6. Supervised Techniques III

DS-GA 1015, Text as Data
Arthur Spirling

March 16, 2021

Where Are We?

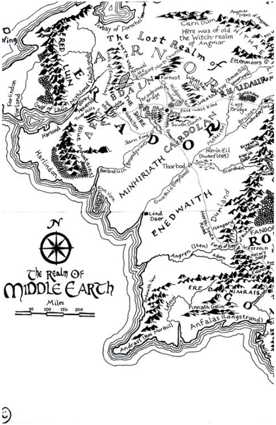
Where Are We?

The Lost Realm of
ARNOR
Carn Dûm
Here was of old the Witch-realm of Angmar
Ettenmoors
Fornost
Weather Hills
RHUDAUR
Last Bridge
East-West Road
Bree
South Downs
CARDOLAN
ERIADOR
THE SHIRE
Hobbiton
Sarn Ford
Harlond
Forlond
Forlindon
Gulf of Lhûn
Harlindon
MINHIRIATH
Tharbad
Nîn-in-Eilph (Swanfleet)
ENEDWAITH
Lond Daer
Dunland
Isengard
FANGORN
Helm's Deep
Edoras
Angren (Isen)
Adorn
Anfalas (Langstrand)
Pinnath Gelin
Andrast (Ras Morthil)
The Realm Of
MIDDLE EARTH
Miles
50 100 150 200

Where Are We?

We've looked some simple (but high performing) supervised learning ideas for estimating the class of individual documents (Naive Bayes), and for estimating proportions.

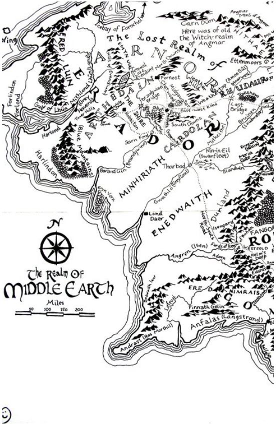
Where Are We?



We've looked some simple (but high performing) supervised learning ideas for estimating the class of individual documents (Naive Bayes), and for estimating proportions.

Now want to cover powerful, commonly used techniques from machine learning that appear in social science research:

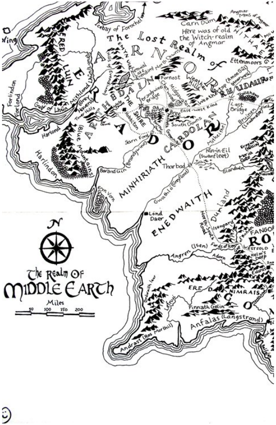
Where Are We?



We've looked some simple (but high performing) supervised learning ideas for estimating the class of individual documents (Naive Bayes), and for estimating proportions.

Now want to cover powerful, commonly used techniques from machine learning that appear in social science research: $\boxed{\text{SVM}}$, KNN, CART etc.

Where Are We?



We've looked some simple (but high performing) supervised learning ideas for estimating the class of individual documents (Naive Bayes), and for estimating proportions.

Now want to cover powerful, commonly used techniques from machine learning that appear in social science research: $\boxed{\text{SVM}}$, KNN, CART etc.

These techniques involve some important decisions about the bias-variance tradeoff, and the use of (cross) validation in checking model performance and selecting the best model.

Remember. . .

Remember. . .

Unsupervised techniques:

Remember...

Unsupervised techniques: learning (hidden or latent) structure in unlabeled data.

e.g. PCA of legislators's votes:

Remember...

Unsupervised techniques: learning (hidden or latent) structure in unlabeled data.

e.g. PCA of legislators's votes: want to see how they are organized—

Remember...

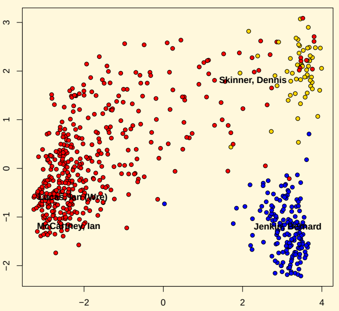
Unsupervised techniques: learning (hidden or latent) structure in unlabeled data.

e.g. PCA of legislators's votes: want to see how they are organized—by party? by ideology? by race?

Remember...

Unsupervised techniques: learning (hidden or latent) structure in unlabeled data.

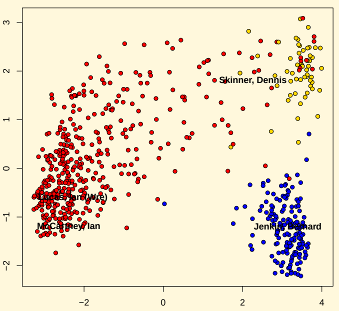
e.g. PCA of legislators's votes: want to see how they are organized—by party? by ideology? by race?



Remember. . .

Unsupervised techniques: learning (hidden or latent) structure in unlabeled data.

e.g. PCA of legislators's votes: want to see how they are organized—by party? by ideology? by race?

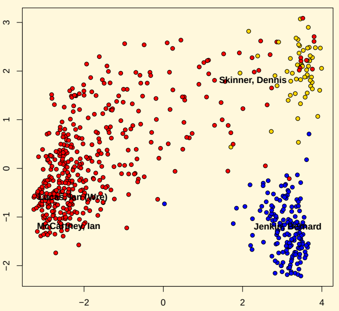


Supervised techniques:

Remember. . .

Unsupervised techniques: learning (hidden or latent) structure in unlabeled data.

e.g. PCA of legislators's votes: want to see how they are organized—by party? by ideology? by race?

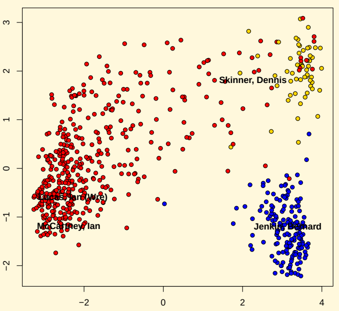


Supervised techniques: learning relationship between inputs and a labeled set of outputs.

Remember...

Unsupervised techniques: learning (hidden or latent) structure in unlabeled data.

e.g. PCA of legislators's votes: want to see how they are organized—by party? by ideology? by race?



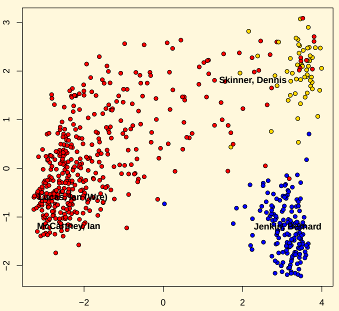
Supervised techniques: learning relationship between inputs and a labeled set of outputs.

e.g. opinion mining:

Remember...

Unsupervised techniques: learning (hidden or latent) structure in unlabeled data.

e.g. PCA of legislators's votes: want to see how they are organized—by party? by ideology? by race?



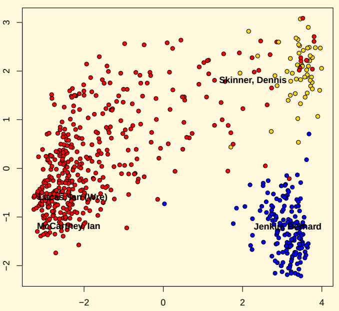
Supervised techniques: learning relationship between inputs and a labeled set of outputs.

e.g. opinion mining: what makes a critic like or dislike a movie ($y_i \in \{0, 1\}$)?

Remember...

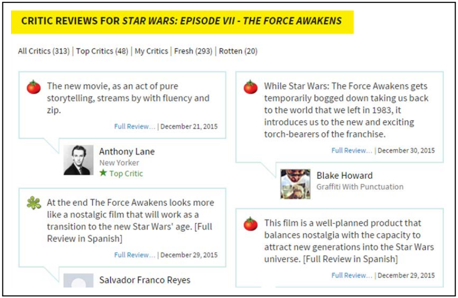
[Unsupervised]{} techniques: learning (hidden or latent) structure in unlabeled data.

e.g. PCA of legislators's votes: want to see how they are organized—by party? by ideology? by race?



Supervised techniques: learning relationship between inputs and a labeled set of outputs.

e.g. opinion mining: what makes a critic like or dislike a movie ($y_i \in \{0, 1\}$)?



Workflow of Supervised Learning: Bias/Variance Tradeoff

Workflow of Supervised Techniques

Workflow of Supervised Techniques

1 Obtain/code the training set and decide on relevant features (preprocess).

Workflow of Supervised Techniques

1. Obtain/code the training set and decide on relevant features (preprocess).
2. Decide on the algorithm,

Workflow of Supervised Techniques

1. Obtain/code the training set and decide on relevant features (preprocess).
2. Decide on the algorithm, possibly matched in some way to nature of problem.

Workflow of Supervised Techniques

1. Obtain/code the training set and decide on relevant features (preprocess).
2. Decide on the algorithm, possibly matched in some way to nature of problem.
3. Adjust algorithm for optimal performance,

Workflow of Supervised Techniques

1. Obtain/code the training set and decide on relevant features (preprocess).
2. Decide on the algorithm, possibly matched in some way to nature of problem.
3. Adjust algorithm for optimal performance, perhaps using validation set and/or some kind of cross-validation.

Workflow of Supervised Techniques

1. Obtain/code the training set and decide on relevant features (preprocess).
2. Decide on the algorithm, possibly matched in some way to nature of problem.
3. Adjust algorithm for optimal performance, perhaps using validation set and/or some kind of cross-validation.
4. Report accuracy in test set,

Workflow of Supervised Techniques

1. Obtain/code the training set and decide on relevant features (preprocess).
2. Decide on the algorithm, possibly matched in some way to nature of problem.
3. Adjust algorithm for optimal performance, perhaps using validation set and/or some kind of cross-validation.
4. Report accuracy in test set, possibly combine with other learners in ensemble.

Notes and Issues

Notes and Issues

Supervised techniques are about learning relationship between X and labeled data.

Notes and Issues

Supervised techniques are about learning relationship between X and labeled data. Often used interchangeably with *machine learning*: idea that computers could ‘learn’ relationships without specific programming.

Notes and Issues

Supervised techniques are about learning relationship between X and labeled data. Often used interchangeably with *machine learning*: idea that computers could 'learn' relationships without specific programming.

Often used when $p >> n$:

Notes and Issues

Supervised techniques are about learning relationship between X and labeled data. Often used interchangeably with *machine learning*: idea that computers could 'learn' relationships without specific programming.

