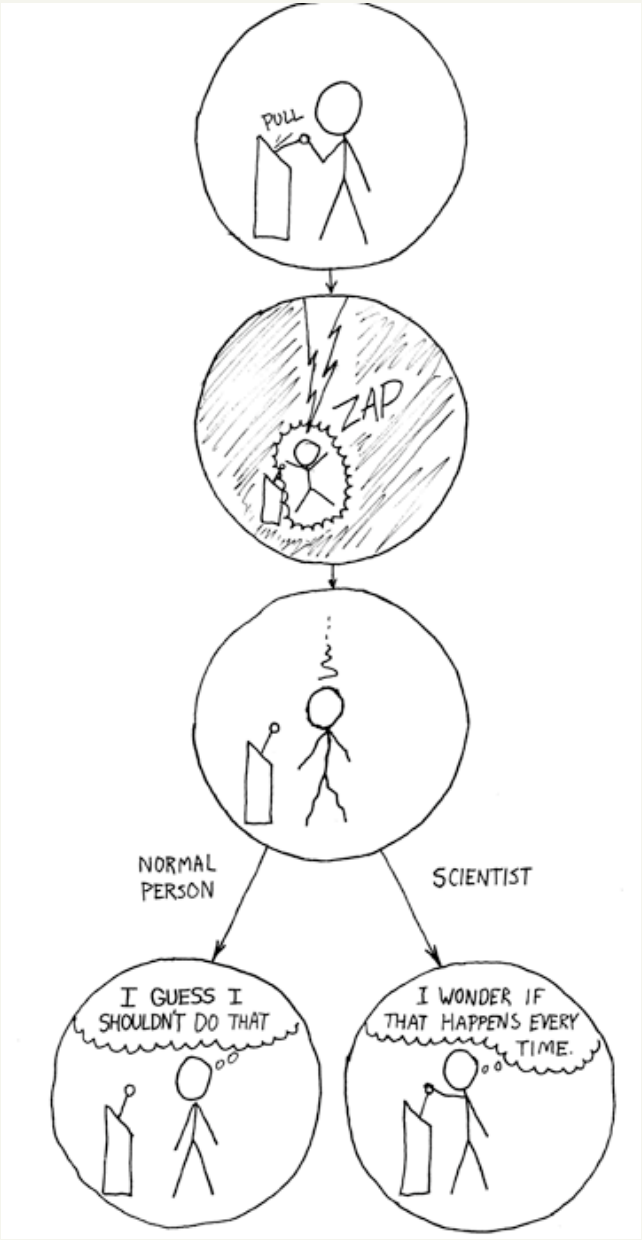


http://xkcd.com/242/

Text Classification 2

David Kauchak

cs160

Fall 2009

adapted from:

http://www.stanford.edu/class/cs276/handouts/lecture10-textcat-naivebayes.ppt

http://www.stanford.edu/class/cs276/handouts/lecture11-vector-classify.ppt

http://www.stanford.edu/class/cs276/handouts/lecture12-SVMs.ppt

Administrative

- Image teams/GUI teams, let's setup a meeting time
- Project deadlines
- hw6 out

Bias/Variance

- Bias: How well does the model predict the training data?
  - high bias – the model doesn't do a good job of predicting the training data (high training set error)
  - The model predictions are biased by the model
- Variance: How sensitive to the training data is the learned model?
  - high variance – changing the training data can drastically change the learned model

