Classification II: Decision Trees and SVMs

Digging into Data: Jordan Boyd-Graber

February 25, 2013

COLLEGE OF INFORMATION STUDIES

Slides adapted from Tom Mitchell, Eric Xing, and Lauren Hannah

Roadmap

- Classification: machines labeling data for us
- Last time: naïve Bayes and logistic regression
- This time:
  - Decision Trees
    - Simple, nonlinear, interpretable
  - SVMs
    - (another) example of linear classifier
    - State-of-the-art classification
  - Examples in Rattle (Logistic, SVM, Trees)
  - **Discussion:** Which classifier should I use for my problem?

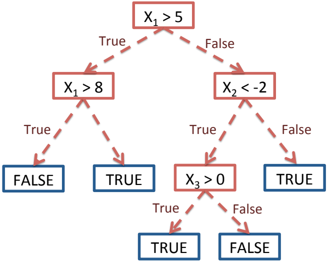
Outline

1. Decision Trees
2. Learning Decision Trees
3. Overfitting
4. Vector space classification
5. Linear Classifiers
6. Which hyperplane is best for me?
7. Support Vector Machines
8. Trying Out Classifiers in Rattle
9. Recap

Trees

Suppose that we want to construct a set of rules to represent the data

- can represent data as a series of if-then statements
- here, “if” splits inputs into two categories
- “then” assigns value
- when “if” statements are nested, structure is called a tree

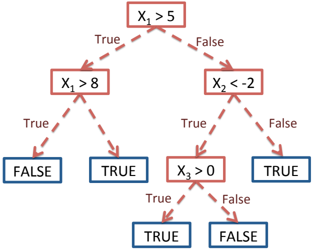


Trees

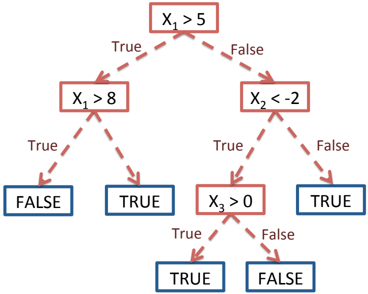
Ex: data (X_1, X_2, X_3, Y) with X_1, X_2, X_3 are real, Y Boolean

First, see if $X_1 > 5$:

- if TRUE, see if $X_1 > 8$
  - if TRUE, return FALSE
  - if FALSE, return TRUE
- if FALSE, see if $X_2 < -2$
  - if TRUE, see if $X_3 > 0$
    - if TRUE, return TRUE
    - if FALSE, return FALSE
  - if FALSE, return TRUE



Trees



Example 1: $(X_1, X_2, X_3) = (1, 1, 1)$

Example 2: $(X_1, X_2, X_3) = (10, -3, 0)$

Trees

Terminology:

- branches: one side of a split
- leaves: terminal nodes that return values

Why trees?

- trees can be used for regression or classification
  - regression: returned value is a real number
  - classification: returned value is a class
- unlike linear regression, SVMs, naive Bayes, etc, trees fit *local models*
  - in large spaces, global models may be hard to fit
  - results may be hard to interpret
- fast, interpretable predictions

Example: Predicting Electoral Results

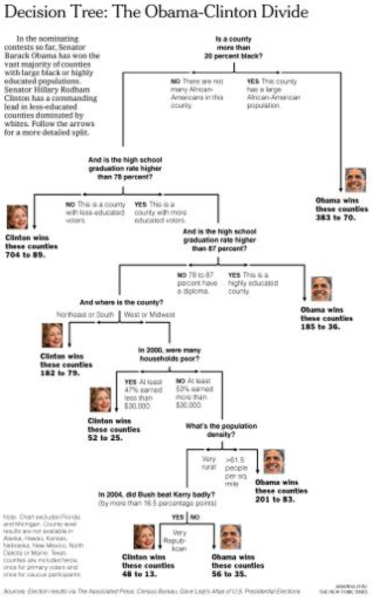
2008 Democratic primary:

- Hillary Clinton
- Barack Obama

Given historical data, how will a count vote?

- can extrapolate to state level data
- might give regions to focus on increasing voter turnout
- would like to know how variables interact

Example: Predicting Electoral Results



Decision Trees

Decision tree representation:

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

How would we represent:

- AND , OR , XOR
- $(A \text{ AND } B) \text{ OR } (C \text{ AND } \neg D \text{ AND } E)$
- M of N

When to Consider Decision Trees

- Instances describable by attribute-value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

Examples:

- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

Outline

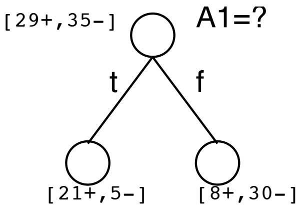
1. Decision Trees
2. **Learning Decision Trees**
3. Overfitting
4. Vector space classification
5. Linear Classifiers
6. Which hyperplane is best for me?
7. Support Vector Machines
8. Trying Out Classifiers in Rattle
9. Recap

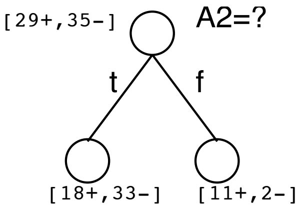
Top-Down Induction of Decision Trees

Main loop:

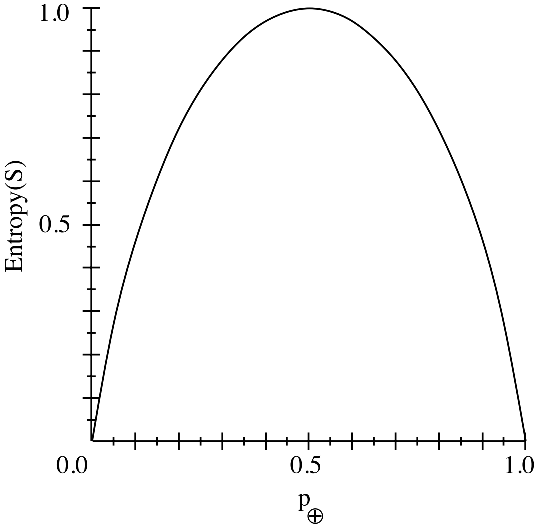
1. $A \leftarrow$ the “best” decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A, create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Which attribute is best?