Often used when $p >> n$: number of parameters (e.g. coefficients on terms) is far larger than number of observations (e.g. speakers or documents).

Notes and Issues

Supervised techniques are about learning relationship between X and labeled data. Often used interchangeably with *machine learning*: idea that computers could 'learn' relationships without specific programming.

Often used when $p >> n$: number of parameters (e.g. coefficients on terms) is far larger than number of observations (e.g. speakers or documents).

$\rightarrow$ results in general curse of dimensionality wherein feature matrix is large (e.g. $100k$ columns) and sparse and thus obtaining meaningful estimates is difficult.

Notes and Issues

Supervised techniques are about learning relationship between X and labeled data. Often used interchangeably with *machine learning*: idea that computers could 'learn' relationships without specific programming.

Often used when $p >> n$: number of parameters (e.g. coefficients on terms) is far larger than number of observations (e.g. speakers or documents).

$\rightarrow$ results in general curse of dimensionality wherein feature matrix is large (e.g. $100k$ columns) and sparse and thus obtaining meaningful estimates is difficult.

So techniques may require careful tuning of *regularization parameters* to obtain good performance.

Notes and Issues II

Notes and Issues II

Once we have the training set we face a dilemma. . .

Notes and Issues II

Once we have the training set we face a dilemma...

1. we can use our technique to fit a very complicated model to this data (perfectly),

Notes and Issues II

Once we have the training set we face a dilemma. . .

1 we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements.

Notes and Issues II

Once we have the training set we face a dilemma...

1 we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

Notes and Issues II

Once we have the training set we face a dilemma...

1 we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

$\rightarrow$ we have overfit to our training set,

Notes and Issues II

Once we have the training set we face a dilemma. . .

1 we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

$\rightarrow$ we have overfit to our training set, and may do poorly on our test set.

Notes and Issues II

Once we have the training set we face a dilemma...

1 we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

$\rightarrow$ we have overfit to our training set, and may do poorly on our test set.

2 we can use be more relaxed about the fit of our algorithm in the training set,

Notes and Issues II

Once we have the training set we face a dilemma...

1. we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

$\rightarrow$ we have overfit to our training set, and may do poorly on our test set.

2. we can use be more relaxed about the fit of our algorithm in the training set, and accept some imperfections in performance.

Notes and Issues II

Once we have the training set we face a dilemma...

1 we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

→ we have overfit to our training set, and may do poorly on our test set.

2 we can use be more relaxed about the fit of our algorithm in the training set, and accept some imperfections in performance. This avoids high variance,

Notes and Issues II

Once we have the training set we face a dilemma...

1. we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

$\rightarrow$ we have overfit to our training set, and may do poorly on our test set.

2. we can use be more relaxed about the fit of our algorithm in the training set, and accept some imperfections in performance. This avoids high variance, but may induce bias in the sense that we miss important relationships in the data.

Notes and Issues II

Once we have the training set we face a dilemma...

1 we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

$\rightarrow$ we have overfit to our training set, and may do poorly on our test set.

2 we can use be more relaxed about the fit of our algorithm in the training set, and accept some imperfections in performance. This avoids high variance, but may induce bias in the sense that we miss important relationships in the data.

$\rightarrow$ we have underfit to our training set,

Notes and Issues II

Once we have the training set we face a dilemma...

1. we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

$\rightarrow$ we have overfit to our training set, and may do poorly on our test set.

2. we can use be more relaxed about the fit of our algorithm in the training set, and accept some imperfections in performance. This avoids high variance, but may induce bias in the sense that we miss important relationships in the data.

$\rightarrow$ we have underfit to our training set, and may do poorly on our test set.

Notes and Issues II

Once we have the training set we face a dilemma...

1 we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

→ we have overfit to our training set, and may do poorly on our test set.

2 we can use be more relaxed about the fit of our algorithm in the training set, and accept some imperfections in performance. This avoids high variance, but may induce bias in the sense that we miss important relationships in the data.

→ we have underfit to our training set, and may do poorly on our test set.

So managing the *bias-variance tradeoff* is a key element of supervised learning,

Notes and Issues II

Once we have the training set we face a dilemma...

1. we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

→ we have overfit to our training set, and may do poorly on our test set.

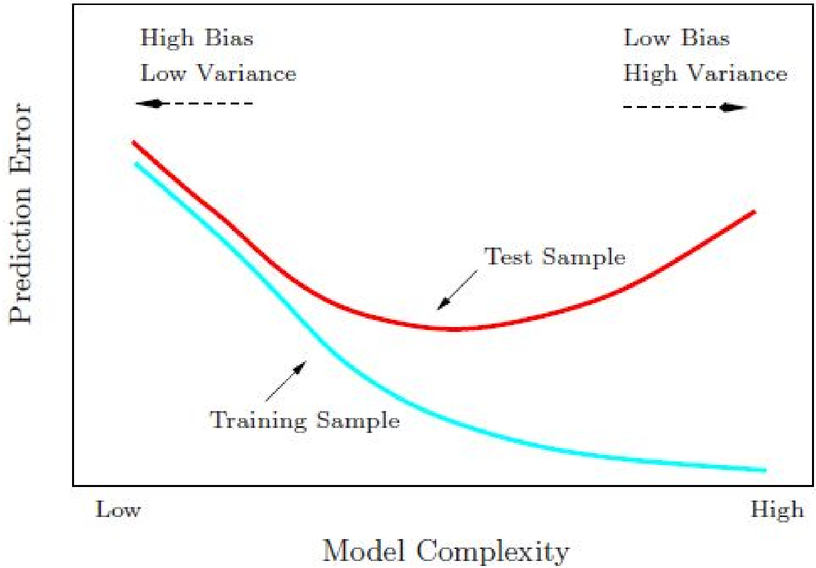
2. we can use be more relaxed about the fit of our algorithm in the training set, and accept some imperfections in performance. This avoids high variance, but may induce bias in the sense that we miss important relationships in the data.

→ we have underfit to our training set, and may do poorly on our test set.

So managing the *bias-variance tradeoff* is a key element of supervised learning, and we may need to tune our algorithms with that in mind.

Bias-Variance Tradeoff (Hastie et al, p38)

Bias-Variance Tradeoff (Hastie et al, p38)



Cross-Validation

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'),

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,
2. repeat the following k times (folds):

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,
2. repeat the following k times (folds):
    1. grab one of the k chunks as a validation set (each only used once)

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,
2. repeat the following k times (folds):
   1. grab one of the k chunks as a validation set (each only used once)
   2. grab the other $k - 1$ chunks as a training set

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,
2. repeat the following k times (folds):
   1. grab one of the k chunks as a validation set (each only used once)
   2. grab the other $k - 1$ chunks as a training set
   3. test on the validation set,

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,
2. repeat the following k times (folds):
    1. grab one of the k chunks as a validation set (each only used once)
    2. grab the other $k - 1$ chunks as a training set
    3. test on the validation set, record prediction error

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,
2. repeat the following k times (folds):
   1. grab one of the k chunks as a validation set (each only used once)
   2. grab the other $k - 1$ chunks as a training set
   3. test on the validation set, record prediction error
3. Average over runs to get prediction error estimate.

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,
2. repeat the following k times (folds):
    1. grab one of the k chunks as a validation set (each only used once)
    2. grab the other $k - 1$ chunks as a training set
    3. test on the validation set, record prediction error
3. Average over runs to get prediction error estimate.

→ Can follow same steps for models of different specifications; variant on this approach can be used for model selection, directly.

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

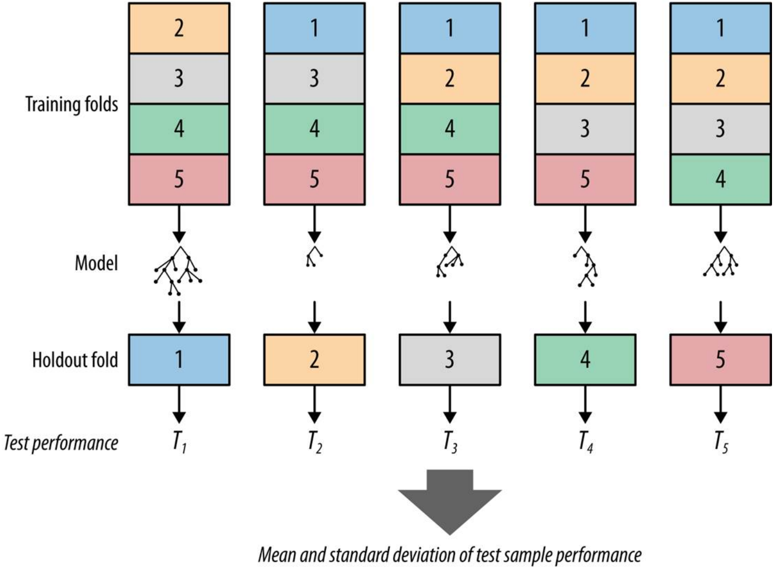
Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,
2. repeat the following k times (folds):
   1. grab one of the k chunks as a validation set (each only used once)
   2. grab the other $k - 1$ chunks as a training set
   3. test on the validation set, record prediction error
3. Average over runs to get prediction error estimate.

$\rightarrow$ Can follow same steps for models of different specifications; variant on this approach can be used for model selection, directly.

Graphically

Graphically



Support Vector Machines

Motivating Example Diermeier et al, 2011

Motivating Example Diermeier et al, 2011

Motivating Example Diermeier et al, 2011

Diermeier et al want to know what terms are most indicative of conservative or liberal positions in legislative debates.

Motivating Example Diermeier et al, 2011

Diermeier et al want to know what terms are most indicative of conservative or liberal positions in legislative debates.

Study the speeches of the 25 most liberal and the 25 most conservative senators (1989–2004),

Diermeier et al want to know what terms are most indicative of conservative or liberal positions in legislative debates.

Study the speeches of the 25 most liberal and the 25 most conservative senators (1989–2004), selected based on their NOMINATE scores

Motivating Example Diermeier et al, 2011

Diermeier et al want to know what terms are most indicative of conservative or liberal positions in legislative debates.

Study the speeches of the 25 most liberal and the 25 most conservative senators (1989–2004), selected based on their NOMINATE scores (from roll call behavior, only)

Motivating Example Diermeier et al, 2011

Diermeier et al want to know what terms are most indicative of conservative or liberal positions in legislative debates.