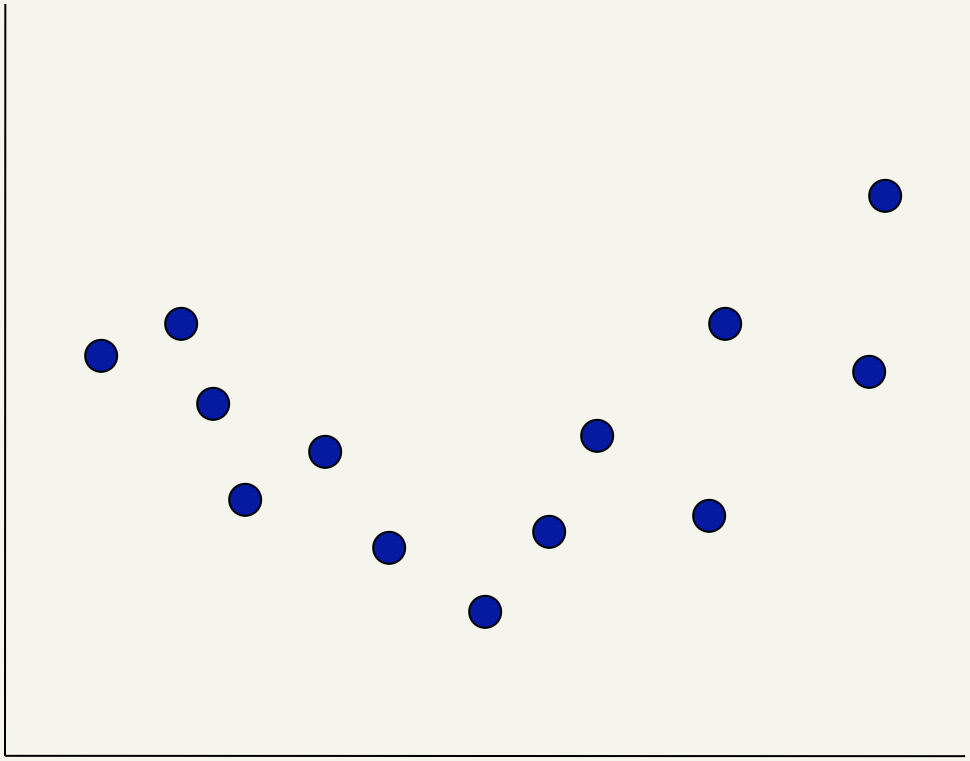
Bias/Variance

- Another way to think about it is model complexity

- Simple models
  - may not model data well
  - high bias
- Complicated models
  - may overfit to the training data
  - high variance

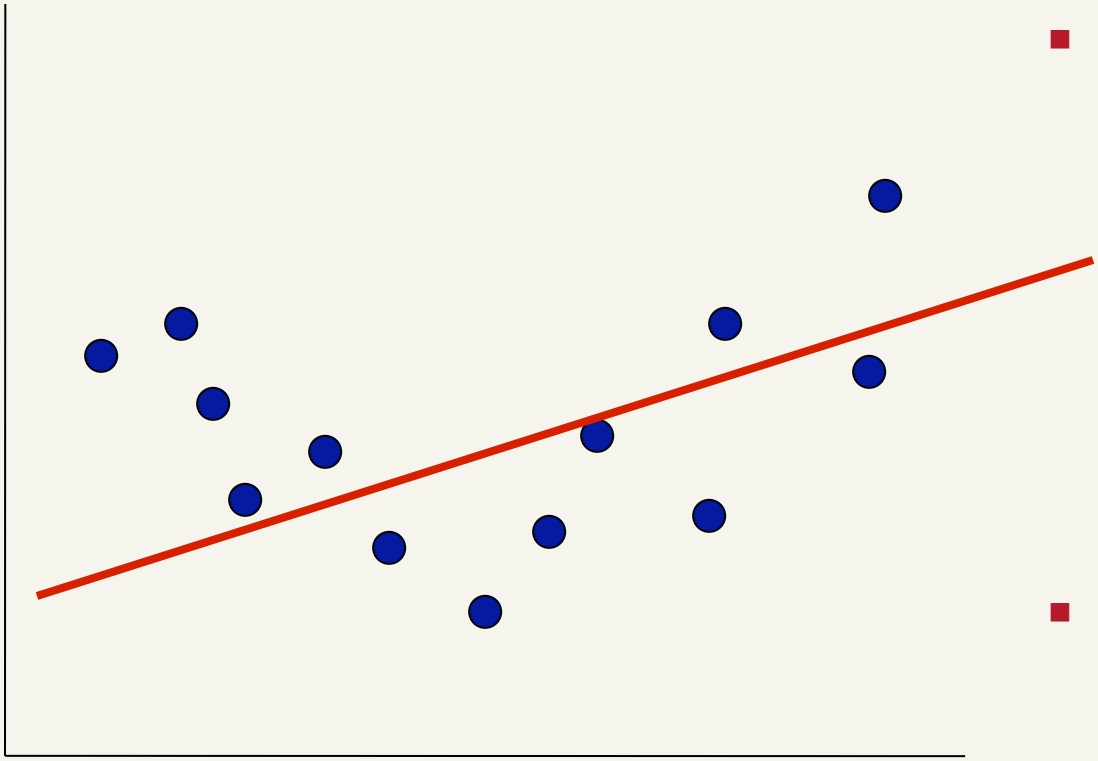
- Why do we care about bias/variance?

Bias/variance trade-off



We want to fit a polynomial to this, which one should we use?

