Department of Computer Science
UNIVERSITY OF COLORADO **BOULDER**

Support Vector Machines

Jordan Boyd-Graber
University of Colorado Boulder
LECTURE 8

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

Roadmap

- Classification: machines labeling data for us
- Previously: naïve Bayes and logistic regression
- This time: SVMs
  - (another) example of linear classifier
  - State-of-the-art classification
  - Good theoretical properties

Thinking Geometrically

- Suppose you have two classes: vacations and sports
- Suppose you have four documents

Sports

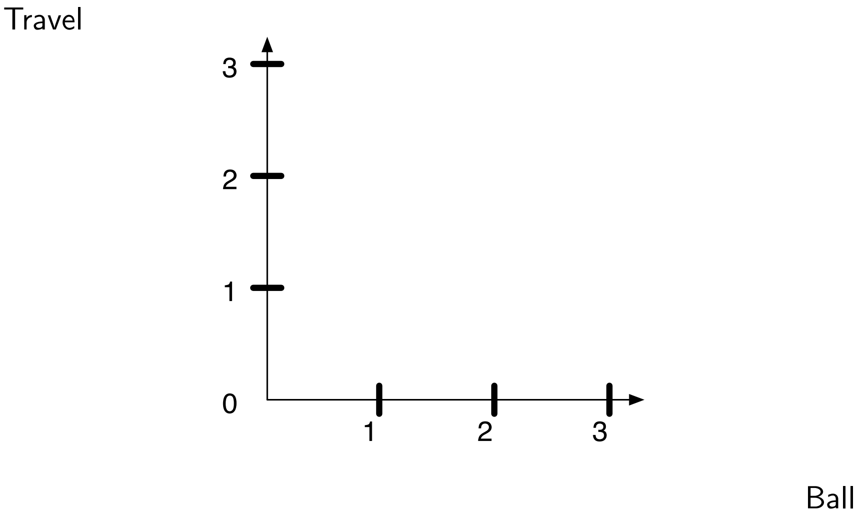
Doc_1: {ball, ball, ball, travel}
Doc_2: {ball, ball}

Vacations

Doc_3: {travel, ball, travel}
Doc_4: {travel}

- What does this look like in vector space?

Put the documents in vector space



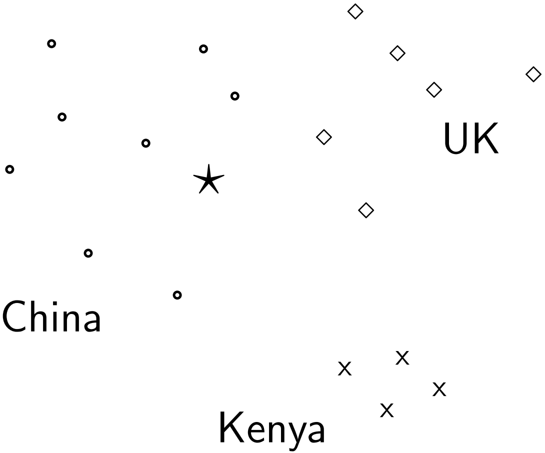
Vector space representation of documents