Study the speeches of the 25 most liberal and the 25 most conservative senators (1989–2004), selected based on their NOMINATE scores (from roll call behavior, only)

Code the Republicans as $+1$, Dems as -1:

Motivating Example Diermeier et al, 2011

Diermeier et al want to know what terms are most indicative of conservative or liberal positions in legislative debates.

Study the speeches of the 25 most liberal and the 25 most conservative senators (1989–2004), selected based on their NOMINATE scores (from roll call behavior, only)

Code the Republicans as $+1$, Dems as -1: $y_i \in \{-1, +1\}$

Motivating Example Diermeier et al, 2011

Diermeier et al want to know what terms are most indicative of conservative or liberal positions in legislative debates.

Study the speeches of the 25 most liberal and the 25 most conservative senators (1989–2004), selected based on their NOMINATE scores (from roll call behavior, only)

Code the Republicans as $+1$, Dems as -1: $y_i \in \{-1, +1\}$

Have the (stemmed, stopped, weighted etc) speech term matrix for each Senator as X.

Methodological Problem

Methodological Problem

Want to know the relationship between using different terms and ideological views.

Methodological Problem

Want to know the relationship between using different terms and ideological views.

e.g. is ‘`Ethanol`’ a ‘Democrat’ term or a ‘Republican’ term,

Methodological Problem

Want to know the relationship between using different terms and ideological views.

e.g. is ‘`Ethanol`’ a ‘Democrat’ term or a ‘Republican’ term, and to what degree?

Methodological Problem

Want to know the relationship between using different terms and ideological views.

e.g. is '`Ethanol`' a 'Democrat' term or a 'Republican' term, and to what degree?

Their training set is speech output of most extreme Senators between the 101st and 107th Congress

Methodological Problem

Want to know the relationship between using different terms and ideological views.

e.g. is '`Ethanol`' a 'Democrat' term or a 'Republican' term, and to what degree?

Their training set is speech output of most extreme Senators between the 101st and 107th Congress

Their test set is speech output of most extreme Senators in 108th Congress.

Methodological Problem

Want to know the relationship between using different terms and ideological views.

e.g. is '`Ethanol`' a 'Democrat' term or a 'Republican' term, and to what degree?

Their training set is speech output of most extreme Senators between the 101st and 107th Congress

Their test set is speech output of most extreme Senators in 108th Congress.

What method to use?

Support Vector Machines

Idea a classifier that builds a model from a binary labeled training set and returns an optimal hyperplane that puts new examples into one of the categories non-probabilistically.

Support Vector Machines

Idea a classifier that builds a model from a binary labeled training set and returns an optimal hyperplane that puts new examples into one of the categories non-probabilistically.

Here We have 10 Senators that belong to one of two classes: five Republicans, five Democrats.

Support Vector Machines

Idea a classifier that builds a model from a binary labeled training set and returns an optimal hyperplane that puts new examples into one of the categories non-probabilistically.

Here We have 10 Senators that belong to one of two classes: five Republicans, five Democrats. $y_i \in \{-1, +1\}$

Each senator has a number of p features, which are the term weights from their speeches.

Support Vector Machines

Idea a classifier that builds a model from a binary labeled training set and returns an optimal hyperplane that puts new examples into one of the categories non-probabilistically.

Here We have 10 Senators that belong to one of two classes: five Republicans, five Democrats. $y_i \in \{-1, +1\}$

Each senator has a number of p features, which are the term weights from their speeches. To make things simple, suppose that $p = 2$:

Support Vector Machines

Idea a classifier that builds a model from a binary labeled training set and returns an optimal hyperplane that puts new examples into one of the categories non-probabilistically.

Here We have 10 Senators that belong to one of two classes: five Republicans, five Democrats. $y_i \in \{-1, +1\}$

Each senator has a number of p features, which are the term weights from their speeches. To make things simple, suppose that $p = 2$: there are (only) two features, x_1 and x_2 per observation.

Support Vector Machines

Idea a classifier that builds a model from a binary labeled training set and returns an optimal hyperplane that puts new examples into one of the categories non-probabilistically.

Here We have 10 Senators that belong to one of two classes: five Republicans, five Democrats. $y_i \in \{-1, +1\}$

Each senator has a number of p features, which are the term weights from their speeches. To make things simple, suppose that $p = 2$: there are (only) two features, x_1 and x_2 per observation.

Assume that the observations are linearly separable:

Support Vector Machines

Idea a classifier that builds a model from a binary labeled training set and returns an optimal hyperplane that puts new examples into one of the categories non-probabilistically.

Here We have 10 Senators that belong to one of two classes: five Republicans, five Democrats. $y_i \in \{-1, +1\}$

Each senator has a number of p features, which are the term weights from their speeches. To make things simple, suppose that $p = 2$: there are (only) two features, x_1 and x_2 per observation.

Assume that the observations are linearly separable: if we plot the Senators in two dimensions (x_1, x_2), we can divide them (perfectly) into the two parties using a straight line.

Support Vector Machines

Idea a classifier that builds a model from a binary labeled training set and returns an optimal hyperplane that puts new examples into one of the categories non-probabilistically.

Here We have 10 Senators that belong to one of two classes: five Republicans, five Democrats. $y_i \in \{-1, +1\}$

Each senator has a number of p features, which are the term weights from their speeches. To make things simple, suppose that $p = 2$: there are (only) two features, x_1 and x_2 per observation.

Assume that the observations are linearly separable: if we plot the Senators in two dimensions (x_1, x_2), we can divide them (perfectly) into the two parties using a straight line.

$\rightarrow$ of course,

Support Vector Machines

Idea a classifier that builds a model from a binary labeled training set and returns an optimal hyperplane that puts new examples into one of the categories non-probabilistically.

Here We have 10 Senators that belong to one of two classes: five Republicans, five Democrats. $y_i \in \{-1, +1\}$

Each senator has a number of p features, which are the term weights from their speeches. To make things simple, suppose that $p = 2$: there are (only) two features, x_1 and x_2 per observation.

Assume that the observations are linearly separable: if we plot the Senators in two dimensions (x_1, x_2), we can divide them (perfectly) into the two parties using a straight line.

$\rightarrow$ of course, a real problem would have p being large,

Support Vector Machines

Idea a classifier that builds a model from a binary labeled training set and returns an optimal hyperplane that puts new examples into one of the categories non-probabilistically.

Here We have 10 Senators that belong to one of two classes: five Republicans, five Democrats. $y_i \in \{-1, +1\}$

Each senator has a number of p features, which are the term weights from their speeches. To make things simple, suppose that $p = 2$: there are (only) two features, x_1 and x_2 per observation.

Assume that the observations are linearly separable: if we plot the Senators in two dimensions (x_1, x_2), we can divide them (perfectly) into the two parties using a straight line.

→ of course, a real problem would have p being large, and the space being very high dimensional,

Support Vector Machines

Idea a classifier that builds a model from a binary labeled training set and returns an optimal hyperplane that puts new examples into one of the categories non-probabilistically.

Here We have 10 Senators that belong to one of two classes: five Republicans, five Democrats. $y_i \in \{-1, +1\}$

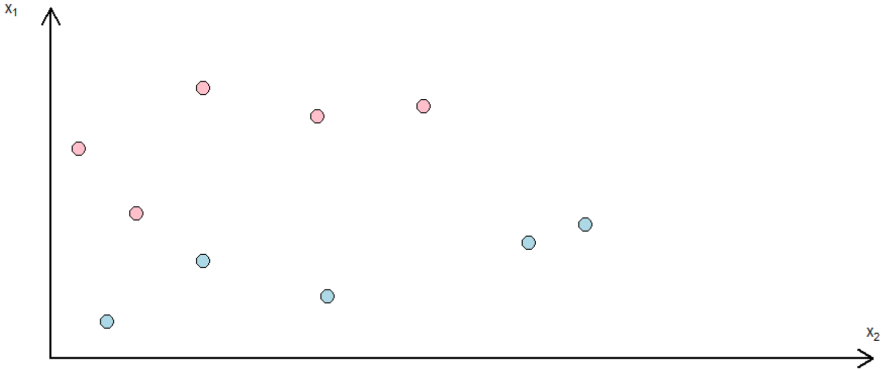
Each senator has a number of p features, which are the term weights from their speeches. To make things simple, suppose that $p = 2$: there are (only) two features, x_1 and x_2 per observation.

Assume that the observations are linearly separable: if we plot the Senators in two dimensions (x_1, x_2), we can divide them (perfectly) into the two parties using a straight line.

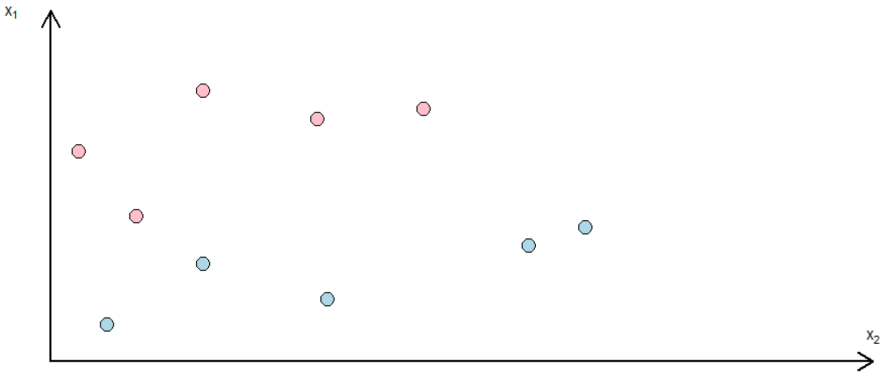
$\rightarrow$ of course, a real problem would have p being large, and the space being very high dimensional, but the logic is the same.