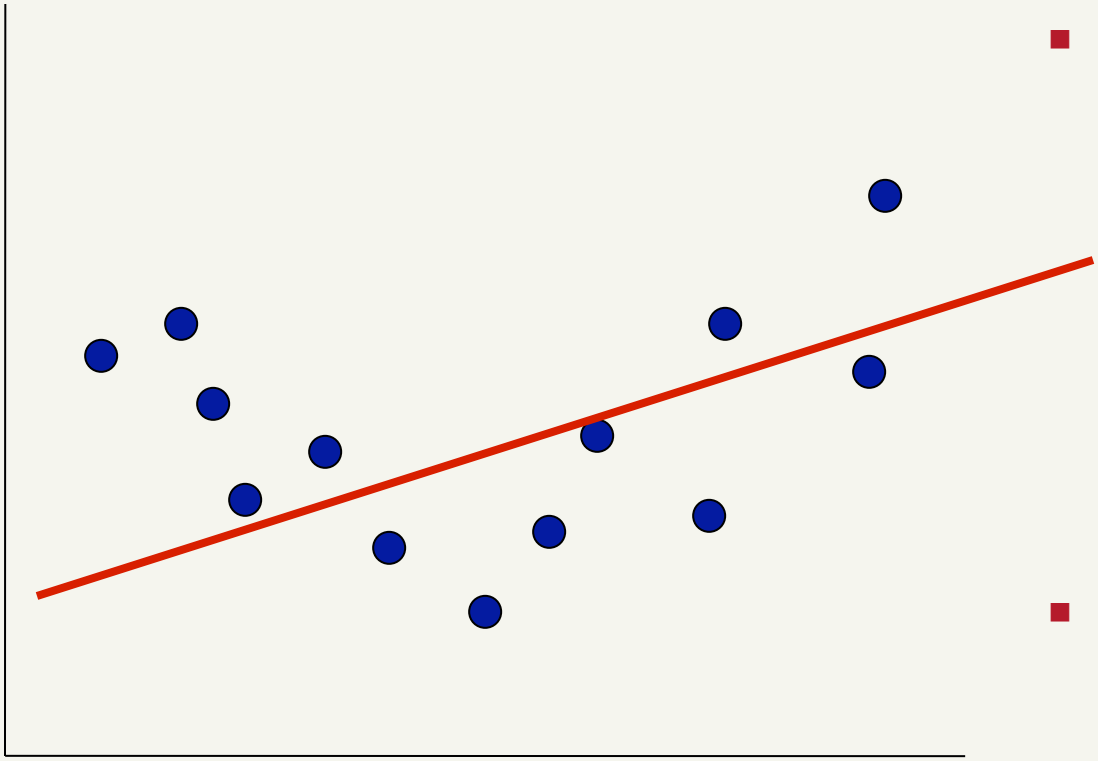
Bias/variance trade-off



High variance OR high bias?

- Bias: How well does the model predict the training data?
  - high bias – the model doesn't do a good job of predicting the training data (high training set error)
  - The model predictions are biased by the model
- Variance: How sensitive to the training data is the learned model?
  - high variance – changing the training data can drastically change the learned model

