Machine Learning

K-Nearest Neighbours and Naive Bayes

Topics of previous lectures

- ✓ Ingredients of Machine Learning
- ✓ Basics of Classification
- ✓ Basic Linear Classifier

Topics of today's lecture

Classifiers:

- K-Nearest Neighbours
- Naive Bayes

Tennis dataset

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Tennis dataset

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain				
D7	Overca				
D8	Sunny				
D9	Sunny				
D10	Rain				
D11	Sunny				
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Let's leave these features out for now:

- Temp
- Humidity
- Wind

Tennis dataset

Day	Outlook	PlayTennis
D1	Sunny	No
D2	Sunny	No
D3	Overcast	Yes
D4	Rain	Yes
D5	Rain	Yes
D6	Rain	No
D7	Overcast	Yes
D8	Sunny	No
D9	Sunny	Yes
D10	Rain	Yes
D11	Sunny	Yes
D12	Overcast	Yes
D13	Overcast	Yes
D14	Rain	No

Tennis dataset

Now consider the classification task of predicting label **PlayTennis** from a single feature **Outlook**

Day	Outlook	PlayTennis
D1	Sunny	No
	Sunny	No
	Overcast	Yes
	Rain	Yes
	Rain	Yes
	Rain	No
	Overcast	Yes
D8	Sunny	No
D9	Sunny	Yes
D10	Rain	Yes
D11	Sunny	Yes
D12	Overcast	Yes
D13	Overcast	Yes
D14	Rain	No

Tennis dataset

Now consider the classification task of predicting label **PlayTennis** from a single feature **Outlook**

First, let's sort the rows by **Outlook**

Day	Outlook	PlayTennis
D1	Sunny	No
	Sunny	No
	Overcast	Yes
	Rain	Yes
	Rain	Yes
	Rain	No
	Overcast	Yes
D8	Sunny	No
D9	Sunny	Yes
	Rain	Yes
	Sunny	Yes
D12	Overcast	Yes
D13	Overcast	Yes
D14	Rain	No

Tennis dataset

Day	Outlook	PlayTennis
D3	Overcast	Yes
D7	Overcast	Yes
D12	Overcast	Yes
D13	Overcast	Yes
D4	Rain	Yes
D5	Rain	Yes
D6	Rain	No
D10	Rain	Yes
D14	Rain	No
D1	Sunny	No
D2	Sunny	No
D8	Sunny	No
D9	Sunny	Yes
D11	Sunny	Yes

Tennis dataset

Can we apply some learning algorithm that we already know?

Day	Outlook	PlayTennis
D3	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
D4	Rain	Yes
D5	Rain	Yes
D6	Rain	No
D10	Rain	Yes
D14	Rain	No
D1	Sunny	No
D2	Sunny	No
D8	Sunny	No
D9	Sunny	Yes
D11	Sunny	Yes

Tennis dataset

Day	Outlook	PlayTennis
D3	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
D4	Rain	Yes
D5	Rain	Yes
D6	Rain	No
D10	Rain	Yes
D14	Rain	No
	Sunny	No
	Sunny	No
	Sunny	No
	Sunny	Yes
	Sunny	Yes

Can we apply some learning algorithm that we already know?

No! Linear classifiers need numeric features! We must transform the data into numeric first!

Tennis dataset

Day	Outlook	PlayTennis
D3	Overcast	Yes
D7	Overcast	Yes
D12	Overcast	Yes
D13	Overcast	Yes
D4	Rain	Yes
D5	Rain	Yes
D6	Rain	No
D10	Rain	Yes
D14	Rain	No
D1	Sunny	No
D2	Sunny	No
D8	Sunny	No
D9	Sunny	Yes
D11	Sunny	Yes

Tennis dataset

Is this a good encoding?

- 0=Overcast
- 1=Sunny
- 2=Rain

Day	Outlook	PlayTennis
D3	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
	Rain	Yes
	Rain	Yes
	Rain	No
D10	Rain	Yes
D14	Rain	No
D1	Sunny	No
D2	Sunny	No
D8	Sunny	No
D9	Sunny	Yes
D11	Sunny	Yes

Tennis dataset

Day	Outlook	PlayTennis
D3	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
	Rain	Yes
	Rain	Yes
	Rain	No

Is this a good encoding?

- 0=Overcast
- 1=Sunny
- 2=Rain

No, it's not a good encoding!
No combination of weights results in predicting *PlayTennis=Yes* for *Overcast* and *Rain* and *PlayTennis=No* for *Sunny*.
If the weight for *Outlook* is positive, then changing *Outlook* from *Overcast* $\rightarrow$ *Sunny* $\rightarrow$ *Rain* increases the total score (and if weight is negative then decreases).

Better encoding (1-hot encoding)

Day	Outlook=Overcast	Outlook=Rain	Outlook=Sunny	PlayTennis
D3	1	0	0	Yes
D7	1	0	0	Yes
D12	1	0	0	Yes
D13	1	0	0	Yes
D4	0	1	0	Yes
D5	0	1	0	Yes
D6	0	1	0	No
D10	0	1	0	Yes
D14	0	1	0	No
D1	0	0	1	No
D2	0	0	1	No
D8	0	0	1	No
D9	0	0	1	Yes
D11	0	0	1	Yes

Better encoding (1-hot encoding)

Day	Outlook=Overcast	Outlook=Rain	Outlook=Sunny	PlayTennis
D3	1	0	0	Yes
D7	1	0	0	Yes
D12	1	0	0	Yes
D13	1	0	0	Yes
D4	0			Yes
D5	0			Yes
D6	0			No
D10	0			Yes
D14	0	1	0	No
D1	0	0	1	No
D2	0	0	1	No
D8	0	0	1	No
D9	0	0	1	Yes
D11	0	0	1	Yes

Now for linear models there would be a separate weight for each value of *Outlook*

One-hot encoding

- One-hot encoding: Introduce a binary (1/0) variable for each possible value of the categorical variable
- Using one-hot encoding is a standard method of transforming a nominal feature (unordered categorical feature) into numeric features
- For ordinal features (ordered categorical features, e.g. grades) it can also be used, but it would ignore the ordering

Are the 2 classes linearly separable?

Day	Outlook=Overcast	Outlook=Rain	Outlook=Sunny	PlayTennis
D3	1	0	0	Yes
D7	1	0	0	Yes
D12	1	0	0	Yes
D13	1	0	0	Yes
D4	0	1	0	Yes
D5	0	1	0	Yes
D6	0	1	0	No
D10	0	1	0	Yes
D14	0	1	0	No
D1	0	0	1	No
D2	0	0	1	No
D8	0	0	1	No
D9	0	0	1	Yes
D11	0	0	1	Yes

Are the 2 classes linearly separable?