Entropy



- *S* is a sample of training examples

Entropy

$Entropy(S)$ = expected number of bits needed to encode class ($\oplus$ or $\ominus$) of randomly drawn member of S (under the optimal, shortest-length code)

Why?

Information theory: optimal length code assigns $-\log_2 p$ bits to message having probability p.

So, expected number of bits to encode $\oplus$ or $\ominus$ of random member of S:

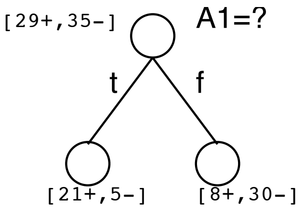
$$p_\oplus(-\log_2 p_\oplus) + p_\ominus(-\log_2 p_\ominus)$$

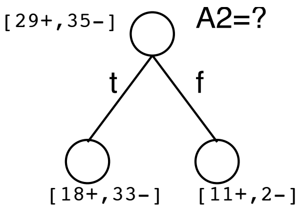
$$Entropy(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$

Information Gain

$Gain(S,A)$ = expected reduction in entropy due to sorting on A

$$Gain(S,A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$



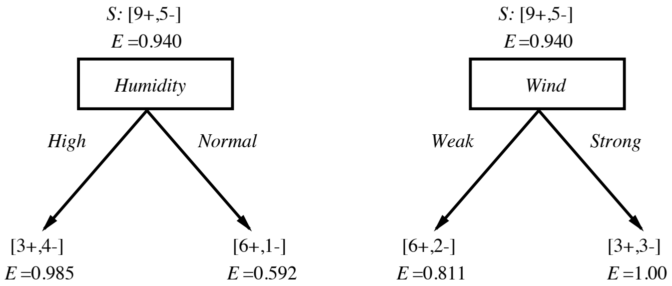


Training Examples

Day	Outlook	Temperature	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

Selecting the Next Attribute

Which attribute is the best classifier?



Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

ID3 Algorithm

- Start at root, look for best attribute
- Repeat for subtrees at each attribute outcome
- Stop when information gain is below a threshold
- Bias: prefers shorter trees (Occam's Razor)
  - $\longrightarrow$ a short hyp that fits data unlikely to be coincidence
  - $\longrightarrow$ a long hyp that fits data might be coincidence
  - Prevents overfitting

Outline

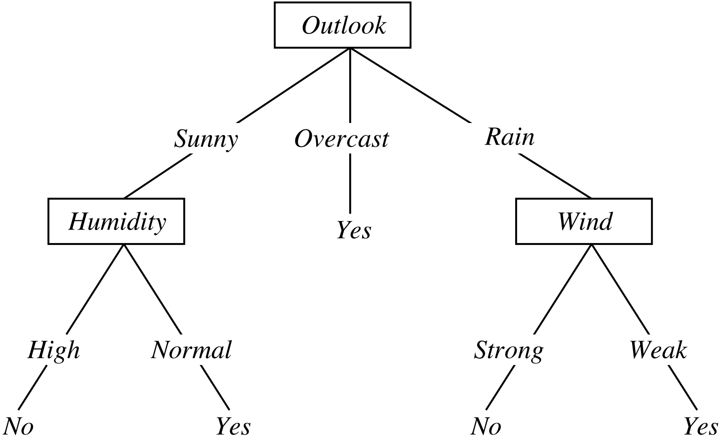
1. Decision Trees
2. Learning Decision Trees
3. **Overfitting**
4. Vector space classification
5. Linear Classifiers
6. Which hyperplane is best for me?
7. Support Vector Machines
8. Trying Out Classifiers in Rattle
9. Recap

Overfitting in Decision Trees

Consider adding noisy training example #15:

$$Sunny, Hot, Normal, Strong, PlayTennis = No$$

What effect on earlier tree?



Overfitting

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution $\mathscr{D}$ of data: $error_{\mathscr{D}}(h)$

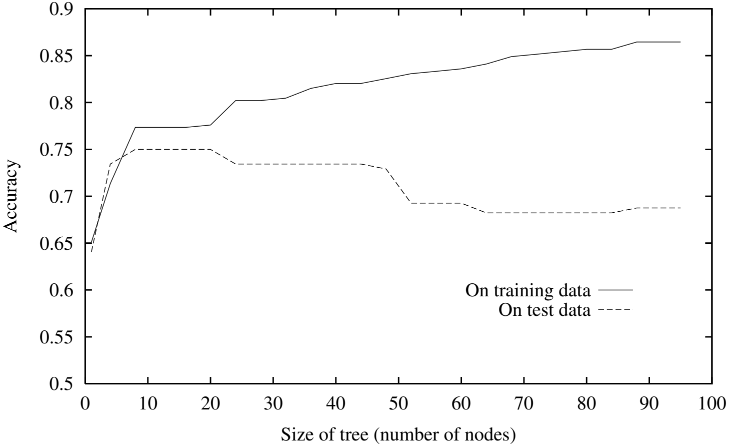
Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathscr{D}}(h) > error_{\mathscr{D}}(h')$$

Overfitting in Decision Tree Learning



Avoiding Overfitting

How can we avoid overfitting?