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

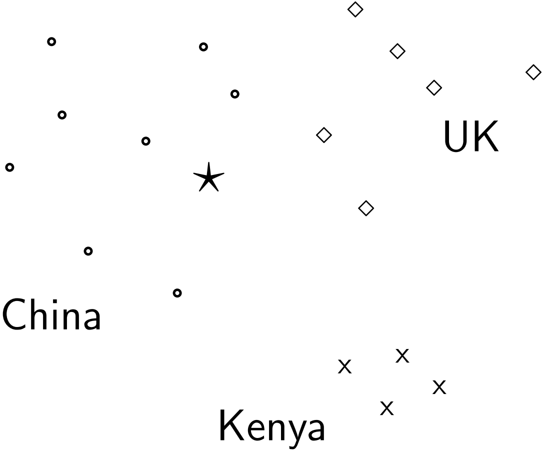
Vector space classification

- 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

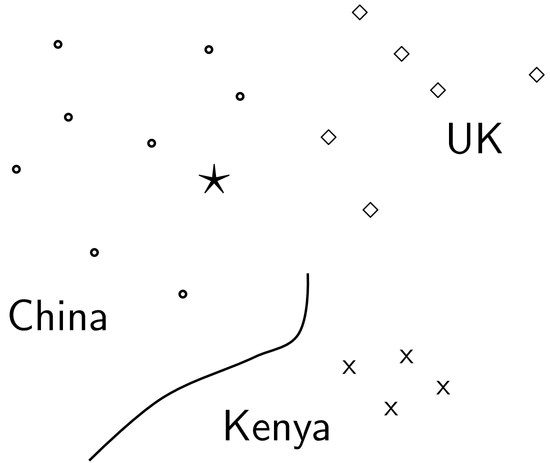


Classes in the vector space



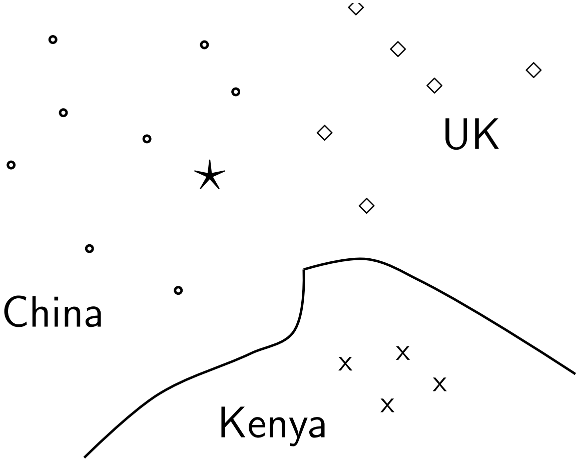
Should the document $\star$ be assigned to `China`, `UK` or `Kenya`?

Classes in the vector space



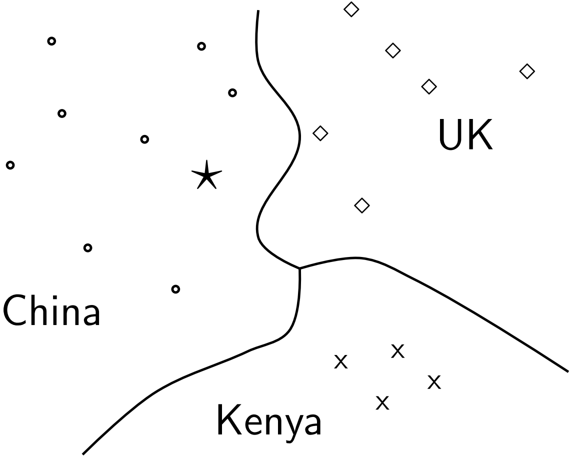
Find separators between the classes

Classes in the vector space



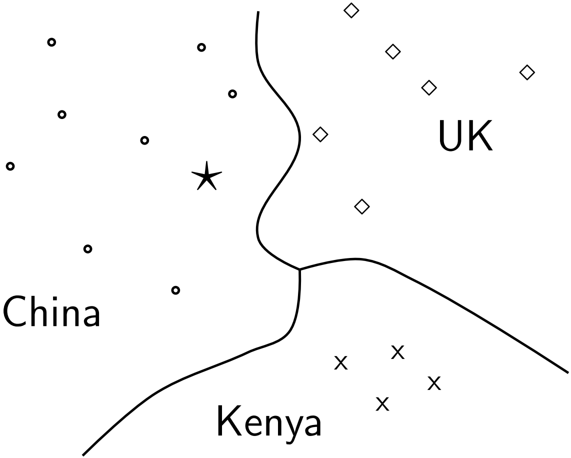
Find separators between the classes

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 $\star$? – Main topic of today

Plan

Linear Classifiers

Support Vector Machines

Formulation

Theoretical Guarantees

Recap

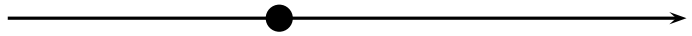
Linear classifiers

- 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: logistic regression, naïve Bayes, linear SVM
- Assumption: The classes are **linearly separable**.

Linear classifiers

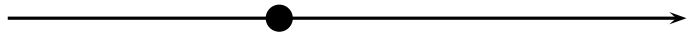
- Definition:
  - A linear classifier computes a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
  - Classification decision: $\sum_i w_i x_i > \theta$?
  - ...where θ (the threshold) is a parameter.
- (First, we only consider binary classifiers.)
- Geometrically, this corresponds 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: logistic regression, naïve Bayes, linear SVM
- Assumption: The classes are **linearly separable**.
- Before, we just talked about equations. What's the geometric intuition?

