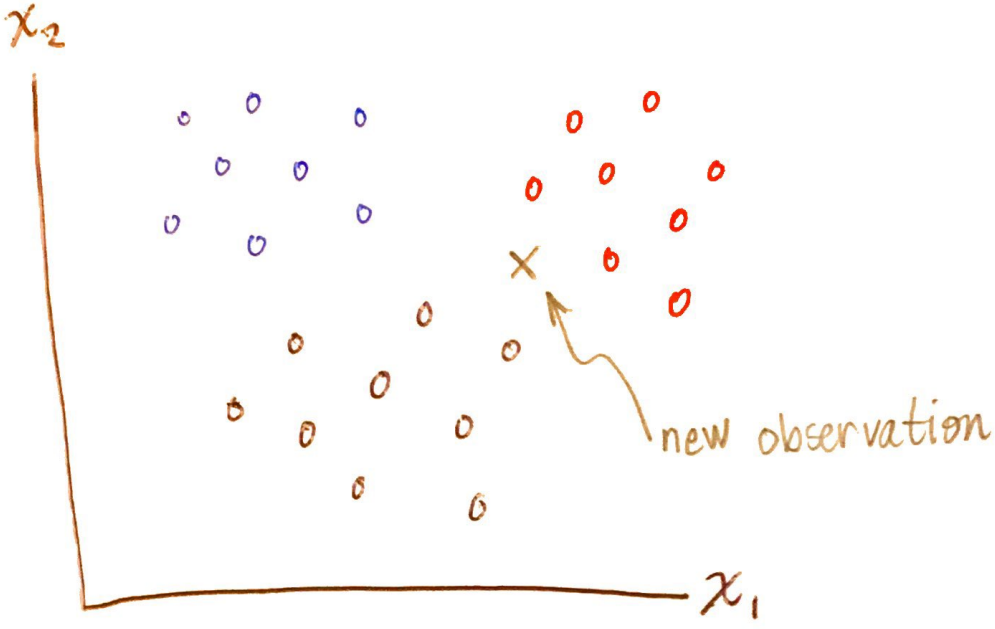
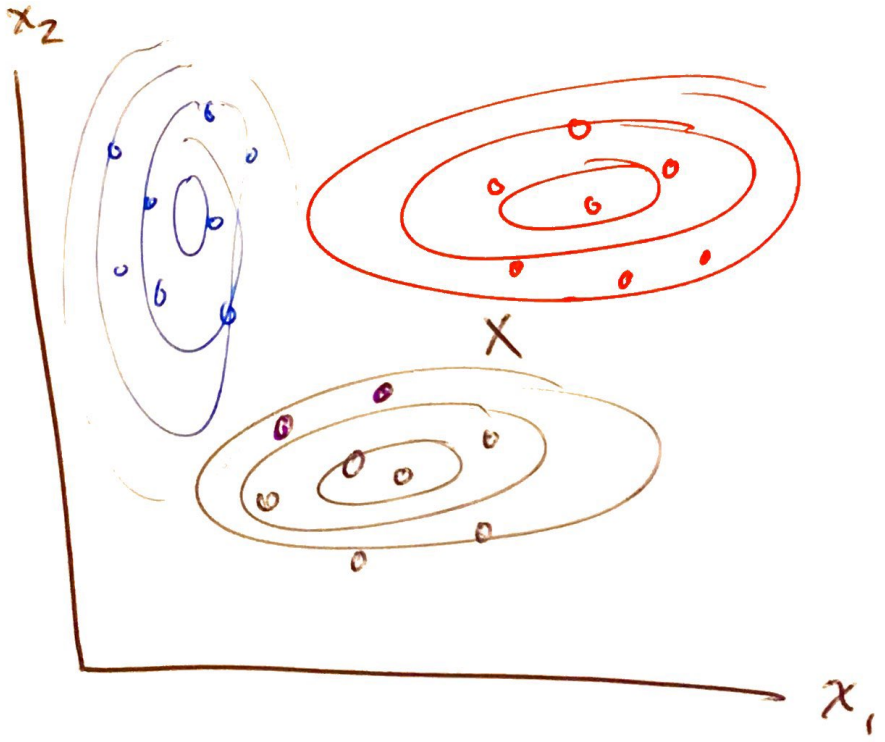
Naïve Bayes Classification