Day	Outlook=Overcast	Outlook=Rain	Outlook=Sunny	PlayTennis
D3	1			Yes
D7	1			Yes
D12	1			Yes
D13	1			Yes
D4	0			Yes
D5	0	1	0	Yes
D6	0	1	0	No
D10	0	1	0	Yes
D14	0	1	0	No
D1	0	0	1	No
D2	0	0	1	No
D8	0	0	1	No
D9	0	0	1	Yes
D11	0	0	1	Yes

Not linearly separable!
For example, look at **D6, D10**
instances that have the same features
but belong to different classes.

Tennis dataset

Day	Outlook	PlayTennis
D3	Overcast	Yes
D7	Overcast	Yes
D12	Overcast	Yes
D13	Overcast	Yes
D4	Rain	Yes
D5	Rain	Yes
D6	Rain	No
D10	Rain	Yes
D14	Rain	No
D1	Sunny	No
D2	Sunny	No
D8	Sunny	No
D9	Sunny	Yes
D11	Sunny	Yes

Tennis dataset

If *Outlook*=*Rain*, what would you predict?

Day	Outlook	PlayTennis
D3	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
D4	Rain	Yes
D5	Rain	Yes
D6	Rain	No
D10	Rain	Yes
D14	Rain	No
D1	Sunny	No
D2	Sunny	No
D8	Sunny	No
D9	Sunny	Yes
D11	Sunny	Yes

Tennis dataset

If *Outlook=Rain*, what would you predict?

If *Outlook=Rain*, what would be the probability of *PlayTennis=Yes*?

Day	Outlook	PlayTennis
D3	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
	Rain	Yes
	Rain	Yes
	Rain	No
	Rain	Yes
	Rain	No
D1	Sunny	No
D2	Sunny	No
D8	Sunny	No
D9	Sunny	Yes
D11	Sunny	Yes

Tennis dataset

If *Outlook=Rain*, what would you predict?

If *Outlook=Rain*, what would be the probability of *PlayTennis=Yes*?

What kind of probability is it?

Day	Outlook	PlayTennis
D3	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
	Rain	Yes
	Rain	Yes
	Rain	No
	Rain	Yes
	Rain	No
	Sunny	No
	Sunny	No
	Sunny	No
D9	Sunny	Yes
D11	Sunny	Yes

Tennis dataset

Day	Outlook	PlayTennis
D3	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
	Rain	Yes
	Rain	Yes
	Rain	No
	Rain	Yes
	Rain	No
	Sunny	No
	Sunny	No
	Sunny	No
		Yes
D11	Sunny	Yes

If *Outlook=Rain*, what would you predict?

If *Outlook=Rain*, what would be the probability of *PlayTennis=Yes*?

What kind of probability is it?

Conditional Probability!

Conditional Probability

Day	Outlook	PlayTennis		Day	Outlook	PlayTennis
D3	Overcast	Yes		D6	Rain	No
D7	Overcast	Yes		D10	Rain	Yes
D12	Overcast	Yes		D14	Rain	No
D13	Overcast	Yes		D1	Sunny	No
D4	Rain	Yes		D2	Sunny	No
D5	Rain	Yes		D8	Sunny	No
D9	Sunny	Yes		D11	Sunny	Yes

Our intuition tells us to consider the following conditional probability.

$$P(PlayTennis = Yes|Outlook = Rain) =$$

Conditional Probability

Day	Outlook	PlayTennis		Day	Outlook	PlayTennis
D3	Overcast	Yes		D6	Rain	No
D7	Overcast	Yes		D10	Rain	Yes
D12	Overcast	Yes		D14	Rain	No
D13	Overcast	Yes		D1	Sunny	No
D4	Rain	Yes		D2	Sunny	No
D5	Rain	Yes		D8	Sunny	No
D9	Sunny	Yes		D11	Sunny	Yes

Our intuition tells us to consider the following conditional probability.

$$P(PlayTennis = Yes|Outlook = Rain) =$$
$$= \frac{P(PlayTennis = Yes, Outlook = Rain)}{P(Outlook = Rain)} =$$

Conditional Probability

Day	Outlook	PlayTennis		Day	Outlook	PlayTennis
D3	Overcast	Yes		D6	Rain	No
D7	Overcast	Yes		D10	Rain	Yes
D12	Overcast	Yes		D14	Rain	No
D13	Overcast	Yes		D1	Sunny	No
D4	Rain	Yes		D2	Sunny	No
D5	Rain	Yes		D8	Sunny	No
D9	Sunny	Yes		D11	Sunny	Yes

Our intuition tells us to consider the following conditional probability.

$$P(PlayTennis = Yes|Outlook = Rain) =$$

$$= \frac{P(PlayTennis = Yes, Outlook = Rain)}{P(Outlook = Rain)} = \frac{\frac{3}{14}}{\frac{5}{14}} = \frac{3}{5} = 0.6$$

Probabilistic Assumptions

- In classification, it is typically assumed that:
  - There is some unknown process that creates labelled instances for us
  - Each instance is created independently from others, by the same process
  - In statistical terms: we assume that the instances are i.i.d. (independent and identically distributed)

Probabilistic Assumptions

- In classification, it is typically assumed that:
  - There is some unknown process that creates labelled instances for us
  - Each instance is created independently from others, by the same process
  - In statistical terms: we assume that the instances are i.i.d. (independent and identically distributed)
- In previous calculations we have assumed even more:
  - The sample space (set of all possible outcomes) is the given dataset
  - In other words, the process which creates instances is just selecting a random row from the given dataset

Probabilistic Assumptions

- In classification, it is typically assumed that:
  - There is some unknown process that creates labelled instances for us
  - Each instance is created independently from others, by the same process
  - In statistical terms: we assume that the instances are i.i.d. (independent and identically distributed)
- In previous calculations we have assumed even more:
  - The sample space (set of all possible outcomes) is the given dataset
  - In other words, the process which creates instances is just selecting a random row from the given dataset
- Under these assumptions we can calculate probabilities by counting the proportion of instances that satisfy the respective condition

Tennis dataset

Is there a classifier that is always correct on this dataset?

Day	Outlook	PlayTennis
D3	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
D4	Rain	Yes
D5	Rain	Yes
D6	Rain	No
D10	Rain	Yes
D14	Rain	No
D1	Sunny	No
D2	Sunny	No
D8	Sunny	No
D9	Sunny	Yes
D11	Sunny	Yes

Optimal classifier on training data?