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

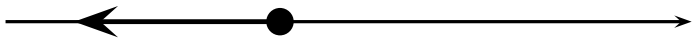


- A linear classifier in 1D is a point x described by the equation $w_1 d_1 = \theta$
- $x = \theta / w_1$

A linear classifier in 1D

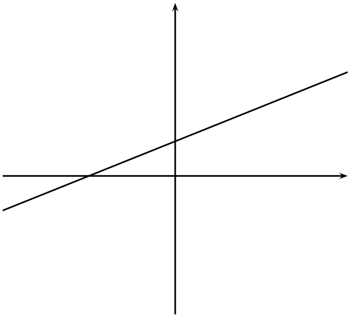
- A linear classifier in 1D is a point x described by the equation $w_1 d_1 = \theta$
- $x = \theta / w_1$
- Points (d_1) with $w_1 d_1 \geq \theta$ are in the class c.

A linear classifier in 1D



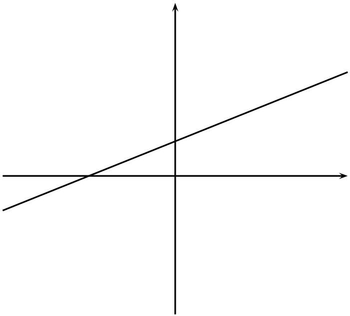
- A linear classifier in 1D is a point x described by the equation $w_1 d_1 = \theta$
- $x = \theta / w_1$
- Points (d_1) with $w_1 d_1 \geq \theta$ are in the class c.
- Points (d_1) with $w_1 d_1 < \theta$ are in the complement class $\overline{c}$.

A linear classifier in 2D



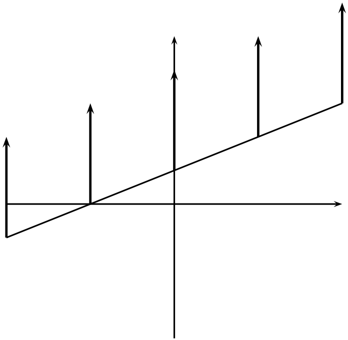
- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$

A linear classifier in 2D



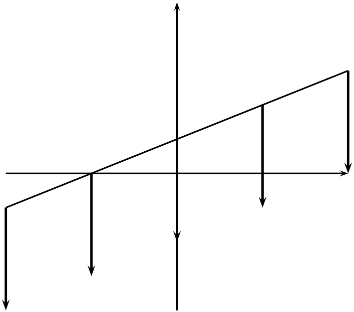
- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$
- Example for a 2D linear classifier

A linear classifier in 2D



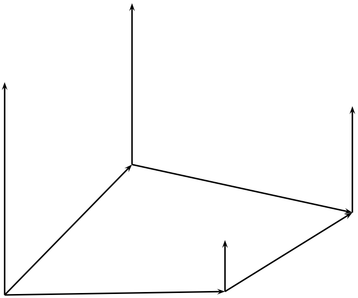
- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$
- Example for a 2D linear classifier
- Points $(d_1 \; d_2)$ with $w_1 d_1 + w_2 d_2 \geq \theta$ are in the class c.

A linear classifier in 2D



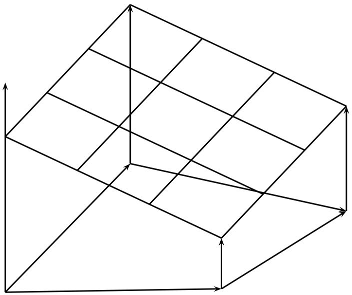
- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$
- Example for a 2D linear classifier
- Points $(d_1 \ d_2)$ with $w_1 d_1 + w_2 d_2 \geq \theta$ are in the class c.
- Points $(d_1 \ d_2)$ with $w_1 d_1 + w_2 d_2 < \theta$ are in the complement class $\overline{c}$.