The 10 Senators

The 10 Senators

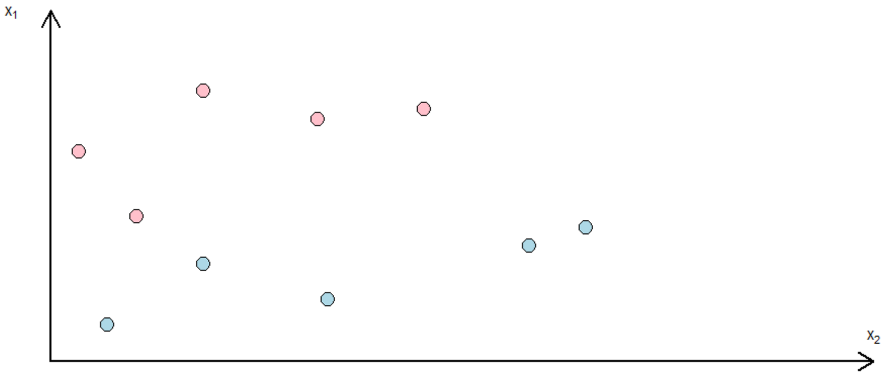


The 10 Senators



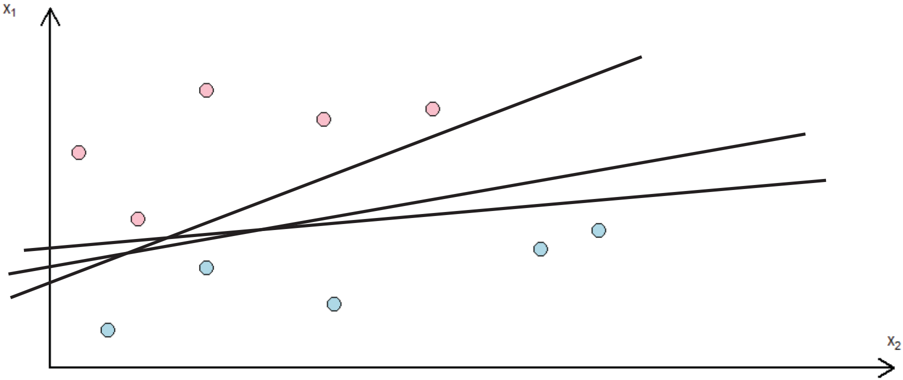
As the parties linearly separably?

The 10 Senators

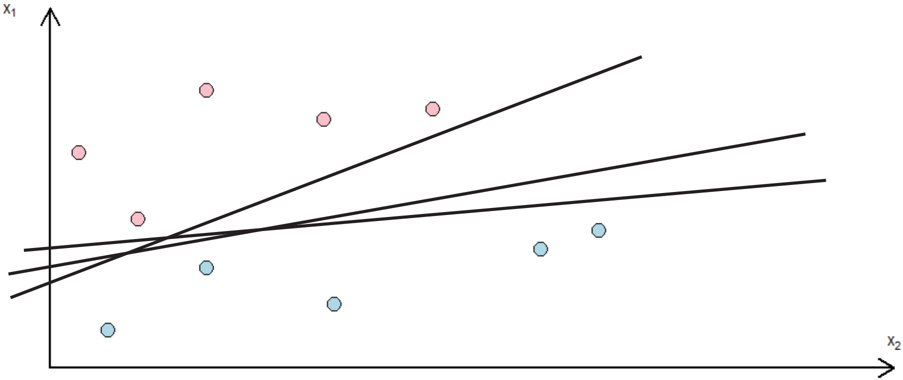


As the parties linearly separably? Where could you draw the line?

The 10 Senators

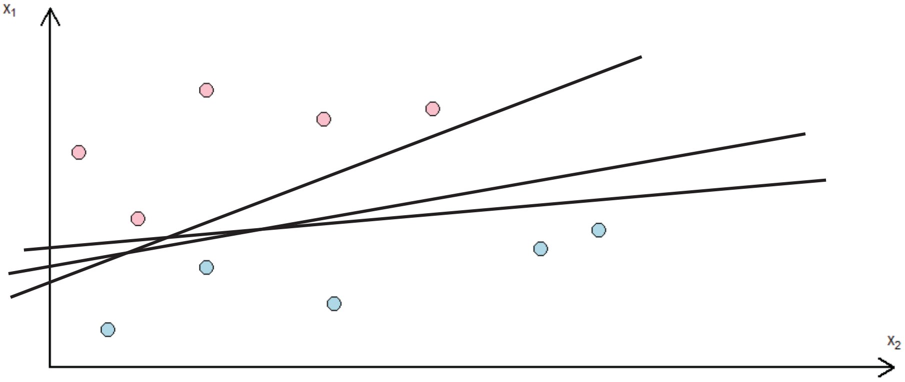


The 10 Senators



Which line should we prefer?

The 10 Senators



Which line should we prefer?

What is the Optimal Hyperplane?

What is the Optimal Hyperplane?

There are many possible lines.

What is the Optimal Hyperplane?

There are many possible lines.

But we want to avoid ones that pass close to the points:

What is the Optimal Hyperplane?

There are many possible lines.

But we want to avoid ones that pass close to the points: such lines will tend to be sensitive to noise and make classification errors with future examples.

What is the Optimal Hyperplane?

There are many possible lines.

But we want to avoid ones that pass close to the points: such lines will tend to be sensitive to noise and make classification errors with future examples.

So pick line that gives largest minimum distance from the training cases.

What is the Optimal Hyperplane?

There are many possible lines.

But we want to avoid ones that pass close to the points: such lines will tend to be sensitive to noise and make classification errors with future examples.

So pick line that gives largest minimum distance from the training cases. That is, the line that's as far as possible from the closest cases on both sides.

What is the Optimal Hyperplane?

There are many possible lines.

But we want to avoid ones that pass close to the points: such lines will tend to be sensitive to noise and make classification errors with future examples.

So pick line that gives largest minimum distance from the training cases. That is, the line that's as far as possible from the closest cases on both sides.

$\rightarrow$ That optimal line—the separating hyperplane—is the maximum margin hyperplane. It will maximize the margin of the training data.

How to get it I: Notation Variants

How to get it I: Notation Variants

We can write any line as $y = ax + b$ or $y - ax - b = 0$.

How to get it I: Notation Variants

We can write any line as $y = ax + b$ or $y - ax - b = 0$.

In machine learning,

How to get it I: Notation Variants

We can write any line as $y = ax + b$ or $y - ax - b = 0$.

In machine learning, often rewrite terms as $\mathbf{w} = [-b, -a, 1]$ and $\mathbf{x} = [1, x, y]$

How to get it I: Notation Variants

We can write any line as $y = ax + b$ or $y - ax - b = 0$.

In machine learning, often rewrite terms as $\mathbf{w} = [-b, -a, 1]$ and $\mathbf{x} = [1, x, y]$

$\rightarrow$ $\mathbf{w}^T\mathbf{x} = -b \times 1 + (-a) \times x + 1 \times y$

How to get it I: Notation Variants

We can write any line as $y = ax + b$ or $y - ax - b = 0$.

In machine learning, often rewrite terms as $\mathbf{w} = [-b, -a, 1]$ and $\mathbf{x} = [1, x, y]$

$\rightarrow$ $\mathbf{w}^T\mathbf{x} = -b \times 1 + (-a) \times x + 1 \times y = y - ax - b = 0$.

How to get it I: Notation Variants

We can write any line as $y = ax + b$ or $y - ax - b = 0$.

In machine learning, often rewrite terms as $\mathbf{w} = [-b, -a, 1]$ and $\mathbf{x} = [1, x, y]$

$\rightarrow$ $\mathbf{w}^T\mathbf{x} = -b \times 1 + (-a) \times x + 1 \times y = y - ax - b = 0$.

or use dot product:

How to get it I: Notation Variants

We can write any line as $y = ax + b$ or $y - ax - b = 0$.

In machine learning, often rewrite terms as $\mathbf{w} = [-b, -a, 1]$ and $\mathbf{x} = [1, x, y]$

$\rightarrow$ $\mathbf{w}^T\mathbf{x} = -b \times 1 + (-a) \times x + 1 \times y = y - ax - b = 0$.

or use dot product: $\mathbf{w} \cdot \mathbf{x} = 0$

How to get it I: Notation Variants

We can write any line as $y = ax + b$ or $y - ax - b = 0$.

In machine learning, often rewrite terms as $\mathbf{w} = [-b, -a, 1]$ and $\mathbf{x} = [1, x, y]$

$\rightarrow$ $\mathbf{w}^T\mathbf{x} = -b \times 1 + (-a) \times x + 1 \times y = y - ax - b = 0$.

or use dot product: $\mathbf{w} \cdot \mathbf{x} = 0$

Also popular:

How to get it I: Notation Variants

We can write any line as $y = ax + b$ or $y - ax - b = 0$.

In machine learning, often rewrite terms as $\mathbf{w} = [-b, -a, 1]$ and $\mathbf{x} = [1, x, y]$

$\rightarrow$ $\mathbf{w}^T\mathbf{x} = -b \times 1 + (-a) \times x + 1 \times y = y - ax - b = 0$.

or use dot product: $\mathbf{w} \cdot \mathbf{x} = 0$

Also popular: can use a two dimensional form—

How to get it I: Notation Variants

We can write any line as $y = ax + b$ or $y - ax - b = 0$.

In machine learning, often rewrite terms as $\mathbf{w} = [-b, -a, 1]$ and $\mathbf{x} = [1, x, y]$

$\rightarrow$ $\mathbf{w}^T\mathbf{x} = -b \times 1 + (-a) \times x + 1 \times y = y - ax - b = 0.$

or use dot product: $\mathbf{w} \cdot \mathbf{x} = 0$

Also popular: can use a two dimensional form—

$\mathbf{w} = [-a, 1]$ and $\mathbf{x} = [x, y]$,

How to get it I: Notation Variants

We can write any line as $y = ax + b$ or $y - ax - b = 0$.

In machine learning, often rewrite terms as $\mathbf{w} = [-b, -a, 1]$ and $\mathbf{x} = [1, x, y]$

$\rightarrow$ $\mathbf{w}^T\mathbf{x} = -b \times 1 + (-a) \times x + 1 \times y = y - ax - b = 0$.

or use dot product: $\mathbf{w} \cdot \mathbf{x} = 0$

Also popular: can use a two dimensional form—

$\mathbf{w} = [-a, 1]$ and $\mathbf{x} = [x, y]$, so that $\mathbf{w} \cdot \mathbf{x} = y - ax$

How to get it I: Notation Variants

We can write any line as $y = ax + b$ or $y - ax - b = 0$.