- The perfect classifier does not exist on Tennis dataset: (x=Rain,y=Yes) and (x=Rain,y=No) are both in the data but any classifier maps Rain to either Yes or No
- Remember that classifier is just a function of features: $\hat{y} = f(\mathbf{x})$
- Which is the optimal classifier on training data? In the sense that it makes the fewest errors

Optimal classifier on training data

Day	Outlook	PlayTennis
D3	Overcast	Yes
D7	Overcast	Yes
D12	Overcast	Yes
D13	Overcast	Yes
D4	Rain	Yes
D5	Rain	Yes
D6	Rain	No
D10	Rain	Yes
D14	Rain	No
D1	Sunny	No
D2	Sunny	No
D8	Sunny	No
D9	Sunny	Yes
D11	Sunny	Yes

Optimal classifier on training data

Optimal classifier on these data:

- $f(Overcast) = Yes$
- $f(Rain) = Yes$
- $f(Sunny) = No$

Day	Outlook	PlayTennis
D3	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
	Overcast	Yes
	Rain	Yes
	Rain	Yes
	Rain	No
D10	Rain	Yes
D14	Rain	No
D1	Sunny	No
D2	Sunny	No
D8	Sunny	No
D9	Sunny	Yes
D11	Sunny	Yes

Optimal classifier on training data

- In general, how to obtain the optimal classifier?

$$\hat{y} = f(\mathbf{x}) = ?$$

Optimal classifier on training data

- In general, how to obtain the optimal classifier?

$$\hat{y} = f(\mathbf{x}) = ?$$

- Proportion of rows with feature values $\mathbf{x}$ is:

$$P(\mathbf{X} = \mathbf{x})$$

Optimal classifier on training data

- In general, how to obtain the optimal classifier?

$$\hat{y} = f(\mathbf{x}) = ?$$

- Proportion of rows with feature values $\mathbf{x}$ is:

$$P(\mathbf{X} = \mathbf{x})$$

- Among those, proportion of positives:

$$P(Y = \oplus | \mathbf{X} = \mathbf{x})$$

- Among those, proportion of negatives:

$$P(Y = \ominus | \mathbf{X} = \mathbf{x})$$

Optimal classifier on training data

- If our model predicts positive: $\hat{y} = f(\mathbf{x}) = \oplus$
  Then it makes mistakes on the negatives:

$$P(Y = \ominus | \mathbf{X} = \mathbf{x})$$

Optimal classifier on training data

- If our model predicts positive: $\hat{y} = f(\mathbf{x}) = \oplus$
  Then it makes mistakes on the negatives:

$$P(Y = \ominus | \mathbf{X} = \mathbf{x})$$

- If our model predicts negative: $\hat{y} = f(\mathbf{x}) = \ominus$
  Then it makes mistakes on the positives:

$$P(Y = \oplus | \mathbf{X} = \mathbf{x})$$

Optimal classifier on training data

- If our model predicts positive: $\hat{y} = f(\mathbf{x}) = \oplus$
  Then it makes mistakes on the negatives:

$$P(Y = \ominus | \mathbf{X} = \mathbf{x})$$

- If our model predicts negative: $\hat{y} = f(\mathbf{x}) = \ominus$
  Then it makes mistakes on the positives:

$$P(Y = \oplus | \mathbf{X} = \mathbf{x})$$

- To minimize errors we should:
  - Predict $\oplus$ if $P(Y = \oplus | \mathbf{X} = \mathbf{x}) > P(Y = \ominus | \mathbf{X} = \mathbf{x})$
  - Predict $\ominus$ if $P(Y = \ominus | \mathbf{X} = \mathbf{x}) > P(Y = \oplus | \mathbf{X} = \mathbf{x})$

Maximum a posteriori (MAP)

Optimal is to follow the **maximum a posteriori (MAP)** decision rule

$$\hat{y} = f(\mathbf{x}) = \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(Y = y | \mathbf{X} = \mathbf{x})$$

That is, predict the class that has the highest probability conditional to the given feature values

Have we solved classification?

- We have just found how to calculate the optimal classifier – have we solved classification?

Have we solved classification?

- We have just found how to calculate the optimal classifier – have we solved classification?
- **No!** We made the assumption that instances are drawn randomly from the training data

Have we solved classification?

- We have just found how to calculate the optimal classifier – have we solved classification?
- **No!** We made the assumption that instances are drawn randomly from the training data
- When testing our classifier on new test data, we might encounter a different distribution of data and even new instances

Bayes classifier

- If we somehow magically know probabilities according to the true process that generates instances, then MAP-decision rule gives us **Bayes classifier**
- Bayes classifier is defined by:

$$\hat{y} = f(\mathbf{x}) = \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(Y = y | \mathbf{X} = \mathbf{x}),$$

where P is the probability measure defined by the true data-generating process

Train and Test data

- So far we have obtained the optimal classifier on our training data, but having a good classifier on training data is not sufficient

Train and Test data

- So far we have obtained the optimal classifier on our training data, but having a good classifier on training data is not sufficient
- The goal is to perform well on future data

Train and Test data

- So far we have obtained the optimal classifier on our training data, but having a good classifier on training data is not sufficient
- The goal is to perform well on future data
- How well would our classifier perform on future data?

Train and Test data

- So far we have obtained the optimal classifier on our training data, but having a good classifier on training data is not sufficient
- The goal is to perform well on future data
- How well would our classifier perform on future data?
  - We will never know exactly in advance, but we can estimate it by applying the classifier on a test dataset, which is separate from the training dataset

Train and Test data

- So far we have obtained the optimal classifier on our training data, but having a good classifier on training data is not sufficient
- The goal is to perform well on future data
- How well would our classifier perform on future data?
  - We will never know exactly in advance, but we can estimate it by applying the classifier on a test dataset, which is separate from the training dataset
- What if we don't have test data?