- stop growing when data split not statistically significant
- grow full tree, then post-prune

How to select "best" tree:

- Measure performance over training data
- Measure performance over separate validation data set
- MDL: minimize $\mathit{size}(\mathit{tree}) + \mathit{size}(\mathit{misclassifications}(\mathit{tree}))$

Outline

1 Decision Trees

2 Learning Decision Trees

3 Overfitting

4 **Vector space classification**

5 Linear Classifiers

6 Which hyperplane is best for me?

7 Support Vector Machines

8 Trying Out Classifiers in Rattle

9 Recap

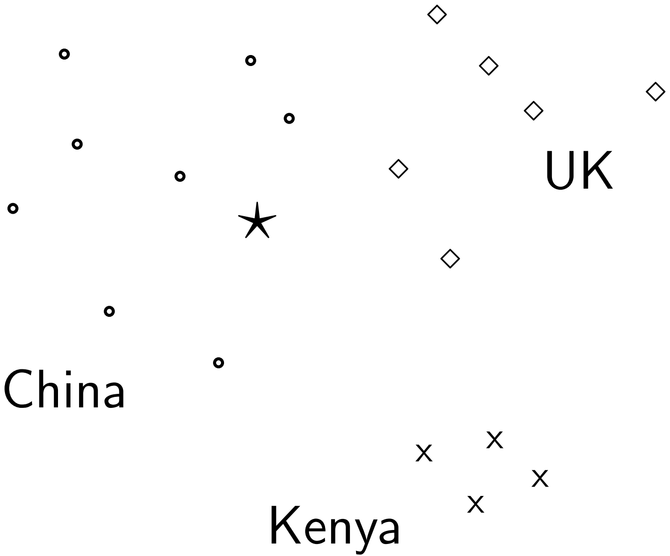
Recall vector space representation

- Each document is a vector, one component for each term.
- Terms are axes.
- High dimensionality: 10,000s of dimensions and more
- Normalize vectors (documents) to unit length
- How can we do classification in this space?

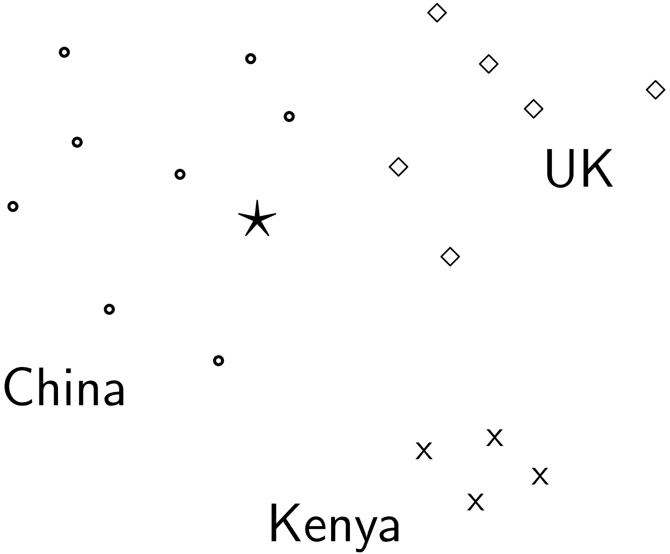
Vector space classification

- As before, the training set is a set of documents, each labeled with its class.
- In vector space classification, this set corresponds to a labeled set of points or vectors in the vector space.
- Premise 1: Documents in the same class form a **contiguous region**.
- Premise 2: Documents from different classes **don't overlap**.
- We define lines, surfaces, hypersurfaces to divide regions.

Classes in the vector space

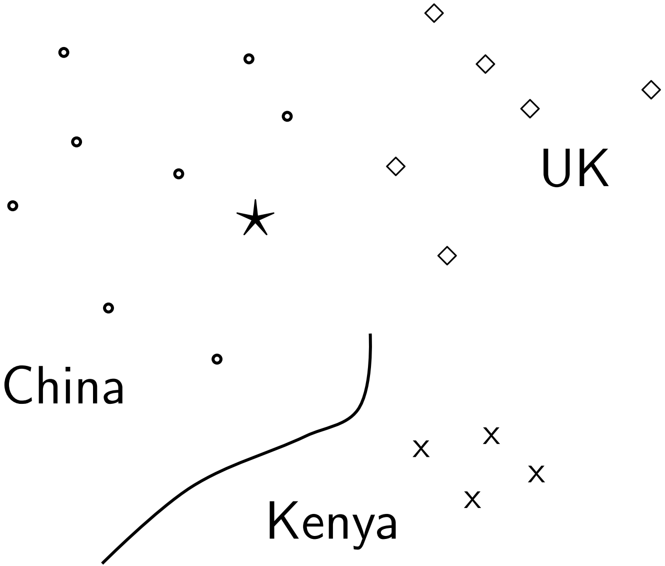


Classes in the vector space



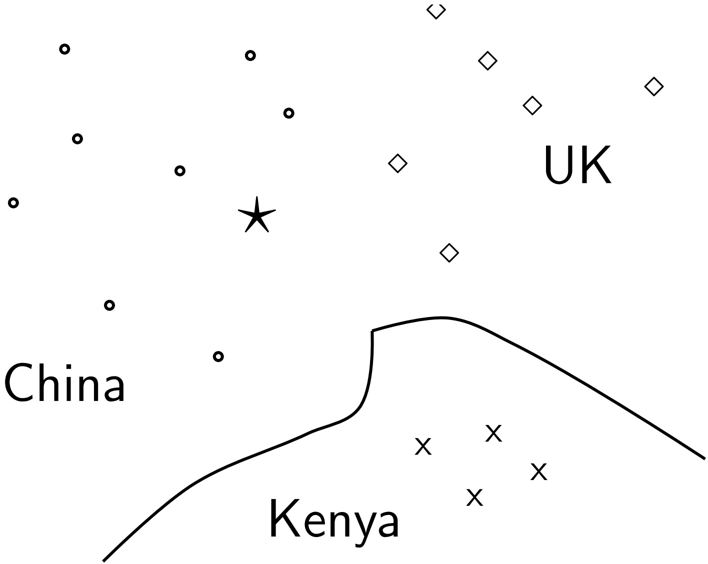
Should the document ⋆ be assigned to `China`, `UK` or `Kenya`?