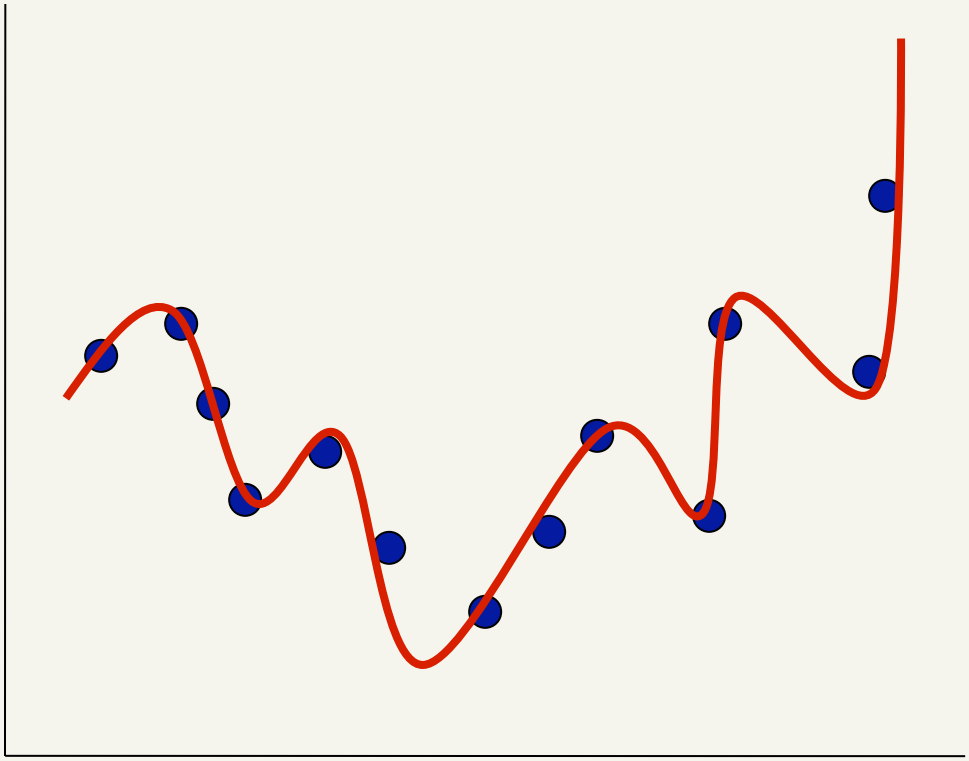
Bias/variance trade-off



High bias

- Bias: How well does the model predict the training data?
  - high bias – the model doesn't do a good job of predicting the training data (high training set error)
  - The model predictions are biased by the model
- Variance: How sensitive to the training data is the learned model?
  - high variance – changing the training data can drastically change the learned model

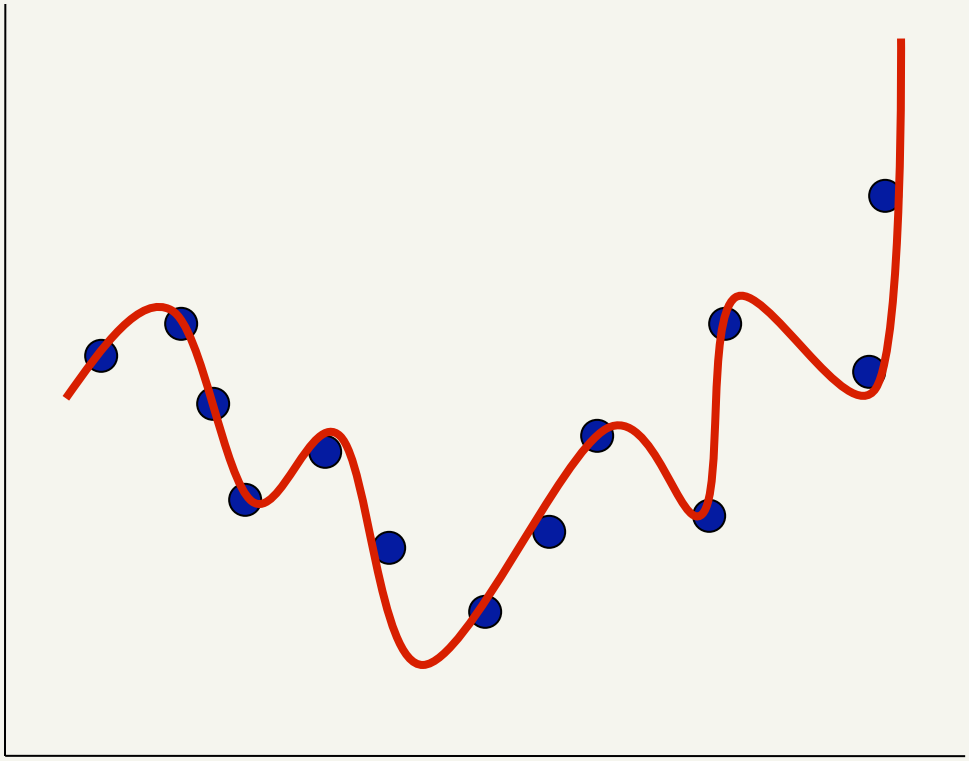
Bias/variance trade-off



High variance OR high bias?