Train and Test data

- So far we have obtained the optimal classifier on our training data, but having a good classifier on training data is not sufficient
- The goal is to perform well on future data
- How well would our classifier perform on future data?
  - We will never know exactly in advance, but we can estimate it by applying the classifier on a test dataset, which is separate from the training dataset
- What if we don't have test data?
  - We can still estimate how well our learning algorithm works by splitting the original training data randomly into training and test data

Random split into train and test sets

Day	Outlook	PlayTennis	Training
D1	Sunny	No	
D2	Sunny	No	
D4	Rain	Yes	
D6	Rain	No	
D7	Overcast	Yes	
D12	Overcast	Yes	
D14	Rain	No	
D3	Overcast	Yes	Testing
D5	Rain	Yes	
D8	Sunny	No	
D9	Sunny	Yes	
D10	Rain	Yes	
D11	Sunny	Yes	
D13	Overcast	Yes	

Random split into train and test sets

Day	Outlook	PlayTennis	
D1	Sunny	No	Training
D2	Sunny	No	
D4	Rain	Yes	
D6	Rain	No	
D7	Overcast	Yes	
D12	Overcast	Yes	
D14	Rain	No	
D3	Overcast	Yes	Testing
D5	Rain	Yes	
D8	Sunny	No	
D9	Sunny	Yes	
D10	Rain	Yes	
D11	Sunny	Yes	
D13	Overcast	Yes	

Optimal classifier on training data:

$f(Overcast) = Yes$

$f(Rain) = No$

$f(Sunny) = No$

Train-optimal classifier makes 4 errors: D5,D9,D10,D11

Bad performance on test data

- Our train-optimal classifier had 4/7 error rate on the test data – why so bad?
  - We might have been unlucky with how the training and test data were split
  - The dataset is small - a bigger dataset might have allowed us to learn more successfully

Bad performance on test data

- Our train-optimal classifier had 4/7 error rate on the test data – why so bad?
  - We might have been unlucky with how the training and test data were split
  - The dataset is small - a bigger dataset might have allowed us to learn more successfully

Let's add another feature from the initial dataset

We have 2 features now

Day	Outlook	Temp	PlayTennis
D7	Overcast	Cool	Yes
D3	Overcast	Hot	Yes
D13	Overcast	Hot	Yes
D12	Overcast	Mild	Yes
D5	Rain	Cool	Yes
D6	Rain	Cool	No
D4	Rain	Mild	Yes
D10	Rain	Mild	Yes
D14	Rain	Mild	No
D9	Sunny	Cool	Yes
D1	Sunny	Hot	No
D2	Sunny	Hot	No
D8	Sunny	Mild	No
D11	Sunny	Mild	Yes

We have 2 features now

Day	Outlook	Temp	PlayTennis
D7	Overcast	Cool	Yes
D3	Overcast	Hot	Yes
D13	Overcast	Hot	Yes
D12	Overcast	Mild	Yes
D5	Rain	Cool	Yes
D6	Rain	Cool	No
D4	Rain	Mild	Yes
D10	Rain	Mild	Yes
D14	Rain	Mild	No
D9	Sunny	Cool	Yes
D1	Sunny	Hot	No
D2	Sunny	Hot	No
D8	Sunny	Mild	No
D11	Sunny	Mild	Yes

Optimal classifier
on these data:

$f(Overcast, Cool) = Yes$
$f(Overcast, Hot) = Yes$
$f(Overcast, Mild) = Yes$
$f(Rain, Cool) = Yes/No$
$f(Rain, Mild) = Yes$
$f(Sunny, Cool) = Yes$
$f(Sunny, Hot) = No$
$f(Sunny, Mild) = Yes/No$

We have 2 features now

Day	Outlook	Temp	PlayTennis
D7	Overcast	Cool	Yes
D3	Overcast	Hot	Yes
D13	Overcast	Hot	Yes
D12	Overcast	Mild	Yes
D5	Rain	Cool	Yes
D6	Rain	Cool	No
D4	Rain	Mild	Yes
D10	Rain	Mild	Yes
D14	Rain	Mild	No
D9	Sunny	Cool	Yes
D1	Sunny	Hot	No
D2	Sunny	Hot	No
D8	Sunny	Mild	No
D11	Sunny	Mild	Yes

Optimal classifier
on these data:

$f(Overcast, Cool) = Yes$
$f(Overcast, Hot) = Yes$
$f(Overcast, Mild) = Yes$
$f(Rain, Cool) = Yes/No$
$f(Rain, Mild) = Yes$
$f(Sunny, Cool) = Yes$
$f(Sunny, Hot) = No$
$f(Sunny, Mild) = Yes/No$

$f(Rain, Hot) =?$

Problem with train-optimal classifier

- Our train-optimal classifer does not define what to predict for $f(Rain, Hot)$

Problem with train-optimal classifier

- Our train-optimal classifer does not define what to predict for $f(Rain, Hot)$
- If an instance with these feature values occurs in future data then we would need to predict randomly

Problem with train-optimal classifier

- Our train-optimal classifer does not define what to predict for $f(Rain, Hot)$
- If an instance with these feature values occurs in future data then we would need to predict randomly
- The same problem gets worse with more features

Tennis dataset with all 4 features

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Tennis dataset with all 4 features

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4					Yes
D5					Yes
D6					No
D7					Yes
D8					No
D9					Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D					
D					
D14	Rain	Mild	High	Strong	No

14 instances but $3 \times 3 \times 2 \times 2 = 36$ different possible feature value combinations:
$\{Overcast, Rain, Sunny\} \times$
$\{Cool, Hot, Mild\} \times$
$\{High, Normal\} \times \{Strong, Weak\}$

Therefore, we would need to predict randomly on most of the feature value combinations in the future

Give up exact feature combinations

- We need to give up on expecting to see the same feature combinations in train and test data
- What to do if we have a test instance with an unseen feature combination?
- We will see two solutions soon:
  - K-Nearest Neighbours
  - Naïve Bayes

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes

Question

What would you predict for $\mathbf{x} = (Sunny, Hot, Normal, Strong)$?

1 Nearest Neighbour (1NN)

- It is natural to hope that if two instances differ by only 1 feature then they are very likely to belong to the same class

1 Nearest Neighbour (1NN)

- It is natural to hope that if two instances differ by only 1 feature then they are very likely to belong to the same class
- 1 nearest neighbour algorithm (1NN):
  - For a given test instance $\mathbf{x}$
  - Find the training instance $\mathbf{x}_i$ that is nearest to $\mathbf{x}$
  - Predict the label $\mathbf{y}_i$ on that training instance

1 Nearest Neighbour (1NN)

- It is natural to hope that if two instances differ by only 1 feature then they are very likely to belong to the same class
- 1 nearest neighbour algorithm (1NN):
  - For a given test instance $\mathbf{x}$
  - Find the training instance $\mathbf{x}_i$ that is nearest to $\mathbf{x}$
  - Predict the label $\mathbf{y}_i$ on that training instance
- Need to define what we mean by "*nearest to*"
  - For categorical features – we could find the instance with the most shared feature values

K-Nearest Neighbours (KNN)

- **Pros**: One of the simplest learning methods, applies for both categorical and numerical data

K-Nearest Neighbours (KNN)

- **Pros**: One of the simplest learning methods, applies for both categorical and numerical data
- K - integer, usually an odd number $(1, 3, 5, \ldots)$

K-Nearest Neighbours (KNN)

- **Pros**: One of the simplest learning methods, applies for both categorical and numerical data
- K - integer, usually an odd number $(1, 3, 5, \ldots)$
- Learning the model=Memorise training data