Classes in the vector space



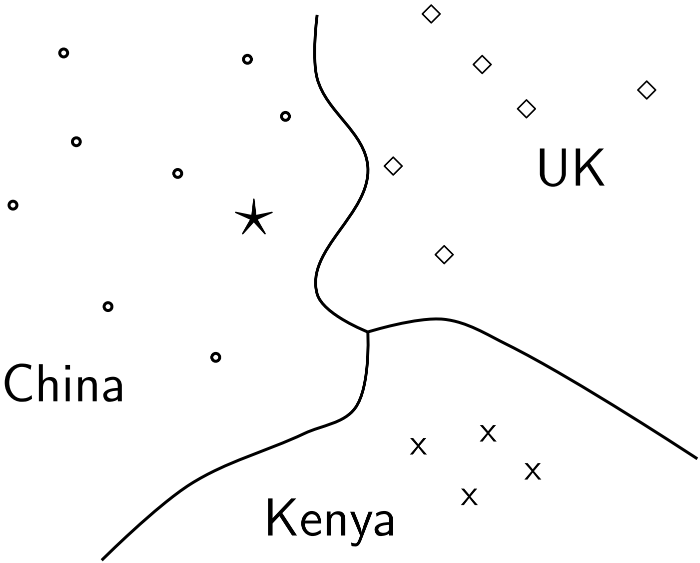
Find separators between the classes

Classes in the vector space



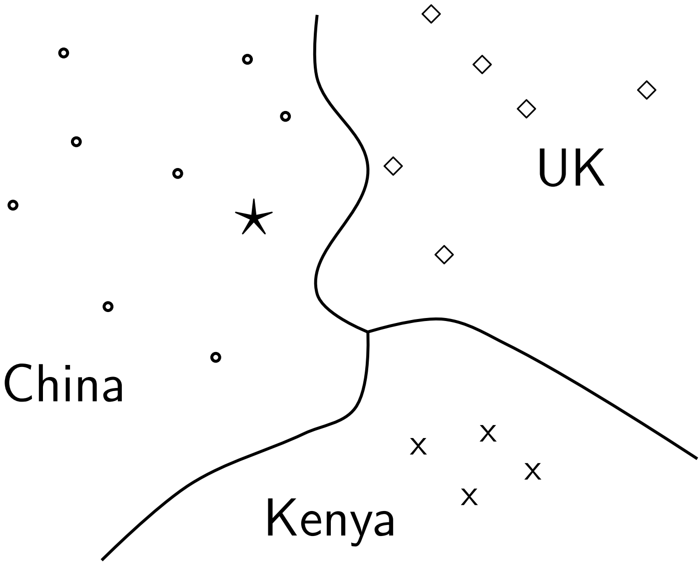
Find separators between the classes

Classes in the vector space



Based on these separators: $\star$ should be assigned to `China`

Classes in the vector space



How do we find separators that do a good job at classifying new documents like $\star$?

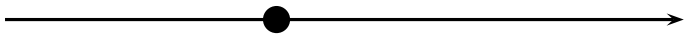
Outline

1. Decision Trees
2. Learning Decision Trees
3. Overfitting
4. Vector space classification
5. **Linear Classifiers**
6. Which hyperplane is best for me?
7. Support Vector Machines
8. Trying Out Classifiers in Rattle
9. Recap

Linear classifiers

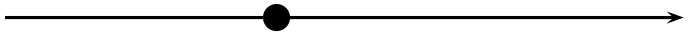
- Definition:
  - A linear classifier computes a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
  - Classification decision: $\sum_i w_i x_i > \theta$?
  - ... where θ (the threshold) is a parameter.
- (First, we only consider binary classifiers.)
- Geometrically, this corresponds to a line (2D), a plane (3D) or a hyperplane (higher dimensionalities).
- We call this the **separator** or **decision boundary.**
- We find the separator based on training set.
- Methods for finding separator: Perceptron, Rocchio, Naive Bayes, linear SVM
- Assumption: The classes are **linearly separable.**

A linear classifier in 1D



- A linear classifier in 1D is a point x described by the equation $w_1 d_1 = \theta$

A linear classifier in 1D

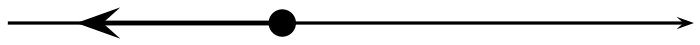


- A linear classifier in 1D is a point x described by the equation $w_1 d_1 = \theta$
- $x = \theta / w_1$

A linear classifier in 1D

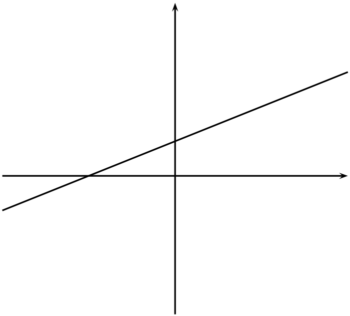
- A linear classifier in 1D is a point x described by the equation $w_1 d_1 = \theta$
- $x = \theta / w_1$
- Points (d_1) with $w_1 d_1 \geq \theta$ are in the class c.

A linear classifier in 1D



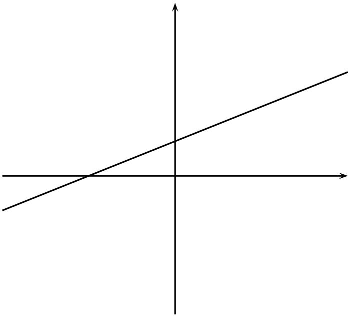
- A linear classifier in 1D is a point x described by the equation $w_1 d_1 = \theta$
- $x = \theta / w_1$
- Points (d_1) with $w_1 d_1 \geq \theta$ are in the class c.
- Points (d_1) with $w_1 d_1 < \theta$ are in the complement class $\overline{c}$.