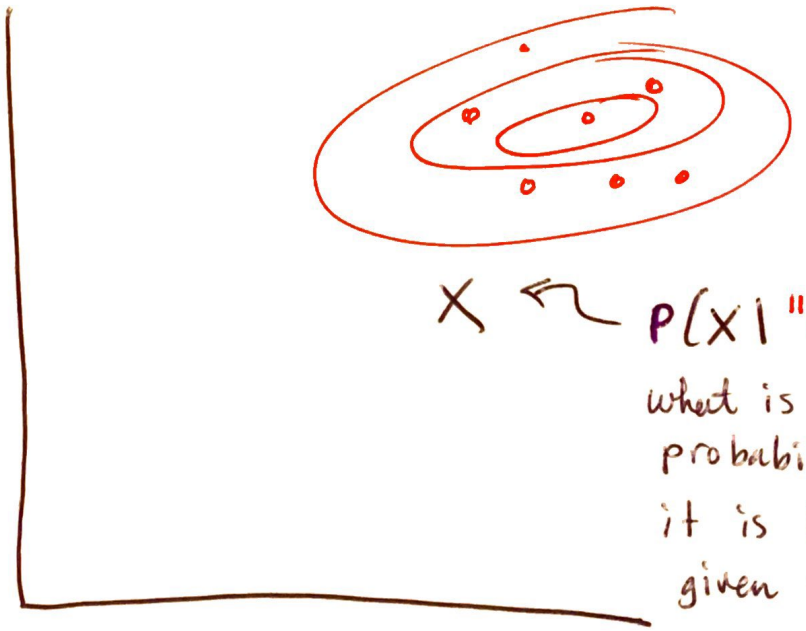
kNN and decision trees do not make any assumptions about the distribution of the underlying data → which might help w/ outliers.

A naïve Bayes classifier assumes that x_1 and x_2 are independent, but it at least assumes some distribution on x_1 and x_2.

Naive Bayes works by assuming that the observations are samples from distributions — a different distribution for each class.

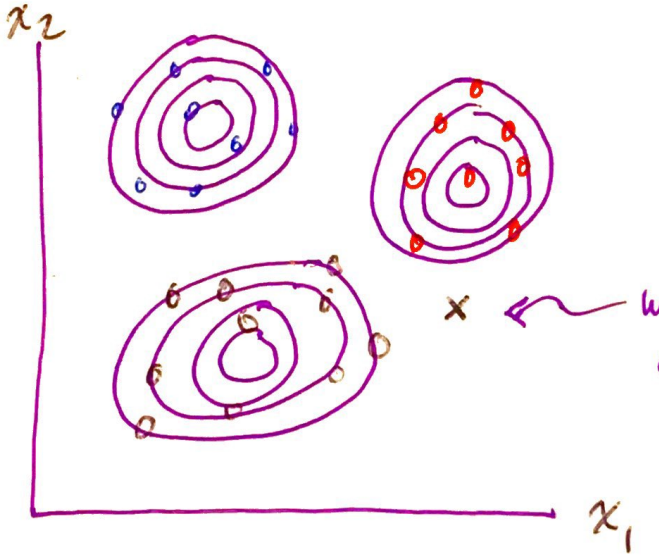


If we knew those distributions, we could determine the probability of a new observation occuring at a specific location, ~~under~~ under the assumption that it is "red"



- easy to calculate

Naive Bayes works by assuming that the observations are samples from distributions — a different distribution for each class.



← what is the probability of this being "red"

$P(x \mid \text{"red"})$

$P(\text{"red"} \mid x)$

vs.

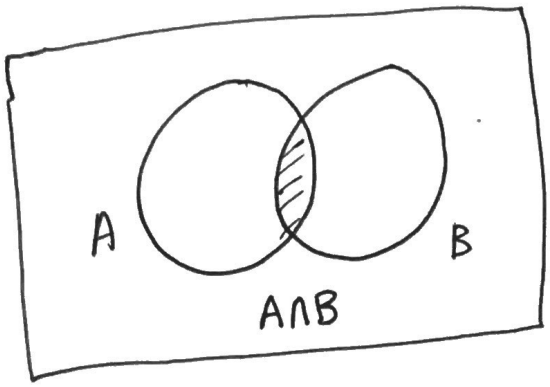
$P(\text{"blue"} \mid x)$

vs.