K-Nearest Neighbours (KNN)

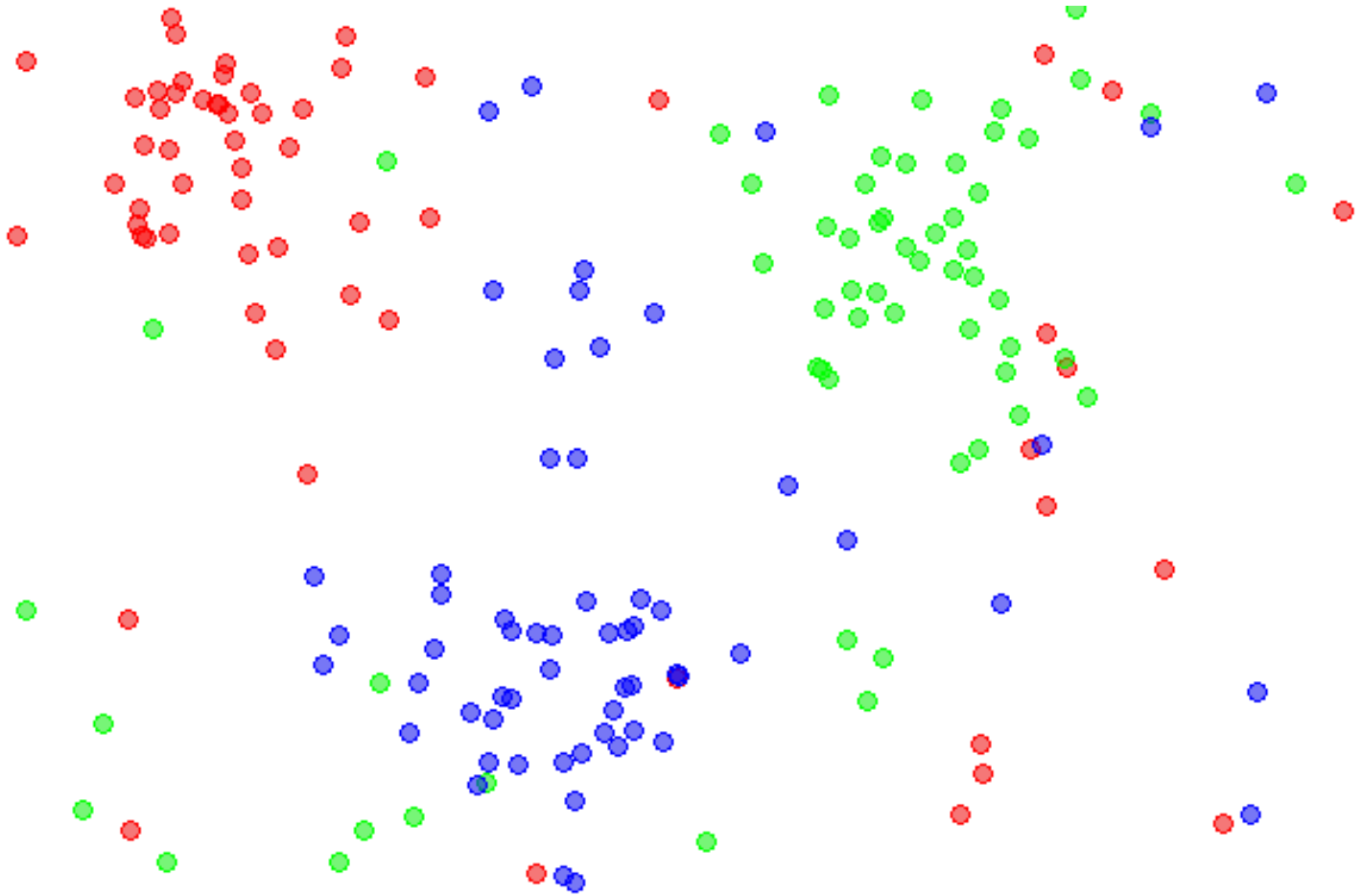
- **Pros**: One of the simplest learning methods, applies for both categorical and numerical data
- K - integer, usually an odd number $(1, 3, 5, \ldots)$
- Learning the model=Memorise training data
- Applying the model on test data:
  - For each test instance:
    - Find the K closest training instances
    - Predict most frequent label (in classification) or average label (in regression) among those K

K-Nearest Neighbours (KNN)

- **Pros**: One of the simplest learning methods, applies for both categorical and numerical data
- K - integer, usually an odd number $(1, 3, 5, \ldots)$
- Learning the model=Memorise training data
- Applying the model on test data:
  - For each test instance:
    - Find the K closest training instances
    - Predict most frequent label (in classification) or average label (in regression) among those K
- **Cons**: hard to choose a good distance measure and many distance calculations are required