A linear classifier in 2D



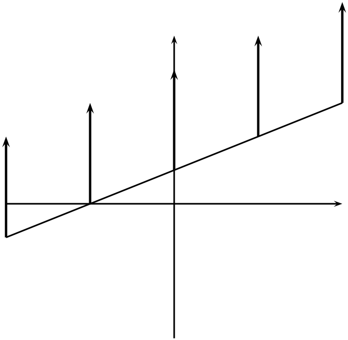
- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$

A linear classifier in 2D



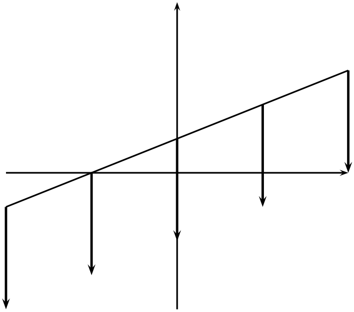
- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$
- Example for a 2D linear classifier

A linear classifier in 2D



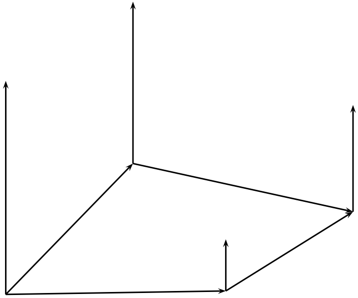
- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$
- Example for a 2D linear classifier
- Points $(d_1 \ d_2)$ with $w_1 d_1 + w_2 d_2 \geq \theta$ are in the class c.

A linear classifier in 2D



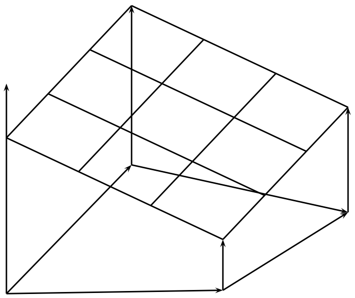
- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$
- Example for a 2D linear classifier
- Points $(d_1 \; d_2)$ with $w_1 d_1 + w_2 d_2 \geq \theta$ are in the class c.
- Points $(d_1 \; d_2)$ with $w_1 d_1 + w_2 d_2 < \theta$ are in the complement class $\overline{c}$.

A linear classifier in 3D



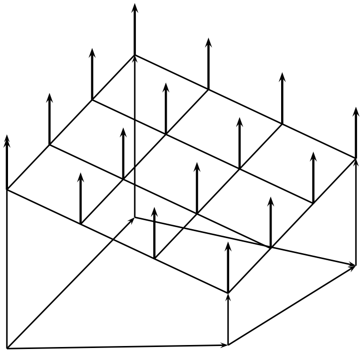
- A linear classifier in 3D is a plane described by the equation $w_1 d_1 + w_2 d_2 + w_3 d_3 = \theta$

A linear classifier in 3D



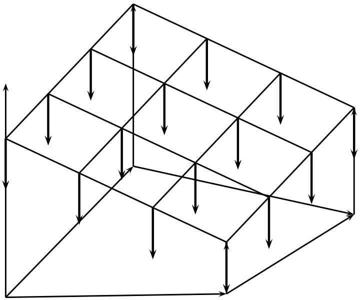
- A linear classifier in 3D is a plane described by the equation $w_1 d_1 + w_2 d_2 + w_3 d_3 = \theta$
- Example for a 3D linear classifier

A linear classifier in 3D



- A linear classifier in 3D is a plane described by the equation $w_1 d_1 + w_2 d_2 + w_3 d_3 = \theta$
- Example for a 3D linear classifier
- Points $(d_1 \; d_2 \; d_3)$ with $w_1 d_1 + w_2 d_2 + w_3 d_3 \geq \theta$ are in the class c.

A linear classifier in 3D



- A linear classifier in 3D is a plane described by the equation $w_1 d_1 + w_2 d_2 + w_3 d_3 = \theta$
- Example for a 3D linear classifier
- Points $(d_1 \; d_2 \; d_3)$ with $w_1 d_1 + w_2 d_2 + w_3 d_3 \geq \theta$ are in the class c.
- Points $(d_1 \; d_2 \; d_3)$ with $w_1 d_1 + w_2 d_2 + w_3 d_3 < \theta$ are in the complement class $\overline{c}$.

Naive Bayes as a linear classifier

Multinomial Naive Bayes is a linear classifier (in log space) defined by:

$$\sum_{i=1}^{M} w_i d_i = \theta$$

where $w_i = \log[\hat{P}(t_i|c)/\hat{P}(t_i|\bar{c})]$, $d_i =$ number of occurrences of t_i in d, and $\theta = -\log[\hat{P}(c)/\hat{P}(\bar{c})]$. Here, the index i, $1 \leq i \leq M$, refers to terms of the vocabulary

Naive Bayes as a linear classifier

Multinomial Naive Bayes is a linear classifier (in log space) defined by:

$$\sum_{i=1}^{M} w_i d_i = \theta$$

where $w_i = \log[\hat{P}(t_i|c)/\hat{P}(t_i|\bar{c})]$, $d_i =$ number of occurrences of t_i in d, and $\theta = -\log[\hat{P}(c)/\hat{P}(\bar{c})]$. Here, the index i, $1 \leq i \leq M$, refers to terms of the vocabulary

Takeway

Naïve Bayes, logistic regression and SVM (which we'll get to in a second) are all linear methods. They choose their hyperplanes based on different objectives: joint likelihood (NB), conditional likelihood (LR), and the margin (SVM).