A linear classifier in 3D



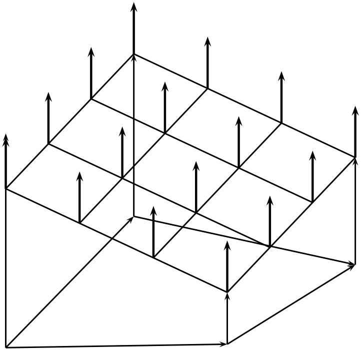
- A linear classifier in 3D is a plane described by the equation $w_1 d_1 + w_2 d_2 + w_3 d_3 = \theta$

A linear classifier in 3D



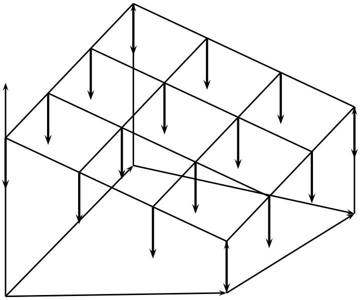
- A linear classifier in 3D is a plane described by the equation $w_1 d_1 + w_2 d_2 + w_3 d_3 = \theta$
- Example for a 3D linear classifier

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 and Logistic Regression as linear classifiers

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.

Logistic regression is the same (we only put it into the logistic function to turn it into a probability).

Naive Bayes and Logistic Regression as linear classifiers

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

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

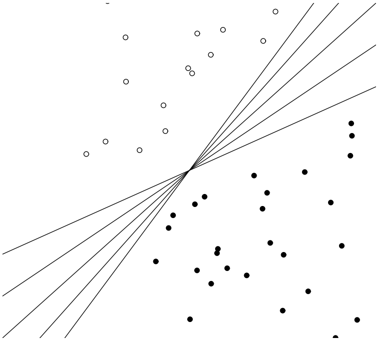
where $w_i = \log[\hat{P}(t_i|c)/\hat{P}(t_i|\bar{c})]$, $d_i =$ number of occurrences of t_i in d, and $\theta = -\log[\hat{P}(c)/\hat{P}(\bar{c})]$. Here, the index i, $1 \leq i \leq M$, refers to terms of the vocabulary.

Logistic regression is the same (we only put it into the logistic function to turn it into a probability).

Takeway

Naïve Bayes, logistic regression and SVM are all linear methods. They choose their hyperplanes based on different objectives: joint likelihood (NB), conditional likelihood (LR), and the margin (SVM).

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?

Plan

Linear Classifiers

Support Vector Machines

Formulation

Theoretical Guarantees

Recap

Support vector machines

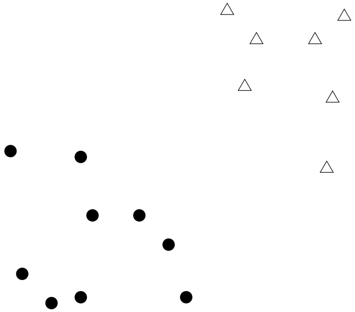
- Machine-learning research in the last two decades has improved classifier effectiveness.
- New generation of state-of-the-art classifiers: support vector machines (SVMs), boosted decision trees, regularized logistic regression, neural networks, and random forests
- Applications to IR problems, particularly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