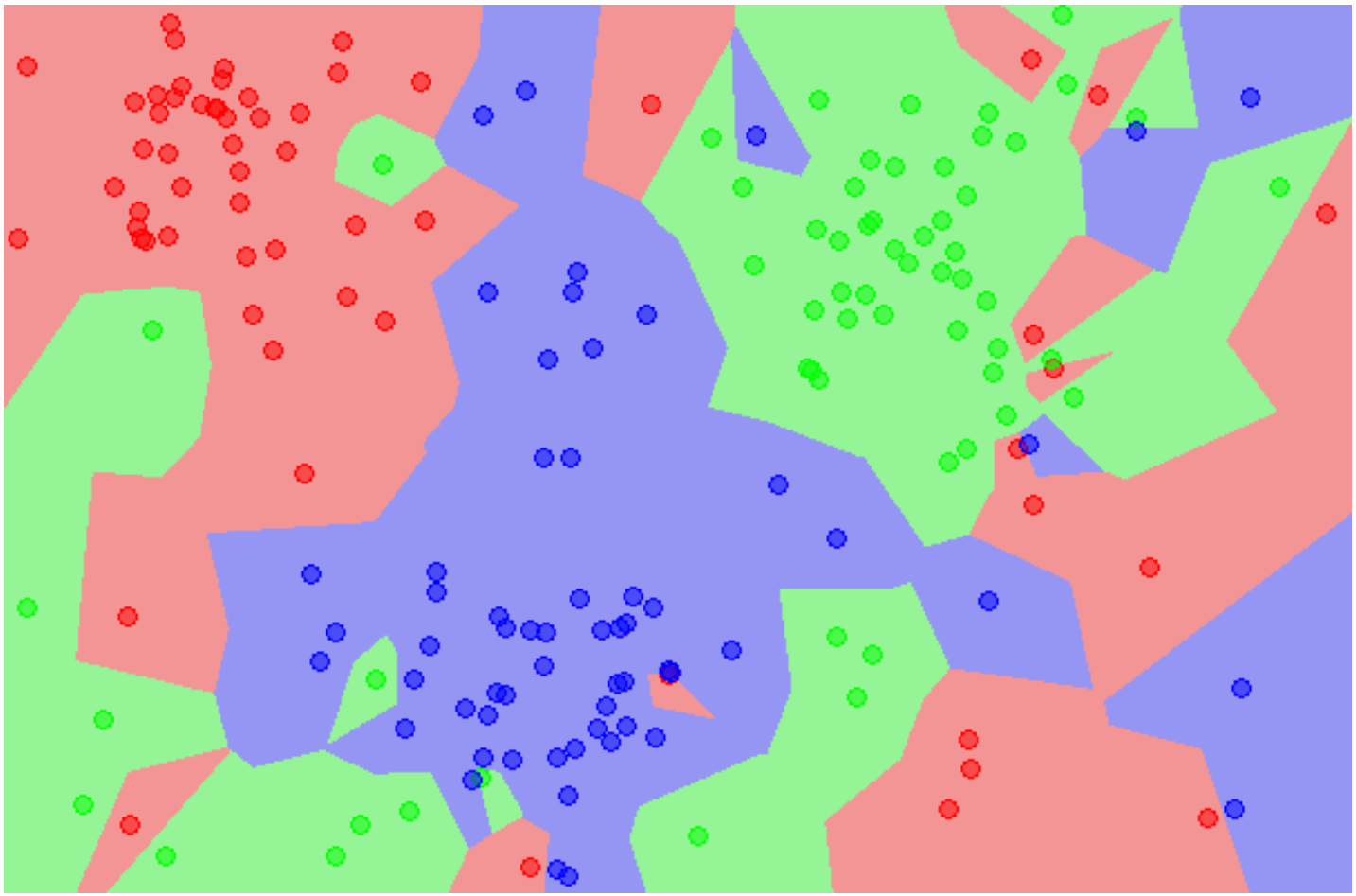
Example in 2D

Figure: Data points in 2D



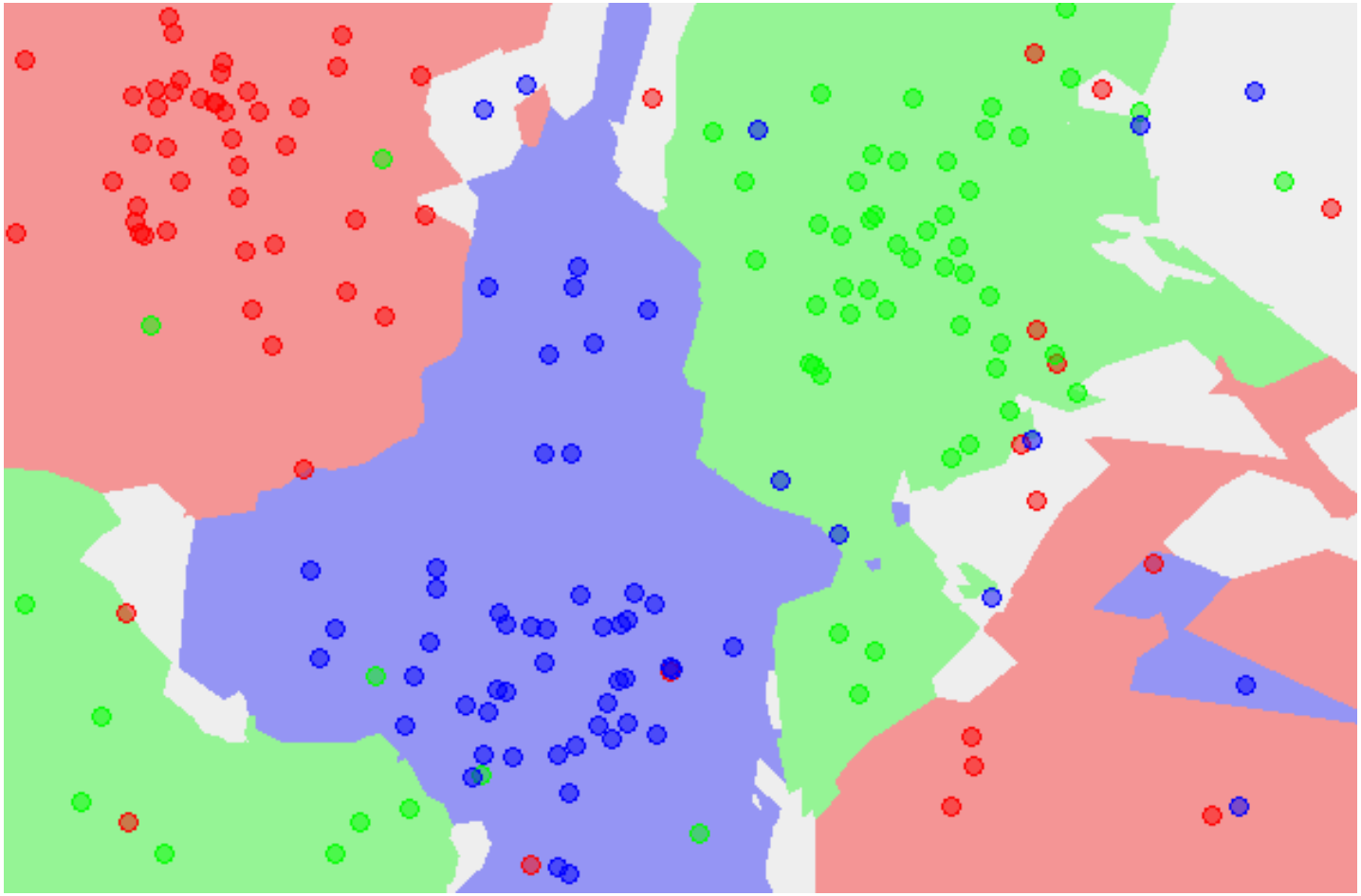
Example in 2D

Figure: Classification regions based on 1NN



Example in 2D

Figure: Classification regions based on 5NN



Well-known distance measures

- For numeric vectors $\mathbf{x} = (x_1, \ldots, x_m)$ and $\mathbf{x}' = (x'_1, \ldots, x'_m)$

Well-known distance measures

- For numeric vectors $\mathbf{x} = (x_1, \ldots, x_m)$ and $\mathbf{x}' = (x'_1, \ldots, x'_m)$
  - Euclidean distance: $d(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{i=1}^{m} (x_i - x'_i)^2}$