Reminder: Example of a linear classifier

t_i	w_i	d_{1i}	d_{2i}	t_i	w_i	d_{1i}	d_{2i}
prime	0.70	0	1	dlrs	-0.71	1	1
rate	0.67	1	0	world	-0.35	1	0
interest	0.63	0	0	sees	-0.33	0	0
rates	0.60	0	0	year	-0.25	0	0
discount	0.46	1	0	group	-0.24	0	0
bundesbank	0.43	0	0	dlr	-0.24	0	0

- This is for the class `interest` in Reuters-21578.
- For simplicity: assume a simple 0/1 vector representation
- d_1: “rate discount dlrs world”
- d_2: “prime dlrs”
- $\theta = 0$
- Exercise: Which class is d_1 assigned to? Which class is d_2 assigned to?

Reminder: Example of a linear classifier

t_i	w_i	d_{1i}	d_{2i}	t_i	w_i	d_{1i}	d_{2i}
prime	0.70	0	1	dlrs	-0.71	1	1
rate	0.67	1	0	world	-0.35	1	0
interest	0.63	0	0	sees	-0.33	0	0
rates	0.60	0	0	year	-0.25	0	0
discount	0.46	1	0	group	-0.24	0	0
bundesbank	0.43	0	0	dlr	-0.24	0	0

- This is for the class `interest` in Reuters-21578.
- For simplicity: assume a simple 0/1 vector representation
- d_1: "rate discount dlrs world"
- d_2: "prime dlrs"
- $\theta = 0$
- Exercise: Which class is d_1 assigned to? Which class is d_2 assigned to?
- We assign document $\vec{d}_1$ "rate discount dlrs world" to `interest` since $\vec{w}^T\vec{d}_1 = 0.67 \cdot 1 + 0.46 \cdot 1 + (-0.71) \cdot 1 + (-0.35) \cdot 1 = 0.07 > 0 = \theta$.

Reminder: Example of a linear classifier

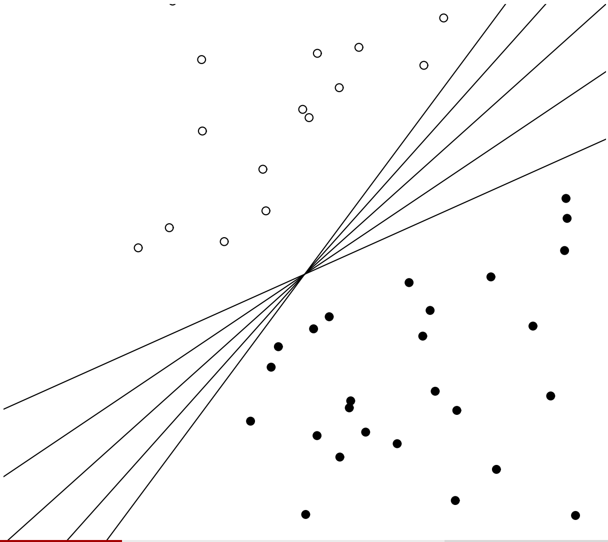
t_i	w_i	d_{1i}	d_{2i}	t_i	w_i	d_{1i}	d_{2i}
prime	0.70	0	1	dlrs	-0.71	1	1
rate	0.67	1	0	world	-0.35	1	0
interest	0.63	0	0	sees	-0.33	0	0
rates	0.60	0	0	year	-0.25	0	0
discount	0.46	1	0	group	-0.24	0	0
bundesbank	0.43	0	0	dlr	-0.24	0	0

- This is for the class `interest` in Reuters-21578.
- For simplicity: assume a simple 0/1 vector representation
- d_1: "rate discount dlrs world"
- d_2: "prime dlrs"
- $\theta = 0$
- Exercise: Which class is d_1 assigned to? Which class is d_2 assigned to?
- We assign document $\vec{d}_1$ "rate discount dlrs world" to `interest` since $\vec{w}^T\vec{d}_1 = 0.67 \cdot 1 + 0.46 \cdot 1 + (-0.71) \cdot 1 + (-0.35) \cdot 1 = 0.07 > 0 = \theta$.
- We assign $\vec{d}_2$ "prime dlrs" to the complement class (not in `interest`) since $\vec{w}^T\vec{d}_2 = -0.01 \leq \theta$.

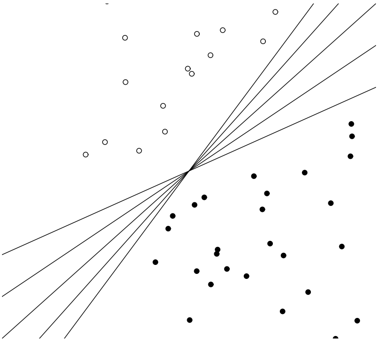
Outline

1. Decision Trees
2. Learning Decision Trees
3. Overfitting
4. Vector space classification
5. Linear Classifiers
6. **Which hyperplane is best for me?**
7. Support Vector Machines
8. Trying Out Classifiers in Rattle
9. Recap

Which hyperplane?



Which hyperplane?



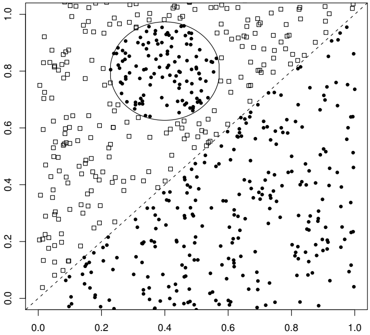
Which hyperplane?