$P(\text{"black"} \mid x)$

Bayes Rule

to the rescue!



$$P(A|B) = \frac{P(A \cap B)}{P(B)} \qquad (*)$$

Likewise

$$P(B|A) = \frac{P(A \cap B)}{P(A)} \qquad (**)$$

Solve $(**)$ for $P(A \cap B)$

$$P(A \cap B) = P(B|A) \cdot P(A)$$

Subst. into $(*)$

what we know

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B}$$

what we want

Construct a naive Bayes classifier, using fitcnb

```
>> mdl = fitcnb(dataTrain, "group")
```

Recall setup:

```
groupData = readtable("groupData.csv")
groupData.group = categorical(groupData.group)
pt = cvpartition(groupData.group, "Holdout", 0.35)
dataTrain = groupData(training(pt), :)
dataTest = groupData(test(pt), :)
```

By default, the numeric predictors are modeled as normal distributions. Try changing the distribution to "kernel"

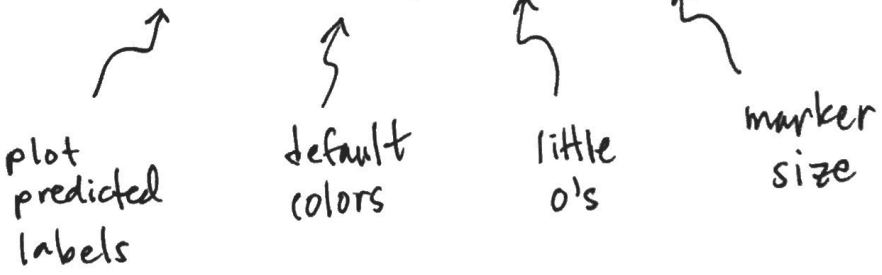
```
>> mdl = fitcnb(dataTrain, "group", ...
    "DistributionNames", "normal")
```

Plot the results

```
>> label = predict(mdl, dataTest)
>> gscatter(groupData.x, groupData.y, ...
                    groupData.group)

% now plot the predicted labels on top
  of the original data

>> hold on
>> gscatter(dataTest.x, dataTest.y, ...
                label, [], "O", 15)
```



Calculate error (loss)

~~predictedLabels = predict(mdl, dataTest);~~

```
>> err = loss(mdl, dataTest)
```

Classifier	Loss
Naive Bayes "kernel"	0.1254
Naive Bayes "normal"	0.1593
Tree	0.1259
kNN "k=10"	0.1171

Heart Disease Analysis

Read and load data:

```
>> heartData = readtable("heartDataNum.csv")
>> heartData.HeartDisease = categorical(...
                    heartData.HeartDisease)
```

Partition into training and test sets:

```
>> pt = cvpartition(heartData.HeartDisease,...
                    "HoldOut", 0.3)
>> hdTrain = heartData(training(pt),:)
>> hdTest = heartData(test(pt),:)
```

Create a naive Bayes classification model

```
>> mdl = fitcnb(hdTrain,"HeartDisease")
```

Calculate error

```
% training error

errTrain = resubLoss(mdl)

% test error

errTest = loss(mdl, hdTest)
```

Heart Disease - Numeric and Categorical

```
>> heartData = readtable("heartDataAll.csv")

% convert text labels to categorical arrays

>> heartData = convertvars(heartData, ...
    12:22, "categorical")

% partition the data into training and test
>> pt = cvpartition(heartData.HeartDisease,"HoldOut",0.3)
>> hdTrain = heartData(training(pt),:)
>> hdTest = heartData(test(pt),:)

% construct a naïve Bayes model on the
  categorical & numeric features

mdl = fitcnb(hdTrain, "HeartDisease")
```

% Use "kernel" for the numeric variables
(the first 11)

and "mvmn" multivariate, multinomial
for the categorical variables
(the last 10)

```
>> dists = [repmat("kernel", 1, 11)
            repmat("mvmn", 1, 10)]
```

```
dists =   1x21 string
  "kernel"  "kernel"  ...  "mvmn"  "mvmn"
```

```
>> mdl = fitnb(hdTrain, "HeartDisease", ...
        "DistributionNames", dists)
```

```
% training and testing error
>> errTrain = resubLoss(mdl)
>> errTest = loss(mdl, hdTest)
```