- Bias: How well does the model predict the training data?
  - high bias – the model doesn't do a good job of predicting the training data (high training set error)
  - The model predictions are biased by the model
- Variance: How sensitive to the training data is the learned model?
  - high variance – changing the training data can drastically change the learned model

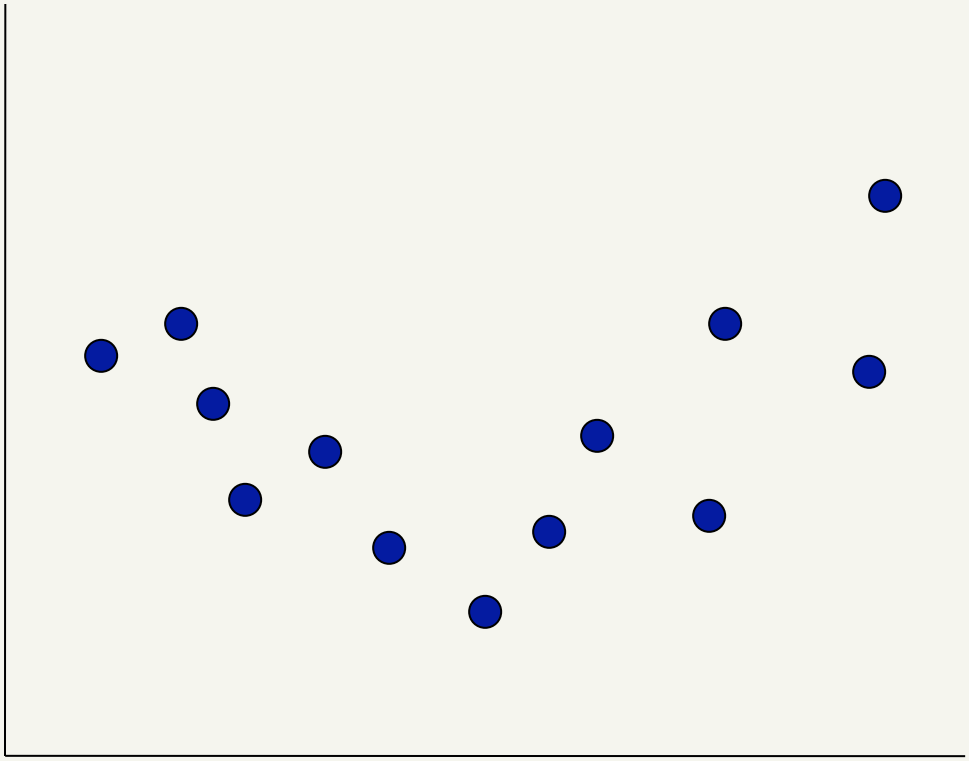
Bias/variance trade-off



High variance

- Bias: How well does the model predict the training data?
  - high bias – the model doesn't do a good job of predicting the training data (high training set error)
  - The model predictions are biased by the model
- Variance: How sensitive to the training data is the learned model?
  - high variance – changing the training data can drastically change the learned model

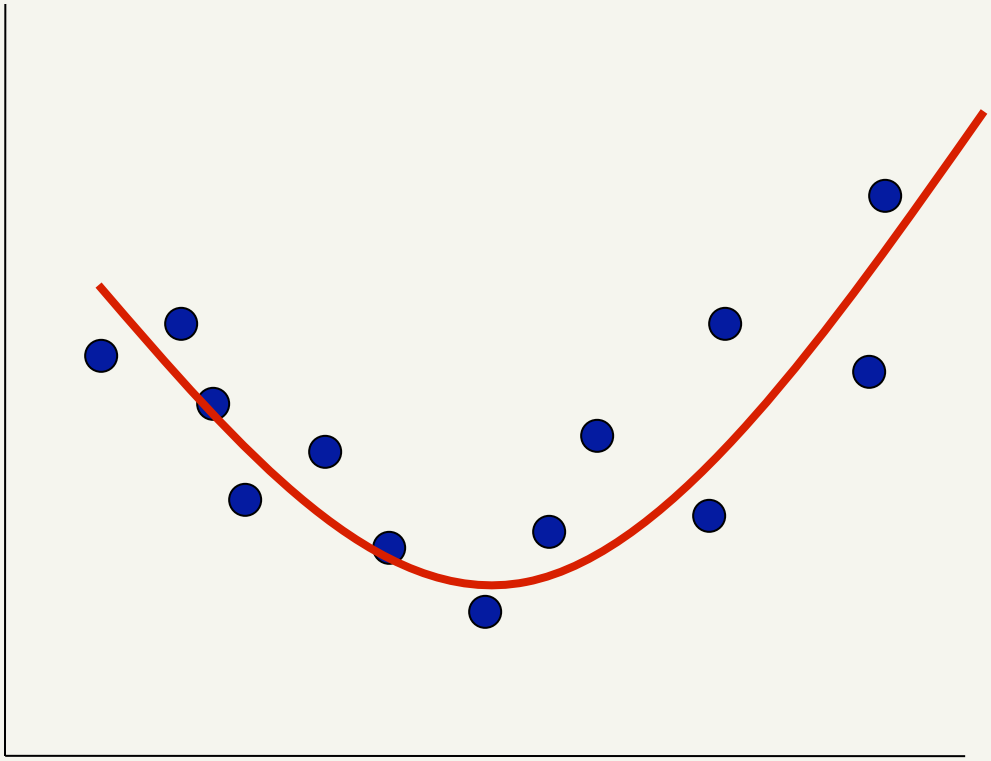
Bias/variance trade-off



What do we want?