Well-known distance measures

- For numeric vectors $\mathbf{x} = (x_1, \ldots, x_m)$ and $\mathbf{x}' = (x'_1, \ldots, x'_m)$
  - Euclidean distance: $d(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{i=1}^{m} (x_i - x'_i)^2}$
  - Manhattan distance: $d(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{m} |x_i - x'_i|$

Well-known distance measures

- For numeric vectors $\mathbf{x} = (x_1, \ldots, x_m)$ and $\mathbf{x}' = (x'_1, \ldots, x'_m)$
  - Euclidean distance: $d(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{i=1}^{m} (x_i - x'_i)^2}$
  - Manhattan distance: $d(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{m} |x_i - x'_i|$
  - Correlation distance: $d(\mathbf{x}, \mathbf{x}') = 1 - \rho(\mathbf{x}, \mathbf{x}')$, where $\rho(\mathbf{x}, \mathbf{x}')$ is Pearson correlation coefficient

Well-known distance measures

- For numeric vectors $\mathbf{x} = (x_1, \ldots, x_m)$ and $\mathbf{x}' = (x'_1, \ldots, x'_m)$
  - Euclidean distance: $d(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{i=1}^{m} (x_i - x'_i)^2}$
  - Manhattan distance: $d(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{m} |x_i - x'_i|$
  - Correlation distance: $d(\mathbf{x}, \mathbf{x}') = 1 - \rho(\mathbf{x}, \mathbf{x}')$, where $\rho(\mathbf{x}, \mathbf{x}')$ is Pearson correlation coefficient
- For categorical vectors
  - Hamming distance: $d(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{m} \mathbb{1}_{\{x_i \neq x'_i\}}$, where $\mathbb{1}_{\{.\}}$ is the indicator function.

New Feature Combination

What to do if we have a test instance with an unseen feature combination?

- KNN provided one possible solution
- Let us look for a probabilistic solution

Revision

- Conditional Probability: $P(A|B) = \frac{P(A,B)}{P(B)}$

Revision

- Conditional Probability: $P(A|B) = \frac{P(A,B)}{P(B)}$
- Chain rule: $P(A,B) = P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$

Revision

- Conditional Probability: $P(A|B) = \frac{P(A,B)}{P(B)}$
- Chain rule: $P(A,B) = P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$
- Bayes' rule: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

Revision

- Conditional Probability: $P(A|B) = \frac{P(A,B)}{P(B)}$
- Chain rule: $P(A,B) = P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$
- Bayes' rule: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$
- If A, B are independent $(A \perp B)$, then $P(A,B) = P(A) \cdot P(B)$ and $P(A|B) = P(A)$, $P(B|A) = P(B)$

Revision

- Conditional Probability: $P(A|B) = \dfrac{P(A,B)}{P(B)}$
- Chain rule: $P(A,B) = P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$
- Bayes' rule: $P(A|B) = \dfrac{P(B|A)P(A)}{P(B)}$
- If A, B are independent $(A \perp B)$, then $P(A,B) = P(A) \cdot P(B)$ and $P(A|B) = P(A)$, $P(B|A) = P(B)$

Definition: Conditional independence

If A, B are conditionally independent given C (we will denote $A \perp B|C$), then

$$P(A,B|C) = P(A|C) \cdot P(B|C)$$

and

$$P(A|B,C) = P(A|C); \quad P(B|A,C) = P(B|C)$$

Naïve Bayes

- Assume that any pair of features are conditionally independent given the label:

$$X_i \perp X_j | Y, \quad i \neq j$$

- Then MAP decision rule gives us:

$$\hat{y} = f(\mathbf{x}) = \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(Y = y | \mathbf{X} = \mathbf{x}) =$$

Naïve Bayes

- Assume that any pair of features are conditionally independent given the label:

$$X_i \perp X_j | Y, \quad i \neq j$$

- Then MAP decision rule gives us:

$$\hat{y} = f(\mathbf{x}) = \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(Y = y | \mathbf{X} = \mathbf{x}) =$$

$$= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(\mathbf{X} = \mathbf{x} | Y = y) \cdot P(Y = y) / P(\mathbf{X} = \mathbf{x}) =$$

Naïve Bayes

- Assume that any pair of features are conditionally independent given the label:

$$X_i \perp X_j | Y, \quad i \neq j$$

- Then MAP decision rule gives us:

$$\begin{aligned}
\hat{y} = f(\mathbf{x}) &= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(Y = y | \mathbf{X} = \mathbf{x}) = \\
&= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(\mathbf{X} = \mathbf{x} | Y = y) \cdot P(Y = y) / P(\mathbf{X} = \mathbf{x}) = \\
&= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(\mathbf{X} = \mathbf{x} | Y = y) \cdot P(Y = y) =
\end{aligned}$$

Naïve Bayes

- Assume that any pair of features are conditionally independent given the label:

$$X_i \perp X_j | Y, \quad i \neq j$$

- Then MAP decision rule gives us:

$$\begin{aligned}
\hat{y} = f(\mathbf{x}) &= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(Y = y | \mathbf{X} = \mathbf{x}) = \\
&= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(\mathbf{X} = \mathbf{x} | Y = y) \cdot P(Y = y) / P(\mathbf{X} = \mathbf{x}) = \\
&= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(\mathbf{X} = \mathbf{x} | Y = y) \cdot P(Y = y) = \\
&= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(X_1 = x_1, \ldots, X_m = x_m | Y = y) \cdot P(Y = y) =
\end{aligned}$$

Naïve Bayes

- Assume that any pair of features are conditionally independent given the label:

$$X_i \perp X_j | Y, \quad i \neq j$$

- Then MAP decision rule gives us:

$$\begin{aligned}
\hat{y} = f(\mathbf{x}) &= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(Y = y | \mathbf{X} = \mathbf{x}) = \\
&= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(\mathbf{X} = \mathbf{x} | Y = y) \cdot P(Y = y) / P(\mathbf{X} = \mathbf{x}) = \\
&= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(\mathbf{X} = \mathbf{x} | Y = y) \cdot P(Y = y) = \\
&= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(X_1 = x_1, \ldots, X_m = x_m | Y = y) \cdot P(Y = y) = \\
&= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(X_1 = x_1 | Y = y) \ldots P(X_m = x_m | Y = y) \cdot P(Y = y) =
\end{aligned}$$