In machine learning, often rewrite terms as $\mathbf{w} = [-b, -a, 1]$ and $\mathbf{x} = [1, x, y]$

$\rightarrow$ $\mathbf{w}^T\mathbf{x} = -b \times 1 + (-a) \times x + 1 \times y = y - ax - b = 0.$

or use dot product: $\mathbf{w} \cdot \mathbf{x} = 0$

Also popular: can use a two dimensional form—

$\mathbf{w} = [-a, 1]$ and $\mathbf{x} = [x, y]$, so that $\mathbf{w} \cdot \mathbf{x} = y - ax$

thus $\mathbf{w} \cdot \mathbf{x} - b = 0$

How to get it I: Notation Variants

We can write any line as $y = ax + b$ or $y - ax - b = 0$.

In machine learning, often rewrite terms as $\mathbf{w} = [-b, -a, 1]$ and $\mathbf{x} = [1, x, y]$

$\rightarrow$ $\mathbf{w}^T\mathbf{x} = -b \times 1 + (-a) \times x + 1 \times y = y - ax - b = 0.$

or use dot product: $\mathbf{w} \cdot \mathbf{x} = 0$

Also popular: can use a two dimensional form—

$\mathbf{w} = [-a, 1]$ and $\mathbf{x} = [x, y]$, so that $\mathbf{w} \cdot \mathbf{x} = y - ax$

thus $\mathbf{w} \cdot \mathbf{x} - b = 0$ (since $0 = y - ax - b$)

How to get it II: working with the equation

How to get it II: working with the equation

Our hyperplane (line) will separate the data, and will satisfy $\mathbf{w} \cdot \mathbf{x} - b = 0$

How to get it II: working with the equation

Our hyperplane (line) will separate the data, and will satisfy $\mathbf{w} \cdot \mathbf{x} - b = 0$

We can think of it as the line that is equidistant,

How to get it II: working with the equation

Our hyperplane (line) will separate the data, and will satisfy $\mathbf{w} \cdot \mathbf{x} - b = 0$

We can think of it as the line that is equidistant, i.e. half-way between,

How to get it II: working with the equation

Our hyperplane (line) will separate the data, and will satisfy $\mathbf{w} \cdot \mathbf{x} - b = 0$

We can think of it as the line that is equidistant, i.e. half-way between, two parallel hyperplanes (H_R and H_D) that separate the Reps from the Dems.

How to get it II: working with the equation

Our hyperplane (line) will separate the data, and will satisfy $\mathbf{w} \cdot \mathbf{x} - b = 0$

We can think of it as the line that is equidistant, i.e. half-way between, two parallel hyperplanes (H_R and H_D) that separate the Reps from the Dems.

Those parallel lines should be as far from each other as possible without misclassifying anybody:

How to get it II: working with the equation

Our hyperplane (line) will separate the data, and will satisfy $\mathbf{w} \cdot \mathbf{x} - b = 0$

We can think of it as the line that is equidistant, i.e. half-way between, two parallel hyperplanes (H_R and H_D) that separate the Reps from the Dems.

Those parallel lines should be as far from each other as possible without misclassifying anybody: the distance between them is the margin and we want that to be maximized.

How to get it II: working with the equation

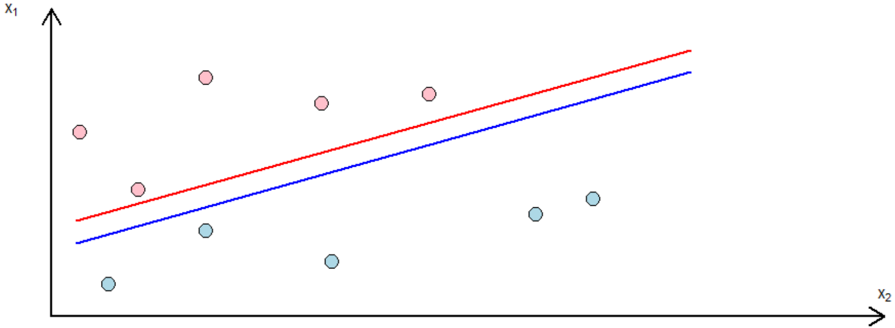
Our hyperplane (line) will separate the data, and will satisfy $\mathbf{w} \cdot \mathbf{x} - b = 0$

We can think of it as the line that is equidistant, i.e. half-way between, two parallel hyperplanes (H_R and H_D) that separate the Reps from the Dems.

Those parallel lines should be as far from each other as possible without misclassifying anybody: the distance between them is the margin and we want that to be maximized.

How Could We Do Better?

How Could We Do Better?



How to get it II: working with the equation

Our hyperplane (line) will separate the data, and will satisfy $\mathbf{w} \cdot \mathbf{x} - b = 0$

We can think of it as the line that is equidistant, i.e. half-way between, two parallel hyperplanes (H_R and H_D) that separate the Reps from the Dems.

Those parallel lines should be as far from each other as possible without misclassifying anybody: the distance between them is the margin and we want that to be maximized.

How to get it II: working with the equation

Our hyperplane (line) will separate the data, and will satisfy $\mathbf{w} \cdot \mathbf{x} - b = 0$

We can think of it as the line that is equidistant, i.e. half-way between, two parallel hyperplanes (H_R and H_D) that separate the Reps from the Dems.

Those parallel lines should be as far from each other as possible without misclassifying anybody: the distance between them is the margin and we want that to be maximized.

NB The hyperplane cannot be anywhere other than equidistant because then it will break the rule about ensuring the largest minimum distance.

The parallel hyperplanes

The parallel hyperplanes

We want all the Republicans ($y_i = 1$)—and *only* the Republicans—to be classified as Republicans (given their xs).

The parallel hyperplanes

We want all the Republicans ($y_i = 1$)—and *only* the Republicans—to be classified as Republicans (given their xs).

This means that 'their' hyperplane should 'capture' them all, and separate them fully from the Democrats (in our 2D space).

The parallel hyperplanes

We want all the Republicans ($y_i = 1$)—and *only* the Republicans—to be classified as Republicans (given their xs).

This means that 'their' hyperplane should 'capture' them all, and separate them fully from the Democrats (in our 2D space).

$\rightarrow$ requires that $\mathbf{w} \cdot \mathbf{x} - b \geq 1$, if $y_i = 1$

The parallel hyperplanes

We want all the Republicans ($y_i = 1$)—and *only* the Republicans—to be classified as Republicans (given their xs).

This means that 'their' hyperplane should 'capture' them all, and separate them fully from the Democrats (in our 2D space).

$\rightarrow$ requires that $\mathbf{w} \cdot \mathbf{x} - b \geq 1$, if $y_i = 1$

Same idea for the Democrats:

The parallel hyperplanes

We want all the Republicans ($y_i = 1$)—and *only* the Republicans—to be classified as Republicans (given their xs).

This means that 'their' hyperplane should 'capture' them all, and separate them fully from the Democrats (in our 2D space).

$\rightarrow$ requires that $\mathbf{w} \cdot \mathbf{x} - b \geq 1$, if $y_i = 1$

Same idea for the Democrats: $\mathbf{w} \cdot \mathbf{x} - b \leq -1$, if $y_i = -1$

The parallel hyperplanes

We want all the Republicans ($y_i = 1$)—and *only* the Republicans—to be classified as Republicans (given their xs).

This means that 'their' hyperplane should 'capture' them all, and separate them fully from the Democrats (in our 2D space).

$\rightarrow$ requires that $\mathbf{w} \cdot \mathbf{x} - b \geq 1$, if $y_i = 1$

Same idea for the Democrats: $\mathbf{w} \cdot \mathbf{x} - b \leq -1$, if $y_i = -1$

The distance between the two hyperplanes (H_R and H_D) we construct is the margin,

The parallel hyperplanes

We want all the Republicans ($y_i = 1$)—and *only* the Republicans—to be classified as Republicans (given their xs).

This means that 'their' hyperplane should 'capture' them all, and separate them fully from the Democrats (in our 2D space).

$\rightarrow$ requires that $\mathbf{w} \cdot \mathbf{x} - b \geq 1$, if $y_i = 1$

Same idea for the Democrats: $\mathbf{w} \cdot \mathbf{x} - b \leq -1$, if $y_i = -1$

The distance between the two hyperplanes (H_R and H_D) we construct is the margin, and its width is $\frac{2}{||\mathbf{w}||}$, where $||\mathbf{w}||$ is the norm of $\mathbf{w}$.

Optimization Problem

Optimization Problem

Need to find minimum value of $||\mathbf{w}||$, subject to the constraints we mentioned. This will ensure the widest possible margin.

Optimization Problem

Need to find minimum value of $||\mathbf{w}||$, subject to the constraints we mentioned. This will ensure the widest possible margin.

Well rearranging:

Optimization Problem

Need to find minimum value of $||\mathbf{w}||$, subject to the constraints we mentioned. This will ensure the widest possible margin.

Well rearranging: minimize $||\mathbf{w}||$ subject to $y_i(\mathbf{w} \cdot \mathbf{x} - b) \geq 1 \ \ \forall i$

Optimization Problem

Need to find minimum value of $||\mathbf{w}||$, subject to the constraints we mentioned. This will ensure the widest possible margin.

Well rearranging: minimize $||\mathbf{w}||$ subject to $y_i(\mathbf{w} \cdot \mathbf{x} - b) \geq 1 \;\; \forall i$

This will give us the two hyperplanes H_R and H_D, and the line equidistant between them will be our classifier.

Optimization Problem

Need to find minimum value of $||\mathbf{w}||$, subject to the constraints we mentioned. This will ensure the widest possible margin.

Well rearranging: minimize $||\mathbf{w}||$ subject to $y_i(\mathbf{w} \cdot \mathbf{x} - b) \geq 1 \;\; \forall i$

This will give us the two hyperplanes H_R and H_D, and the line equidistant between them will be our classifier.

We get the relevant value of b by noting that $\frac{b}{||w||}$ is the (perpendicular) distance from the optimal hyperplane to the origin.

Optimization Problem

Need to find minimum value of $||\mathbf{w}||$, subject to the constraints we mentioned. This will ensure the widest possible margin.