- For linearly separable training sets: there are **infinitely** many separating hyperplanes.
- They all separate the training set perfectly ...
- ... but they behave differently on test data.
- Error rates on new data are low for some, high for others.
- How do we find a low-error separator?
- Naive Bayes: ok; linear SVM: good

Linear classifiers: Discussion

- Many common text classifiers are linear classifiers: Naive Bayes, logistic regression, linear support vector machines etc.
- Each method has a different way of selecting the separating hyperplane
  - Huge differences in performance on test documents
- Can we get better performance with more powerful nonlinear classifiers?
- Not in general: A given amount of training data may suffice for estimating a linear boundary, but not for estimating a more complex nonlinear boundary.

A nonlinear problem



- Linear classifier like Naive Bayes does badly on this task.
- kNN will do well (assuming enough training data)

Outline

1. Decision Trees
2. Learning Decision Trees
3. Overfitting
4. Vector space classification
5. Linear Classifiers
6. Which hyperplane is best for me?
7. **Support Vector Machines**
8. Trying Out Classifiers in Rattle
9. Recap

Support vector machines

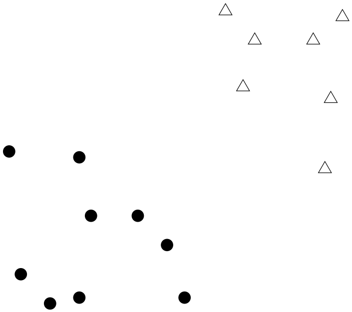
- Machine-learning research in the last two decades has improved classifier effectiveness.
- New generation of state-of-the-art classifiers: support vector machines (SVMs), boosted decision trees, regularized logistic regression, neural networks, and random forests
- Applications to IR problems, particularly text classification

SVMs: A kind of large-margin classifier

Vector space based machine-learning method aiming to find a decision boundary between two classes that is maximally far from any point in the training data (possibly discounting some points as outliers or noise)

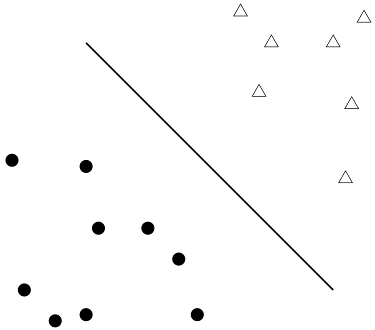
Support Vector Machines

- 2-class training data



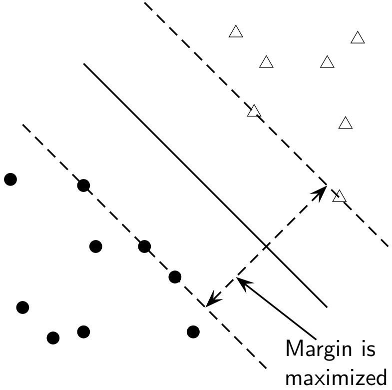
Support Vector Machines

- 2-class training data
- decision boundary → **linear separator**



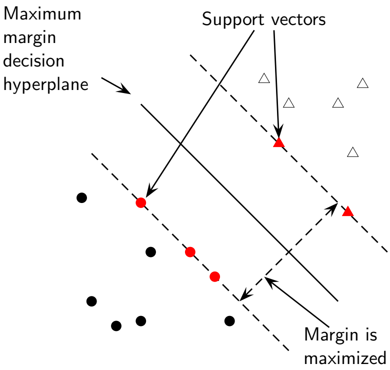
Support Vector Machines

- 2-class training data
- decision boundary ⟶ **linear separator**
- criterion: being maximally far away from any data point ⟶ determines classifier **margin**



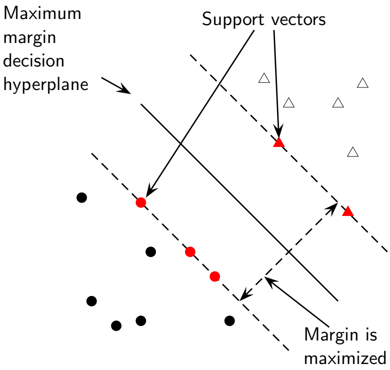
Support Vector Machines

- 2-class training data
- decision boundary → **linear separator**
- criterion: being maximally far away from any data point → determines classifier **margin**
- linear separator position defined by **support vectors**



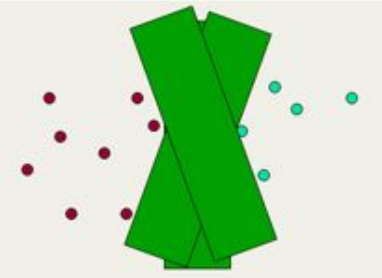
Why maximize the margin?

- Points near decision surface → uncertain classification decisions (50% either way).
- A classifier with a large margin makes no low certainty classification decisions.
- Gives classification safety margin w.r.t slight errors in measurement or documents variation



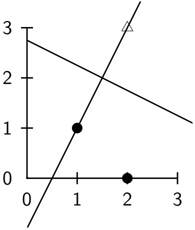
Why maximize the margin?

- SVM classifier: large margin around decision boundary
- compare to decision hyperplane: place fat separator between classes
  - unique solution
- decreased memory capacity
- increased ability to correctly generalize to test data



Walkthrough example: building an SVM over the data set shown in the figure

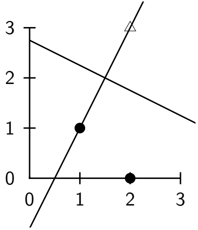
Working geometrically:



Walkthrough example: building an SVM over the data set shown in the figure

Working geometrically:

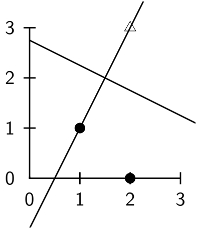
- The maximum margin weight vector will be parallel to the shortest line connecting points of the two classes, that is, the line between $(1,1)$ and $(2,3)$, giving a weight vector of $(1,2)$.