- Bias: How well does the model predict the training data?
  - high bias – the model doesn't do a good job of predicting the training data (high training set error)
  - The model predictions are biased by the model
- Variance: How sensitive to the training data is the learned model?
  - high variance – changing the training data can drastically change the learned model

Bias/variance trade-off



Compromise between bias and variance

- Bias: How well does the model predict the training data?
  - high bias – the model doesn't do a good job of predicting the training data (high training set error)
  - The model predictions are biased by the model
- Variance: How sensitive to the training data is the learned model?
  - high variance – changing the training data can drastically change the learned model

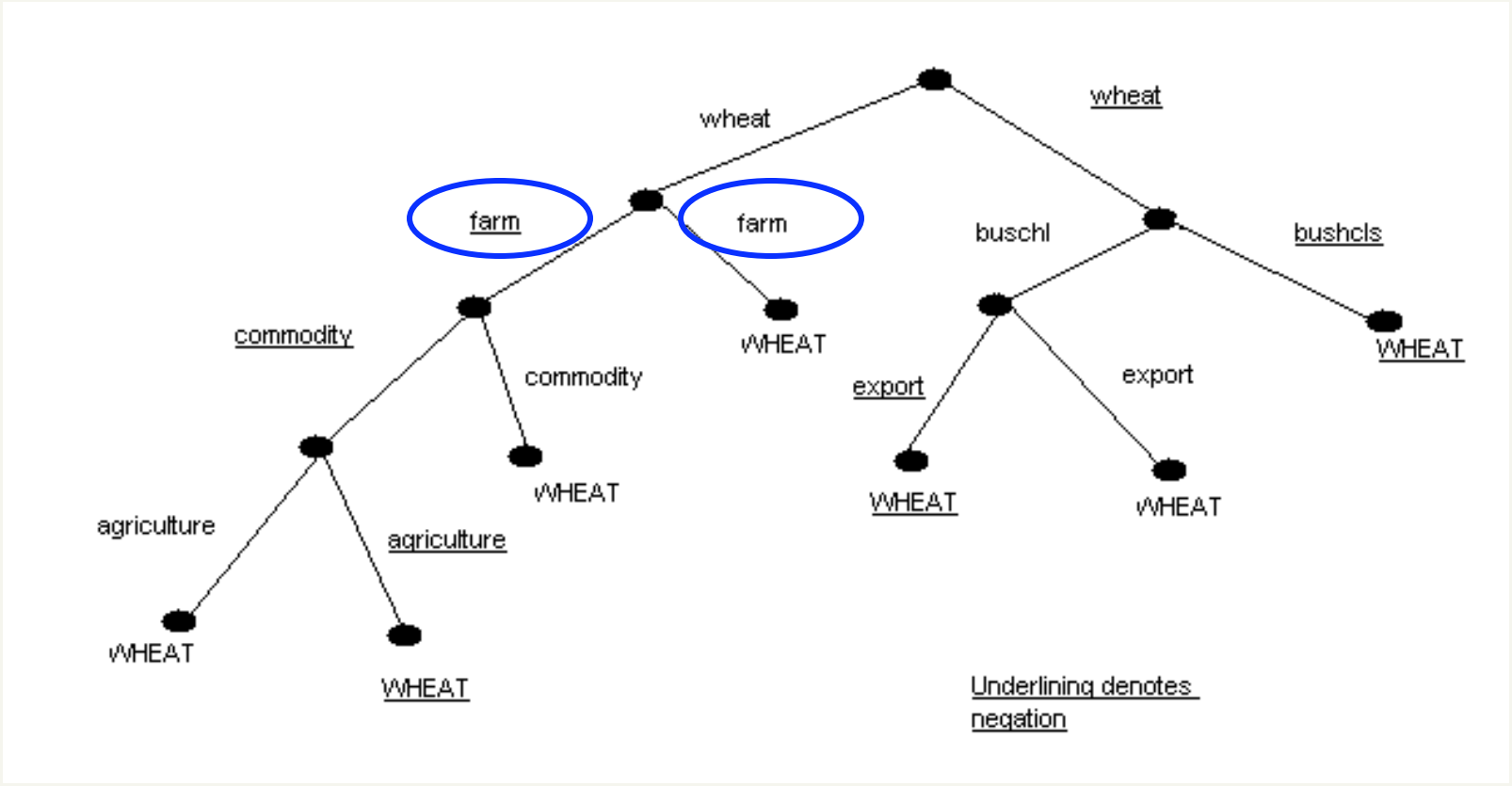
k-NN vs. Naive Bayes

How do k-NN and NB sit on the variance/bias plane?