Well rearranging: minimize $||\mathbf{w}||$ subject to $y_i(\mathbf{w} \cdot \mathbf{x} - b) \geq 1 \;\; \forall i$

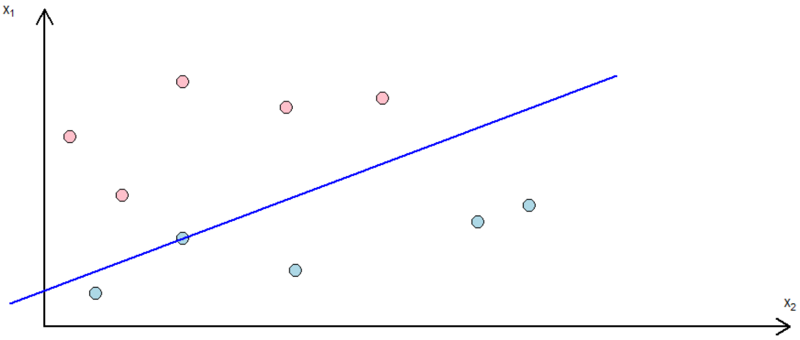
This will give us the two hyperplanes H_R and H_D, and the line equidistant between them will be our classifier.

We get the relevant value of b by noting that $\frac{b}{||w||}$ is the (perpendicular) distance from the optimal hyperplane to the origin.

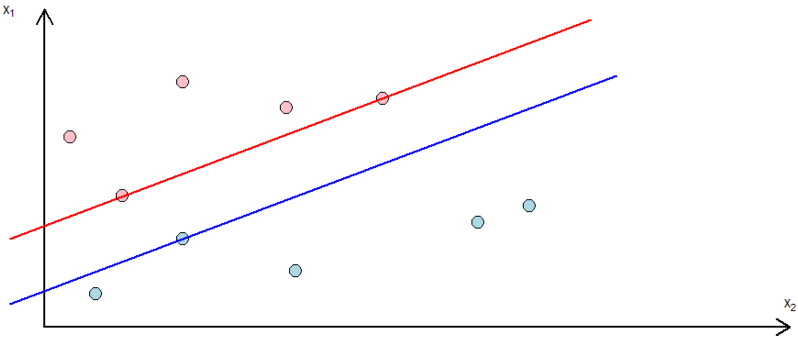
Turns out that the minimization of $||\mathbf{w}||$ is amenable to quadratic programming methods.

Graphically. . .

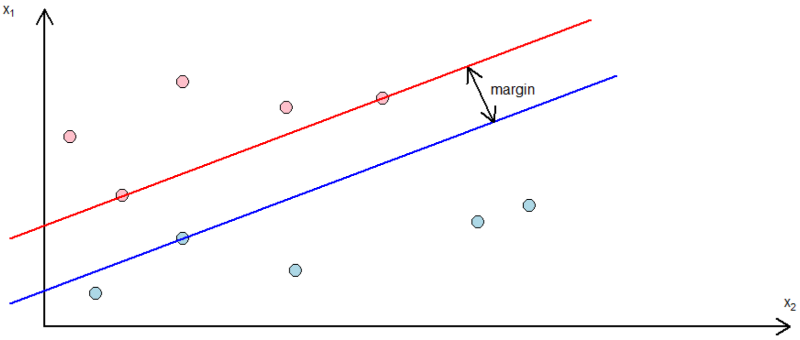
Graphically. . .



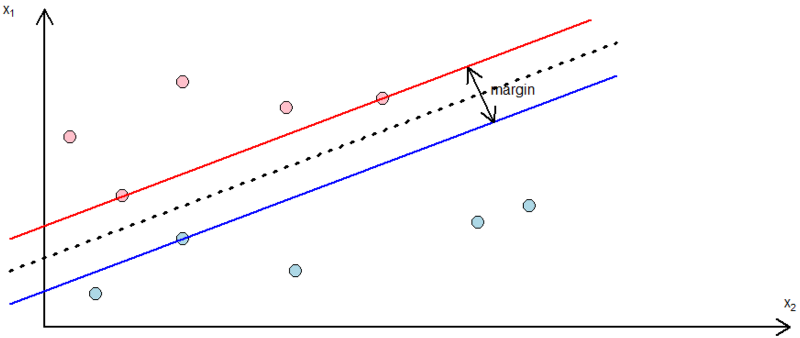
Graphically. . .



Graphically. . .



Graphically...



Notes

Notes

At least one of the Senators, in one of the parties, will be on ‘their’ line.

Notes

At least one of the Senators, in one of the parties, will be on ‘their’ line. Why?

Notes

At least one of the Senators, in one of the parties, will be on ‘their’ line. Why?

A: Suppose nobody is on either line.

Notes

At least one of the Senators, in one of the parties, will be on 'their' line. Why?

A: Suppose nobody is on either line. Then we must be able to move the lines 'apart' a little further and make the margin bigger...

Notes

At least one of the Senators, in one of the parties, will be on 'their' line. Why?

A: Suppose nobody is on either line. Then we must be able to move the lines 'apart' a little further and make the margin bigger...

But as soon as someone is pushed between the lines,

Notes

At least one of the Senators, in one of the parties, will be on 'their' line. Why?

A: Suppose nobody is on either line. Then we must be able to move the lines 'apart' a little further and make the margin bigger...

But as soon as someone is pushed between the lines, we've broken the rule about no misclassifications,

Notes

At least one of the Senators, in one of the parties, will be on 'their' line. Why?

A: Suppose nobody is on either line. Then we must be able to move the lines 'apart' a little further and make the margin bigger...

But as soon as someone is pushed between the lines, we've broken the rule about no misclassifications, and that's a constraint we have to follow.

Notes

At least one of the Senators, in one of the parties, will be on 'their' line. Why?

A: Suppose nobody is on either line. Then we must be able to move the lines 'apart' a little further and make the margin bigger...

But as soon as someone is pushed between the lines, we've broken the rule about no misclassifications, and that's a constraint we have to follow.

In fact, the points closest to the separating hyperplane will be the Senators lying on their (respective) parallel hyperplanes.

Notes

At least one of the Senators, in one of the parties, will be on 'their' line. Why?

A: Suppose nobody is on either line. Then we must be able to move the lines 'apart' a little further and make the margin bigger...

But as soon as someone is pushed between the lines, we've broken the rule about no misclassifications, and that's a constraint we have to follow.

In fact, the points closest to the separating hyperplane will be the Senators lying on their (respective) parallel hyperplanes.

We use the term support vectors to describe the training examples closest to the hyperplane.

Notes

At least one of the Senators, in one of the parties, will be on 'their' line. Why?

A: Suppose nobody is on either line. Then we must be able to move the lines 'apart' a little further and make the margin bigger. . .

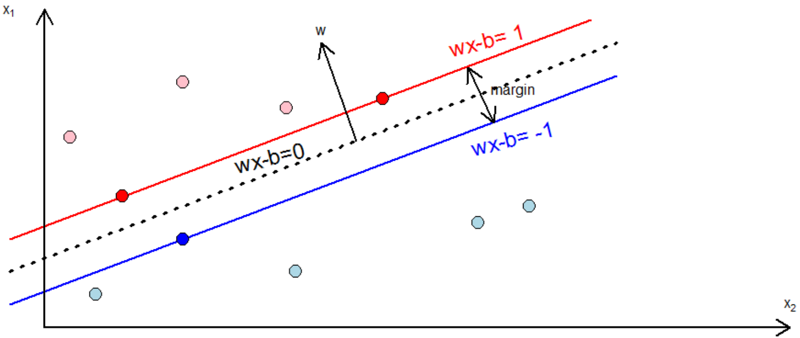
But as soon as someone is pushed between the lines, we've broken the rule about no misclassifications, and that's a constraint we have to follow.

In fact, the points closest to the separating hyperplane will be the Senators lying on their (respective) parallel hyperplanes.

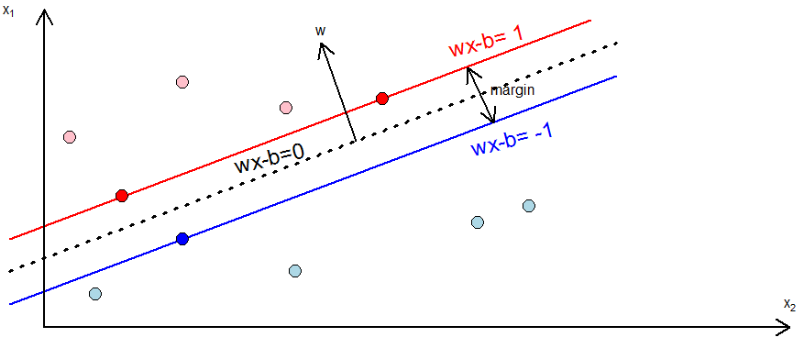
We use the term support vectors to describe the training examples closest to the hyperplane. Those support vectors completely determine where our maximum-margin hyperplane will be.

The Support Vectors

The Support Vectors

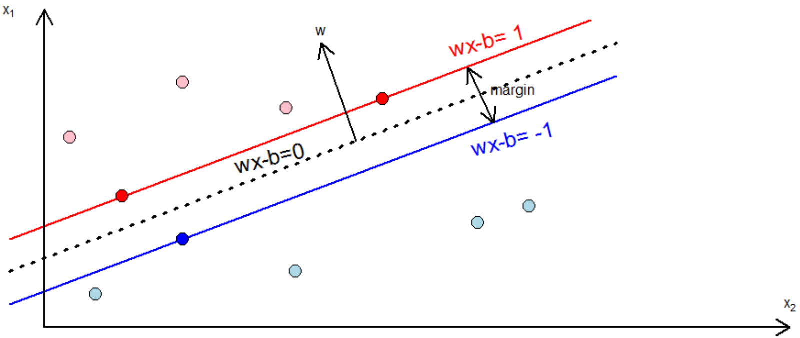


The Support Vectors



The support vectors lie on. . .

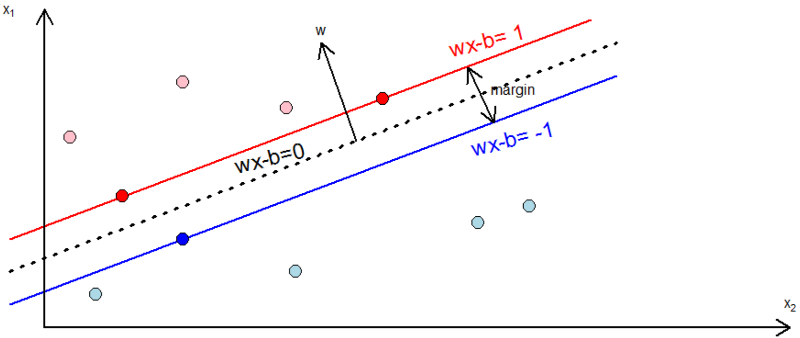
The Support Vectors



The support vectors lie on...