Naïve Bayes

- Assume that any pair of features are conditionally independent given the label:

$$X_i \perp X_j | Y, \quad i \neq j$$

- Then MAP decision rule gives us:

$$\begin{aligned}
\hat{y} = f(\mathbf{x}) &= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(Y = y | \mathbf{X} = \mathbf{x}) = \\
&= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(\mathbf{X} = \mathbf{x} | Y = y) \cdot P(Y = y) / P(\mathbf{X} = \mathbf{x}) = \\
&= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(\mathbf{X} = \mathbf{x} | Y = y) \cdot P(Y = y) = \\
&= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(X_1 = x_1, \ldots, X_m = x_m | Y = y) \cdot P(Y = y) = \\
&= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(X_1 = x_1 | Y = y) \ldots P(X_m = x_m | Y = y) \cdot P(Y = y) = \\
&= \underset{y \in \mathbb{Y}}{\operatorname{argmax}} P(Y = y) \prod_{i=1}^{m} P(X_i = x_i | Y = y)
\end{aligned}$$

Naïve Bayes

- Assume that any pair of features are conditionally independent given the label:

$$X_i \perp X_j | Y, \quad i \neq j$$

- Then MAP decision rule gives us:

$$\hat{y} = f(\mathbf{x}) = \operatorname{argmax} P(Y = y | \mathbf{X} = \mathbf{x}) =$$

Naïve Bayes

Predict the class according to the following rule:

$$\hat{y} = f(\mathbf{x}) = \operatorname*{argmax}_{y \in \mathbb{Y}} P(Y = y) \prod_{i=1}^{m} P(X_i = x_i | Y = y)$$

$$= \operatorname*{argmax}_{y \in \mathbb{Y}} P(X_1 = x_1 | Y = y) \dots P(X_m = x_m | Y = y) \cdot P(Y = y) =$$

$$= \operatorname*{argmax}_{y \in \mathbb{Y}} P(Y = y) \prod_{i=1}^{m} P(X_i = x_i | Y = y)$$

Example

What would Naïve Bayes predict for $x = (Sunny, Hot, Normal, Strong)$?

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example

- What would Naïve Bayes predict for $x = (Sunny, Hot, Normal, Strong)$?

$$P(Y = \oplus) \prod_{i=1}^{m} P(X_i = x_i | Y = \oplus) = \frac{9}{14} \cdot \frac{2}{9} \cdot \frac{2}{9} \cdot \frac{6}{9} \cdot \frac{3}{9} = \frac{4}{567} \approx 0.007$$

$$P(Y = \ominus) \prod_{i=1}^{m} P(X_i = x_i | Y = \ominus) = \frac{5}{14} \cdot \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{3}{5} = \frac{9}{875} \approx 0.01$$

- Naïve Bayes predicts negative because:

$$P(Y = \ominus) \prod_{i=1}^{m} P(X_i = x_i | Y = \ominus) > P(Y = \oplus) \prod_{i=1}^{m} P(X_i = x_i | Y = \oplus)$$

Laplace Smoothing

- When calculating probabilities in

$$P(Y = y) \prod_{i=1}^{m} P(X_i = x_i | Y = y)$$

any zero probability would make the total into zero also

Laplace Smoothing

- When calculating probabilities in

$$P(Y = y) \prod_{i=1}^{m} P(X_i = x_i | Y = y)$$

  any zero probability would make the total into zero also
- Usually, **Laplace smoothing** is applied when calculating these probabilities, so instead of dividing $\frac{a}{b}$, where a are favourable and b are all options, in Laplace smoothing one would use the ratio $\frac{a+1}{b+2}$

Example with Laplace smoothing

- What would Naïve Bayes predict for $x = (Sunny, Hot, Normal, Strong)$?

$$P(Y = \oplus) \prod_{i=1}^{m} P(X_i = x_i | Y = \oplus) = \frac{10}{16} \cdot \frac{3}{11} \cdot \frac{3}{11} \cdot \frac{7}{11} \cdot \frac{4}{11} = \frac{315}{29282} \approx 0.01$$

$$P(Y = \ominus) \prod_{i=1}^{m} P(X_i = x_i | Y = \ominus) = \frac{6}{16} \cdot \frac{4}{7} \cdot \frac{3}{7} \cdot \frac{2}{7} \cdot \frac{4}{7} = \frac{36}{2401} \approx 0.015$$

- Again, Naïve Bayes predicts negative because:

$$P(Y = \ominus) \prod_{i=1}^{m} P(X_i = x_i | Y = \ominus) > P(Y = \oplus) \prod_{i=1}^{m} P(X_i = x_i | Y = \oplus)$$

Naïve Bayes

- **Pro**: simple and takes into account the class prior
- **Con**: assumes conditional independence of features given the label

Naïve Bayes

- **Pro**: simple and takes into account the class prior
- **Con**: assumes conditional independence of features given the label
- Being "Naïve" can be a good thing:

Occam's razor (important guideline in machine learning)

Everything should be made as simple as possible, but not simpler.

What does "Bayesian" mean?

- It means that you have some prior belief about the probability of some event(s) before observing the data

What does "Bayesian" mean?

- It means that you have some prior belief about the probability of some event(s) before observing the data
- You update your belief according to the likelihood of new observations

What does "Bayesian" mean?

- It means that you have some prior belief about the probability of some event(s) before observing the data
- You update your belief according to the likelihood of new observations
- In our case:

What does "Bayesian" mean?

- It means that you have some prior belief about the probability of some event(s) before observing the data
- You update your belief according to the likelihood of new observations
- In our case:
  - Prior belief was $P(Y = y)$

What does "Bayesian" mean?

- It means that you have some prior belief about the probability of some event(s) before observing the data
- You update your belief according to the likelihood of new observations
- In our case:
  - Prior belief was $P(Y = y)$
  - We observed $\mathbf{X} = \mathbf{x}$

What does "Bayesian" mean?

- It means that you have some prior belief about the probability of some event(s) before observing the data
- You update your belief according to the likelihood of new observations
- In our case:
  - Prior belief was $P(Y = y)$
  - We observed $\mathbf{X} = \mathbf{x}$
  - Likelihood of observations: $P(\mathbf{X} = \mathbf{x}|Y = y)$

What does "Bayesian" mean?

- It means that you have some prior belief about the probability of some event(s) before observing the data
- You update your belief according to the likelihood of new observations
- In our case:
  - Prior belief was $P(Y = y)$
  - We observed $\mathbf{X} = \mathbf{x}$
  - Likelihood of observations: $P(\mathbf{X} = \mathbf{x}|Y = y)$
  - Posterior (updated) belief:

$$P(Y = y|\mathbf{X} = \mathbf{x}) = \frac{P(Y = y) \cdot P(\mathbf{X} = \mathbf{x}|Y = y)}{P(\mathbf{X} = \mathbf{x})}$$

What have we learned today?

Classifiers:

- ✓ K-Nearest Neighbours
- ✓ Naive Bayes