- k-NN has high variance and low bias.
  - more complicated model
  - can model any boundary
  - but very dependent on the training data
- NB has low variance and high bias.
  - Decision surface has to be linear
  - Cannot model all data
  - but, less variation based on the training data

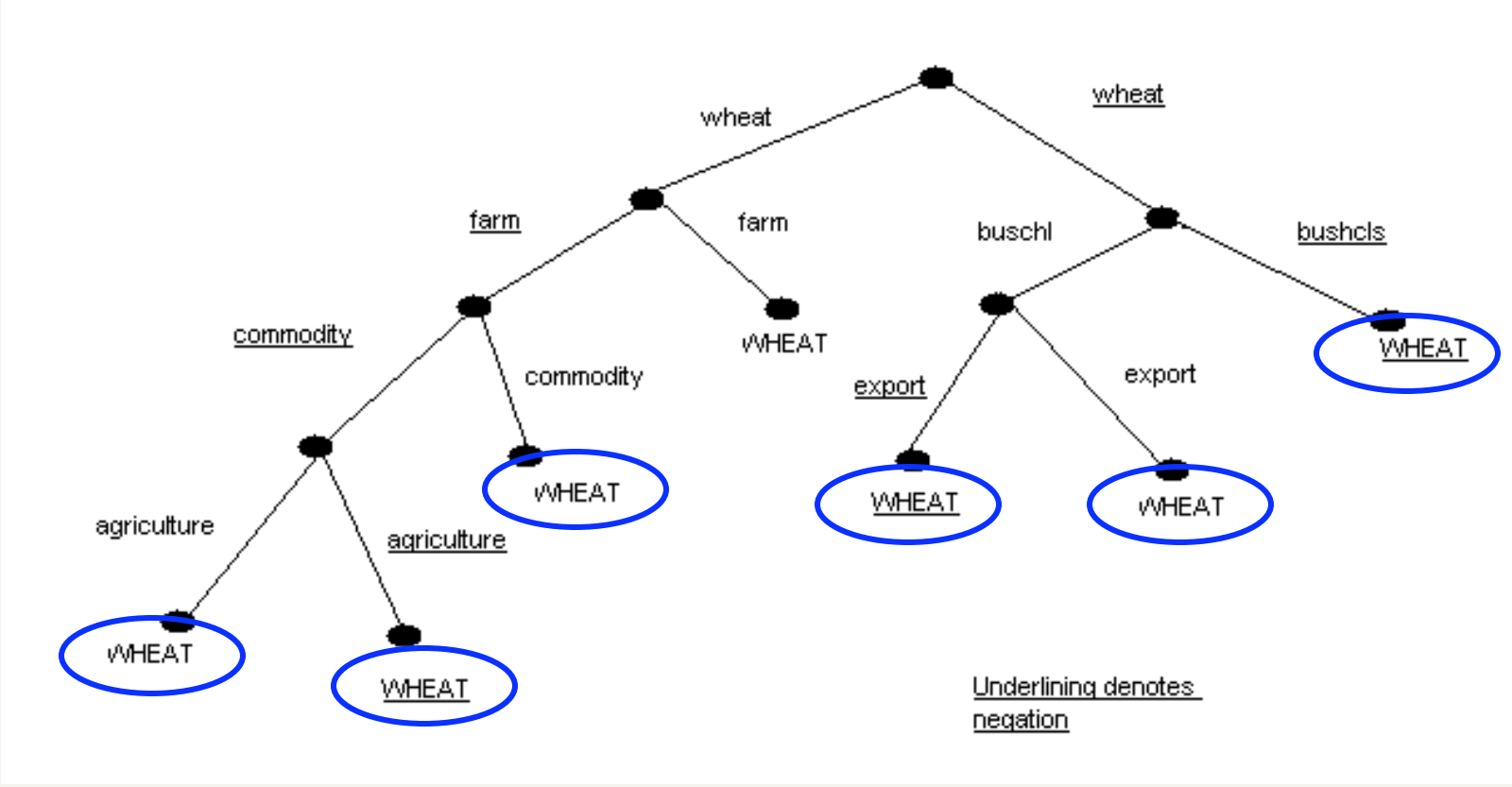
Decision trees

- Tree with internal nodes labeled by terms/features
- Branches are labeled by tests on the weight that the term has
  - farm vs. not farm
  - x > 100