$\mathbf{w} \cdot \mathbf{x} - b = 1$

The Support Vectors



The support vectors lie on...

$\mathbf{w} \cdot \mathbf{x} - b = 1$ or $\mathbf{w} \cdot \mathbf{x} - b = -1$

SVM weights

SVM weights

So We have a hyperplane that separates the Democrats from the Republicans.

SVM weights

So We have a hyperplane that separates the Democrats from the Republicans.

The vector **w** is orthogonal to that, and when we multiply it (dot product) by **x** (and add b term) we get the predicted class of a 'new' Senator.

SVM weights

So We have a hyperplane that separates the Democrats from the Republicans.

The vector **w** is orthogonal to that, and when we multiply it (dot product) by **x** (and add b term) we get the predicted class of a 'new' Senator.

i.e. if the product is positive → Republican; if negative → Democrat

SVM weights

So We have a hyperplane that separates the Democrats from the Republicans.

The vector **w** is orthogonal to that, and when we multiply it (dot product) by **x** (and add b term) we get the predicted class of a 'new' Senator.

i.e. if the product is positive $\rightarrow$ Republican; if negative $\rightarrow$ Democrat

Plus for each feature, $x_1, x_2, \ldots$, the vector **w** gives us a weight.

SVM weights

So We have a hyperplane that separates the Democrats from the Republicans.

The vector **w** is orthogonal to that, and when we multiply it (dot product) by **x** (and add b term) we get the predicted class of a 'new' Senator.

i.e. if the product is positive → Republican; if negative → Democrat

Plus for each feature, $x_1, x_2, \ldots$, the vector **w** gives us a weight. The absolute size of the weights—relative to one another—is a measure of how important that feature is for separating the Senators into Democrat and Republican.

SVM weights

So We have a hyperplane that separates the Democrats from the Republicans.

The vector **w** is orthogonal to that, and when we multiply it (dot product) by **x** (and add b term) we get the predicted class of a 'new' Senator.

i.e. if the product is positive $\rightarrow$ Republican; if negative $\rightarrow$ Democrat

Plus for each feature, $x_1, x_2, \ldots$, the vector **w** gives us a weight. The absolute size of the weights—relative to one another—is a measure of how important that feature is for separating the Senators into Democrat and Republican.

NB SVMs typically have few features with non-zero weights,

SVM weights

So We have a hyperplane that separates the Democrats from the Republicans.

The vector **w** is orthogonal to that, and when we multiply it (dot product) by **x** (and add b term) we get the predicted class of a 'new' Senator.

i.e. if the product is positive $\rightarrow$ Republican; if negative $\rightarrow$ Democrat

Plus for each feature, $x_1, x_2, \ldots$, the vector **w** gives us a weight. The absolute size of the weights—relative to one another—is a measure of how important that feature is for separating the Senators into Democrat and Republican.

NB SVMs typically have few features with non-zero weights, and those that are non-zero come from the support vectors (the 'important' observations)

Back to Diermeier et al, 2011

Back to Diermeier et al, 2011

Achieve 92% accuracy (!)

Back to Diermeier et al, 2011

Achieve 92% accuracy (!)

Sort words according to coefficients: very positive weights imply conservative words; very negative weights imply liberal words

Back to Diermeier et al, 2011

Achieve 92% accuracy (!)

Sort words according to coefficients: very positive weights imply conservative words; very negative weights imply liberal words. Argue that it is 'values' rather than economics that separates liberals from conservatives.

Back to Diermeier et al, 2011

Achieve 92% accuracy (!)

Sort words according to coefficients: very positive weights imply conservative words; very negative weights imply liberal words. Argue that it is ‘values’ rather than economics that separates liberals from conservatives.

Words			
Liberal		Conservative	
FAS: −199.49	SBA: −113.10	habeas: 193.55	homosexual: 103.07
Ethanol: −198.92	Nursing: −109.38	CFTC: 187.16	everglades: 102.87
Wealthiest: −159.74	Providence: −108.73	surtax: 151.81	tower: 101.67
Collider: −142.28	Arctic: −108.30	marriage: 145.79	tripartisan: 101.23
WIC: −140.14	Orange: −107.98	cloning: 141.71	PRC: 102.90
ILO: −139.89	Glaxo: −107.81	tritium: 133.49	scouts: 97.55
Handgun: −129.01	Libraries: −107.70	ranchers: 132.95	nashua: 99.32
Lobbyists: −128.95	Disabilities: −106.44	BTU: 121.92	ballistic: 97.22
Enron: −127.71	Prescription: −106.31	grazing: 121.59	salting: 94.28
Fishery: −127.30	NIH: −105.52	unfunded: 120.82	abortion: 91.94
Hydrogen: −122.59	Lobbying: −105.35	catfish: 120.82	NTSB: 93.81
Souter: −121.40	NRA: −105.20	IRS: 114.91	Haiti: 97.28
PTSD: −119.87	Trident: −104.15	unborn: 111.88	PAC: 92.85
Gun: −119.52	RNC: −103.46	Taiwan: 111.13	taxing: 90.39

Beyond basic (hard margin) SVM

Wait. . .

Wait...

What's so special about SVM?

Wait. . .

What's so special about SVM? If we have $y_i \in \{0, 1\}$ why not just use logistic regression,

Wait...

What's so special about SVM? If we have $y_i \in \{0, 1\}$ why not just use logistic regression, regularized in some way (RLR) to avoid over-fitting?

Wait...

What's so special about SVM? If we have $y_i \in \{0, 1\}$ why not just use logistic regression, regularized in some way (RLR) to avoid over-fitting?

→ yes, this can work well and is commonly used in machine learning.

Wait...

What's so special about SVM? If we have $y_i \in \{0, 1\}$ why not just use logistic regression, regularized in some way (RLR) to avoid over-fitting?

$\rightarrow$ yes, this can work well and is commonly used in machine learning.

But if you want to know the decision boundary then SVM might be a better choice.

Wait...

What's so special about SVM? If we have $y_i \in \{0, 1\}$ why not just use logistic regression, regularized in some way (RLR) to avoid over-fitting?

$\rightarrow$ yes, this can work well and is commonly used in machine learning.

But if you want to know the decision boundary then SVM might be a better choice.

$\rightarrow$ RLR will optimize probabilities of class membership, rather than just trying to learn the boundary (simpler, more direct task).

Wait...

What's so special about SVM? If we have $y_i \in \{0, 1\}$ why not just use logistic regression, regularized in some way (RLR) to avoid over-fitting?

$\rightarrow$ yes, this can work well and is commonly used in machine learning.

But if you want to know the decision boundary then SVM might be a better choice.

$\rightarrow$ RLR will optimize probabilities of class membership, rather than just trying to learn the boundary (simpler, more direct task).

BTW RLR can cope well with noise, and (hard margin) SVM will struggle if there is no linear seperability...

What if. . .

What if. . .

The Senators were not linearly separable? ('soft margin' SVM problem)

What if. . .

The Senators were not linearly separable? ('soft margin' SVM problem)

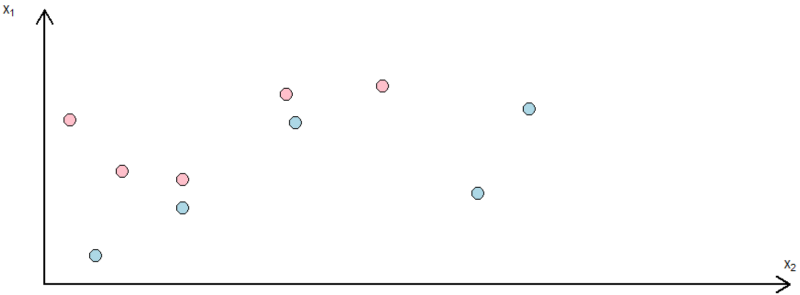
What if. . .

The Senators were not linearly separable? ('soft margin' SVM problem)

What if. . .

The Senators were not linearly separable? ('soft margin' SVM problem)

Graphically...



What if. . .

What if. . .

The Senators were not linearly separable? ('soft margin' SVM problem)

What if...

The Senators were not linearly separable? ('soft margin' SVM problem)

Can introduce a hinge loss function into the minimization problem...

$$\boxed{\mathbb{L}(f(x), y) = \max(0, 1 - f(x)y)}$$

What if...

The Senators were not linearly separable? ('soft margin' SVM problem)

Can introduce a hinge loss function into the minimization problem...

$$\boxed{\mathbb{L}(f(x), y) = \max(0, 1 - f(x)y)}$$

$= 0$ if the xs are on the 'correct' side of the margin...

What if...

The Senators were not linearly separable? ('soft margin' SVM problem)

Can introduce a hinge loss function into the minimization problem...

$$\boxed{\mathbb{L}(f(x), y) = \max(0, 1 - f(x)y)}$$

$= 0$ if the xs are on the 'correct' side of the margin...

And proportional to the distance from the margin *if* the point is on the 'wrong' side of the margin.

What if...

The Senators were not linearly separable? ('soft margin' SVM problem)

Can introduce a hinge loss function into the minimization problem...

$$\boxed{\mathbb{L}(f(x), y) = \max(0, 1 - f(x)y)}$$

$= 0$ if the xs are on the 'correct' side of the margin...

And proportional to the distance from the margin *if* the point is on the 'wrong' side of the margin.

Hyperplane(s) will be drawn in way that is more sensitive to 'bigger' mistakes in classification.

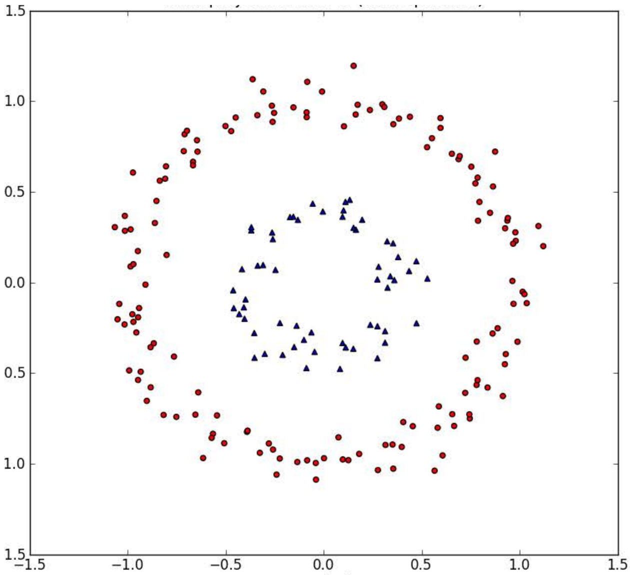
What if. . .