Walkthrough example: building an SVM over the data set shown in the figure

Working geometrically:

- The maximum margin weight vector will be parallel to the shortest line connecting points of the two classes, that is, the line between $(1,1)$ and $(2,3)$, giving a weight vector of $(1,2)$.
- The optimal decision surface is orthogonal to that line and intersects it at the halfway point. Therefore, it passes through $(1.5,2)$.

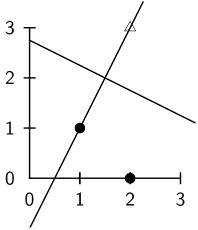


Walkthrough example: building an SVM over the data set shown in the figure

Working geometrically:

- The maximum margin weight vector will be parallel to the shortest line connecting points of the two classes, that is, the line between $(1,1)$ and $(2,3)$, giving a weight vector of $(1,2)$.
- The optimal decision surface is orthogonal to that line and intersects it at the halfway point. Therefore, it passes through $(1.5,2)$.
- The SVM decision boundary is:

$$0 = \frac{1}{2}x + y - \frac{11}{4} \Leftrightarrow 0 = \frac{2}{5}x + \frac{4}{5}y - \frac{11}{5}$$



SVM extensions

- Slack variables: not perfect line
- Kernels: different geometries
- Loss functions: Different penalties for getting the answer wrong

Outline

1 Decision Trees

2 Learning Decision Trees

3 Overfitting

4 Vector space classification

5 Linear Classifiers

6 Which hyperplane is best for me?

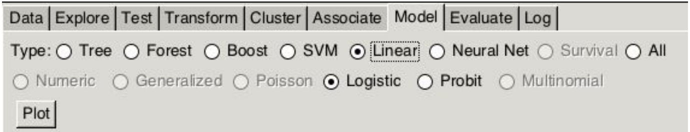
7 Support Vector Machines

8 **Trying Out Classifiers in Rattle**

9 Recap

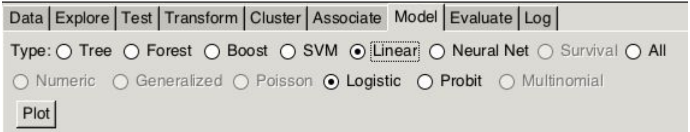
Selecting a model

- Go to “model” tab and select one of the models
- Make sure the model makes sense
- For logistic regression, select “linear” and “logistic”

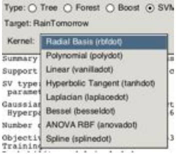


Selecting a model

- Go to “model” tab and select one of the models
- Make sure the model makes sense
- For logistic regression, select “linear” and “logistic”

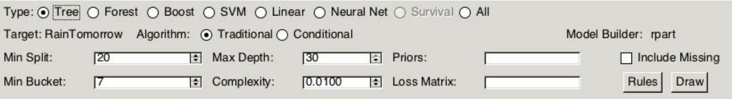


- For SVM, you also need to select a kernel (try linear first, then “Gaussian” which will be much slower)



- Output varies by model
  - SVM is least informative (hard to summarize)
  - Note you can click **draw** to see decision trees

Decision Trees Have Many Options . . .



- **Prior**: The prior observation probabilities (in case your training data are skewed)
- **Min Split**: How many observations can be in an expanded leaf (pre-test)
- **Min Bucket**: How many observations can be in any resulting leaf (post-test)
- **Max Depth**: How many levels the tree has
- **Complexity**: How many “if” statements the tree has

Defaults are reasonable; tweak if you are having complexity issues.

How'd we do?

- Fit the model by clicking on the "execute" button

- Click on the evaluate tab, have your boxes checked for the models you want to compare
- For the weather dataset, SVM does best (.14)
- We'll learn about the other metrics next week!

Outline

1. Decision Trees
2. Learning Decision Trees
3. Overfitting
4. Vector space classification
5. Linear Classifiers
6. Which hyperplane is best for me?
7. Support Vector Machines
8. Trying Out Classifiers in Rattle
9. **Recap**

Text classification

- Many commercial applications
- There are many applications of text classification for corporate Intranets, government departments, and Internet publishers.
- Often greater performance gains from exploiting domain-specific text features than from changing from one machine learning method to another.
- Understanding the data is one of the keys to successful categorization, yet this is an area in which many categorization tool vendors are weak.

Choosing what kind of classifier to use

When building a text classifier, first question: **how much training data is there currently available?**

- None?
- Very little?
- A fair amount?
- A huge amount

Choosing what kind of classifier to use

When building a text classifier, first question: **how much training data is there currently available?**

- None? **Hand write rules or use active learning**
- Very little?
- A fair amount?
- A huge amount

Choosing what kind of classifier to use

When building a text classifier, first question: **how much training data is there currently available?**

- None? **Hand write rules or use active learning**
- Very little? **Naïve Bayes**
- A fair amount?
- A huge amount

Choosing what kind of classifier to use

When building a text classifier, first question: **how much training data is there currently available?**

- None? **Hand write rules or use active learning**
- Very little? **Naïve Bayes**
- A fair amount? **SVM**
- A huge amount

Choosing what kind of classifier to use

When building a text classifier, first question: **how much training data is there currently available?**

- None? **Hand write rules or use active learning**
- Very little? **Naïve Bayes**
- A fair amount? **SVM**
- A huge amount **Doesn't matter, use whatever works**

Recap

- Is there a learning method that is optimal for all text classification problems?
- No, because there is a tradeoff between bias and variance.
- Factors to take into account:
  - How much training data is available?
  - How simple/complex is the problem? (linear vs. nonlinear decision boundary)
  - How noisy is the problem?
  - How stable is the problem over time?
    - For an unstable problem, it's better to use a simple and robust classifier.
    - You'll be investigating the role of features and classifiers in HW2!