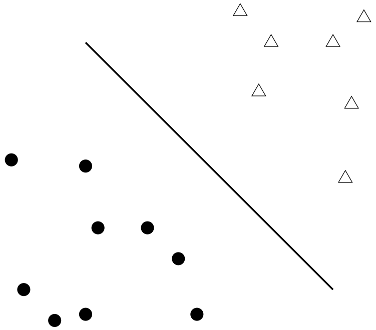
Support Vector Machines

- 2-class training data



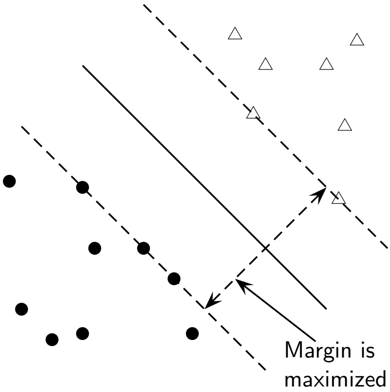
Support Vector Machines

- 2-class training data
- decision boundary $\rightarrow$ **linear separator**



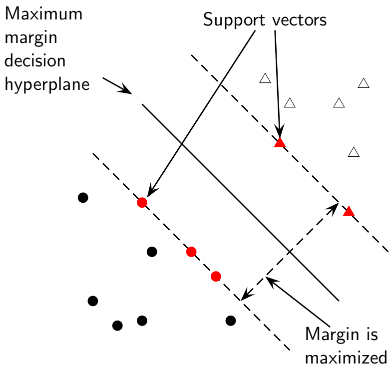
Support Vector Machines

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



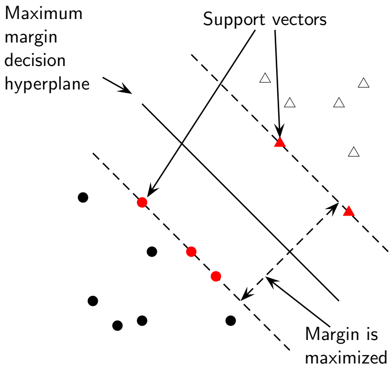
Support Vector Machines

- 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**



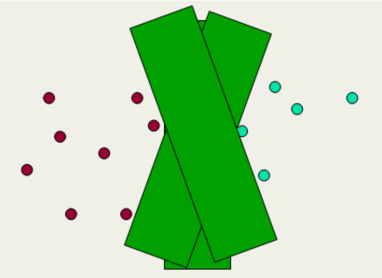
Why maximize the margin?

- Points near decision surface → uncertain classification decisions
- A classifier with a large margin is always confident
- Gives classification safety margin (measurement or 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



Plan

Linear Classifiers

Support Vector Machines

Formulation

Theoretical Guarantees

Recap

Equation

- Equation of a hyperplane

$$\vec{w} \cdot x_i + b = 0 \tag{1}$$

- Distance of a point to hyperplane

$$\frac{|\vec{w} \cdot x_i + b|}{||\vec{w}||} \tag{2}$$

- The margin ρ is given by

$$\rho \equiv \min_{(x,y)\in S} \frac{|\vec{w} \cdot x_i + b|}{||\vec{w}||} = \frac{1}{||\vec{w}||} \tag{3}$$

Equation

- Equation of a hyperplane

$$\vec{w} \cdot x_i + b = 0 \tag{1}$$

- Distance of a point to hyperplane

$$\frac{|\vec{w} \cdot x_i + b|}{||\vec{w}||} \tag{2}$$

- The margin ρ is given by

$$\rho \equiv \min_{(x,y) \in S} \frac{|\vec{w} \cdot x_i + b|}{||\vec{w}||} = \frac{1}{||\vec{w}||} \tag{3}$$

- This is because for any point on the marginal hyperplane, $\vec{w} \cdot x + b = \pm 1$

Optimization Problem

We want to find a weight vector $\vec{w}$ and bias b that optimize

$$\min_{\vec{w},b} \frac{1}{2}||w||^2 \tag{4}$$

subject to $y_i(\vec{w} \cdot x_i + b) \geq 1$, $\forall i \in [1, m]$.

Optimization Problem

We want to find a weight vector $\vec{w}$ and bias b that optimize

$$\min_{\vec{w},b} \frac{1}{2}||w||^2 \tag{4}$$

subject to $y_i(\vec{w} \cdot x_i + b) \geq 1$, $\forall i \in [1, m]$.

Next week: algorithm

Plan

Linear Classifiers

Support Vector Machines

Formulation

Theoretical Guarantees

Recap

Three Proofs that Suggest SVMs will Work

- Leave-one-out error
- VC Dimension
- Margin analysis

Leave One Out Error (sketch)

Leave one out error is the error by using one point as your test set (averaged over all such points).

$$\hat{R}_{LOO} = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\left[h_{s-\{x_i\}} \neq y_i\right] \tag{5}$$

Leave One Out Error (sketch)

Leave one out error is the error by using one point as your test set (averaged over all such points).

$$\hat{R}_{LOO} = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\left[h_{s-\{x_i\}} \neq y_i\right] \tag{5}$$

This serves as an unbiased estimate of generalization error for samples of size $m - 1$:

$$\mathbb{E}_{S \sim D^m}\left[\hat{R}_{LOO}\right] = \mathbb{E}_{S' \sim D^{m-1}}\left[R(h_{S'})\right] \tag{6}$$

Leave One Out Error (sketch)

Let h_S be the hypothesis returned by SVMs for a separable sample S, and let $N_{SV}(S)$ be the number of support vectors that define h_S.

$$\mathbb{E}_{S \sim D^m}\left[R(h_s)\right] \leq \mathbb{E}_{S \sim D^{m+1}}\left[\frac{N_{SV}(S)}{m+1}\right] \tag{7}$$

Consider the held out error for x_i.

- If x_i was not a support vector, the answer doesn't change.
- If x_i was a support vector, it could change the answer; this is when we can have an error.

There are $N_{SV}(S)$ support vectors and thus $N_{SV}(S)$ possible errors.