What if...

The situation was considerably ‘worse’,

What if...

The situation was considerably 'worse', such that neither a 'hard margin' nor 'soft margin' linear approach would work?



from `www.eric-kim.net`

What if. . .

The situation was considerably ‘worse’, such that neither a ‘hard margin’ nor ‘soft margin’ linear approach would work?

What if. . .

The situation was considerably ‘worse’, such that neither a ‘hard margin’ nor ‘soft margin’ linear approach would work?

May be a way to transform the data,

What if...

The situation was considerably 'worse', such that neither a 'hard margin' nor 'soft margin' linear approach would work?

May be a way to transform the data, using a transformation ϕ,

What if...

The situation was considerably 'worse', such that neither a 'hard margin' nor 'soft margin' linear approach would work?

May be a way to transform the data, using a transformation ϕ, such that it *can* now be separated using a linear classifier.

What if...

The situation was considerably 'worse', such that neither a 'hard margin' nor 'soft margin' linear approach would work?

May be a way to transform the data, using a transformation ϕ, such that it *can* now be separated using a linear classifier.

When the data is in a two dimensional feature space,

What if...

The situation was considerably 'worse', such that neither a 'hard margin' nor 'soft margin' linear approach would work?

May be a way to transform the data, using a transformation ϕ, such that it *can* now be separated using a linear classifier.

When the data is in a two dimensional feature space, we can lift it into a three dimensional feature space using

What if...

The situation was considerably 'worse', such that neither a 'hard margin' nor 'soft margin' linear approach would work?

May be a way to transform the data, using a transformation ϕ, such that it *can* now be separated using a linear classifier.

When the data is in a two dimensional feature space, we can lift it into a three dimensional feature space using

$$\phi(x_1, x_2) = (x_1, x_2, x_1^2 + x_2^2).$$

What if...

The situation was considerably 'worse', such that neither a 'hard margin' nor 'soft margin' linear approach would work?

May be a way to transform the data, using a transformation ϕ, such that it *can* now be separated using a linear classifier.

When the data is in a two dimensional feature space, we can lift it into a three dimensional feature space using

$$\phi(x_1, x_2) = (x_1, x_2, x_1^2 + x_2^2).$$

Then use a linear SVM on the transformed data set,

What if. . .

The situation was considerably 'worse', such that neither a 'hard margin' nor 'soft margin' linear approach would work?

May be a way to transform the data, using a transformation ϕ, such that it *can* now be separated using a linear classifier.

When the data is in a two dimensional feature space, we can lift it into a three dimensional feature space using

$$\phi(x_1, x_2) = (x_1, x_2, x_1^2 + x_2^2).$$

Then use a linear SVM on the transformed data set, and then map back to the original 2D space.

What if...

The situation was considerably 'worse', such that neither a 'hard margin' nor 'soft margin' linear approach would work?

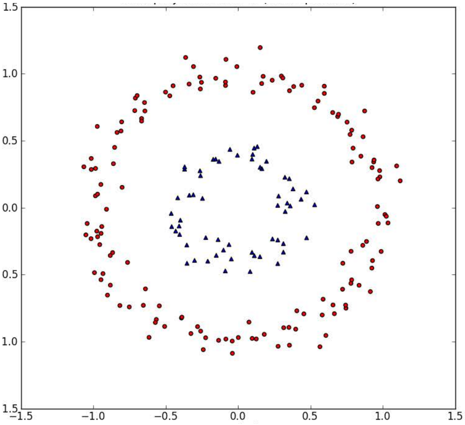
May be a way to transform the data, using a transformation ϕ, such that it *can* now be separated using a linear classifier.

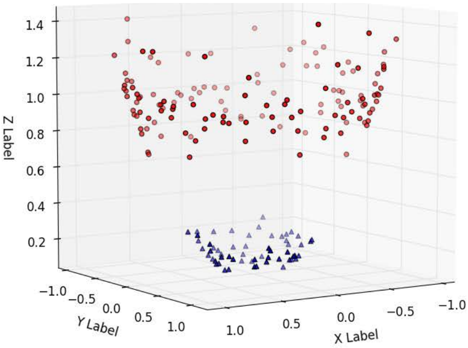
When the data is in a two dimensional feature space, we can lift it into a three dimensional feature space using

$$\phi(x_1, x_2) = (x_1, x_2, x_1^2 + x_2^2).$$

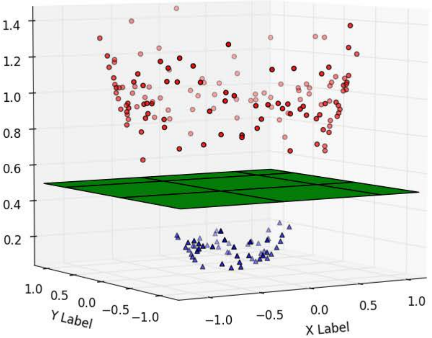
Then use a linear SVM on the transformed data set, and then map back to the original 2D space.

$\rightarrow$ Results in a non-linear hyperplane once back in 2 dimensions.

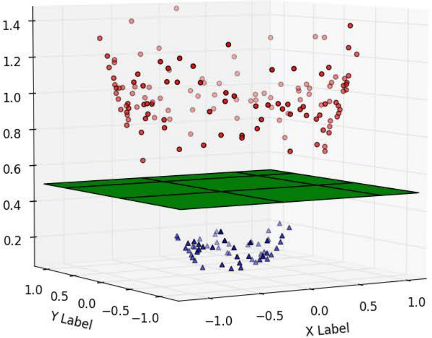


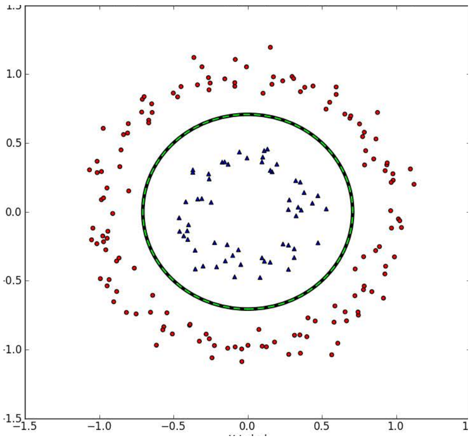


from www.eric-kim.net



1.4
1.2
1.0
0.8
0.6
0.4
0.2
1.0 0.5 0.0 −0.5 −1.0
Y Label
−1.0 −0.5 0.0 0.5 1.0
X Label





from www.eric-kim.net

Kernel Methods

Kernel Methods

Explicitly transforming data into a new space can be very expensive in terms of computation.

Kernel Methods

Explicitly transforming data into a new space can be very expensive in terms of computation.

But What if we could do the transformation, and take the distances between observations implicitly, and use them for our classification?

Kernel Methods

Explicitly transforming data into a new space can be very expensive in terms of computation.

But What if we could do the transformation, and take the distances between observations implicitly, and use them for our classification?

$\rightarrow$ exactly what kernel methods do:

Kernel Methods

Explicitly transforming data into a new space can be very expensive in terms of computation.

But What if we could do the transformation, and take the distances between observations implicitly, and use them for our classification?

$\rightarrow$ exactly what kernel methods do: use kernel functions,

Kernel Methods

Explicitly transforming data into a new space can be very expensive in terms of computation.

But What if we could do the transformation, and take the distances between observations implicitly, and use them for our classification?

$\rightarrow$ exactly what kernel methods do: use kernel functions, $K(\mathbf{x}_i, \mathbf{x}_j)$ to compute the i, j pairwise dot products for training data *as if* it had been transformed.

Kernel Methods

Explicitly transforming data into a new space can be very expensive in terms of computation.

But What if we could do the transformation, and take the distances between observations implicitly, and use them for our classification?

$\rightarrow$ exactly what kernel methods do: use kernel functions, $K(\mathbf{x}_i, \mathbf{x}_j)$ to compute the i, j pairwise dot products for training data *as if* it had been transformed. Can then feed those inner products to the classifier.

Kernel Methods

Explicitly transforming data into a new space can be very expensive in terms of computation.

But What if we could do the transformation, and take the distances between observations implicitly, and use them for our classification?

$\rightarrow$ exactly what kernel methods do: use kernel functions, $K(\mathbf{x}_i, \mathbf{x}_j)$ to compute the i, j pairwise dot products for training data *as if* it had been transformed. Can then feed those inner products to the classifier.

this 'kernel trick' cuts cost considerably,

Kernel Methods

Explicitly transforming data into a new space can be very expensive in terms of computation.

But What if we could do the transformation, and take the distances between observations implicitly, and use them for our classification?

$\rightarrow$ exactly what kernel methods do: use kernel functions, $K(\mathbf{x}_i, \mathbf{x}_j)$ to compute the i, j pairwise dot products for training data *as if* it had been transformed. Can then feed those inner products to the classifier.

this 'kernel trick' cuts cost considerably, though choosing and tuning the 'correct' kernel may be difficult.

Kernel Methods

Explicitly transforming data into a new space can be very expensive in terms of computation.

But What if we could do the transformation, and take the distances between observations implicitly, and use them for our classification?

$\rightarrow$ exactly what kernel methods do: use kernel functions, $K(\mathbf{x}_i, \mathbf{x}_j)$ to compute the i, j pairwise dot products for training data *as if* it had been transformed. Can then feed those inner products to the classifier.

this 'kernel trick' cuts cost considerably, though choosing and tuning the 'correct' kernel may be difficult.

For text analysis,

Kernel Methods

Explicitly transforming data into a new space can be very expensive in terms of computation.

But What if we could do the transformation, and take the distances between observations implicitly, and use them for our classification?

$\rightarrow$ exactly what kernel methods do: use kernel functions, $K(\mathbf{x}_i, \mathbf{x}_j)$ to compute the i, j pairwise dot products for training data *as if* it had been transformed. Can then feed those inner products to the classifier.

this 'kernel trick' cuts cost considerably, though choosing and tuning the 'correct' kernel may be difficult.

For text analysis, string kernels use a function $K(a, b)$ to implicity calculate the distance between strings of characters via the number of subsequences they have in common.