VC Dimension Argument

Remember discussion VC dimension for d-dimensional hyperplanes? That applies here:

$$R(h) \leq \hat{R}(h) + \sqrt{\frac{2(d+1)\log\frac{\epsilon}{d+1}}{m}} + \sqrt{\frac{\log\frac{1}{\delta}}{2m}} \tag{8}$$

VC Dimension Argument

Remember discussion VC dimension for d-dimensional hyperplanes? That applies here:

$$R(h) \leq \hat{R}(h) + \sqrt{\frac{2(d+1)\log\frac{\epsilon}{d+1}}{m}} + \sqrt{\frac{\log\frac{1}{\delta}}{2m}} \tag{8}$$

But this is useless when d is large (e.g. for text).

Margin Theory

Margin Loss Function

To see where SVMs really shine, consider the margin loss ρ:

$$\Phi_\rho(x) = \begin{cases} 0 & \text{if } \rho \leq x \\ 1 - \frac{x}{\rho} & \text{if } 0 \leq x \leq \rho \\ 1 & \text{if } x \leq 0 \end{cases} \tag{9}$$

Margin Loss Function

To see where SVMs really shine, consider the margin loss ρ:

$$\Phi_\rho(x) = \begin{cases} 0 & \text{if } \rho \leq x \\ 1 - \frac{x}{\rho} & \text{if } 0 \leq x \leq \rho \\ 1 & \text{if } x \leq 0 \end{cases} \tag{9}$$

The empirical margin loss of a hypothesis h is

$$\hat{R}_\rho(h) = \frac{1}{m} \sum_{i=1}^{m} \Phi_\rho(y_i h(x_i)) \tag{10}$$

Margin Loss Function

To see where SVMs really shine, consider the margin loss ρ:

$$\Phi_\rho(x) = \begin{cases} 0 & \text{if } \rho \leq x \\ 1 - \frac{x}{\rho} & \text{if } 0 \leq x \leq \rho \\ 1 & \text{if } x \leq 0 \end{cases} \tag{9}$$

The empirical margin loss of a hypothesis h is

$$\hat{R}_\rho(h) = \frac{1}{m} \sum_{i=1}^{m} \Phi_\rho(y_i h(x_i)) \leq \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\left[y_i h(x_i) \leq \rho\right] \tag{10}$$

Margin Loss Function

To see where SVMs really shine, consider the margin loss ρ:

$$\Phi_\rho(x) = \begin{cases} 0 & \text{if } \rho \leq x \\ 1 - \frac{x}{\rho} & \text{if } 0 \leq x \leq \rho \\ 1 & \text{if } x \leq 0 \end{cases} \tag{9}$$

The empirical margin loss of a hypothesis h is

$$\hat{R}_\rho(h) = \frac{1}{m} \sum_{i=1}^{m} \Phi_\rho(y_i h(x_i)) \leq \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\left[y_i h(x_i) \leq \rho\right] \tag{10}$$

The fraction of the points in the training sample S that have been misclassified or classified with confidence less than ρ.

Generalization

For linear classifiers $H = \{x \mapsto w \cdot x : ||w|| \leq \Lambda\}$ and data $X \in \{x : ||x|| \leq r\}$. Fix $\rho > 0$ then with probability at least $1 - \delta$, for any $h \in H$,

$$R(h) \leq \hat{R}_\rho(h) + 2\sqrt{\frac{r^2\Lambda^2}{\rho^2 m}} + \sqrt{\frac{\log \frac{1}{\delta}}{2m}} \tag{11}$$

Generalization

For linear classifiers $H = \{x \mapsto w \cdot x : ||w|| \leq \Lambda\}$ and data $X \in \{x : ||x|| \leq r\}$. Fix $\rho > 0$ then with probability at least $1 - \delta$, for any $h \in H$,

$$R(h) \leq \hat{R}_\rho(h) + 2\sqrt{\frac{r^2\Lambda^2}{\rho^2 m}} + \sqrt{\frac{\log \frac{1}{\delta}}{2m}} \tag{11}$$

- Data-dependent: must be separable with a margin
- Fortunately, many data do have good margin properties
- SVMs can find good classifiers in those instances

Plan

Linear Classifiers

Support Vector Machines

Formulation

Theoretical Guarantees

Recap

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**

SVM extensions: What's next

- Finding solutions
- Slack variables: not perfect line
- Kernels: different geometries