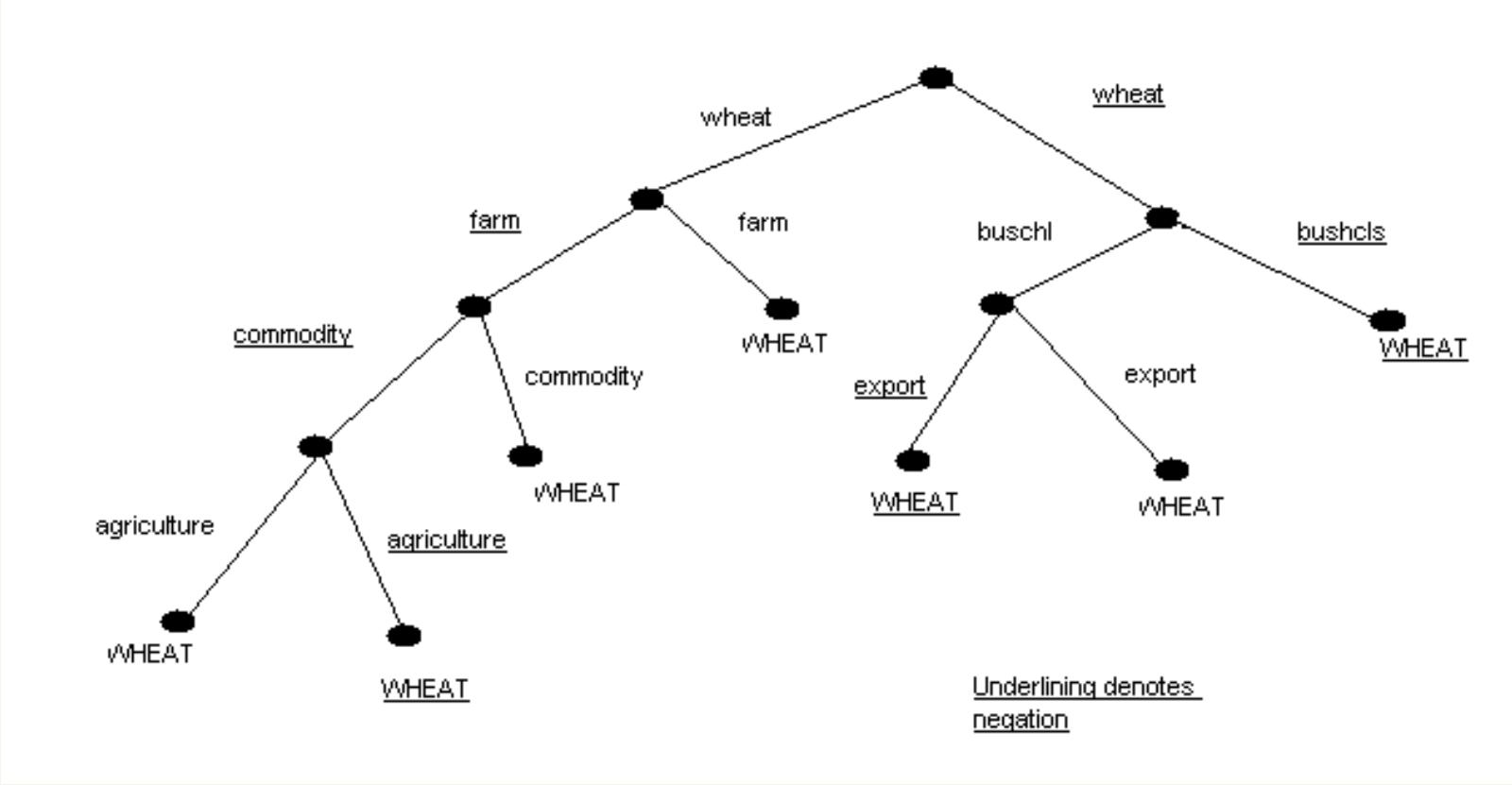
Decision trees

- Roots are labeled with the class



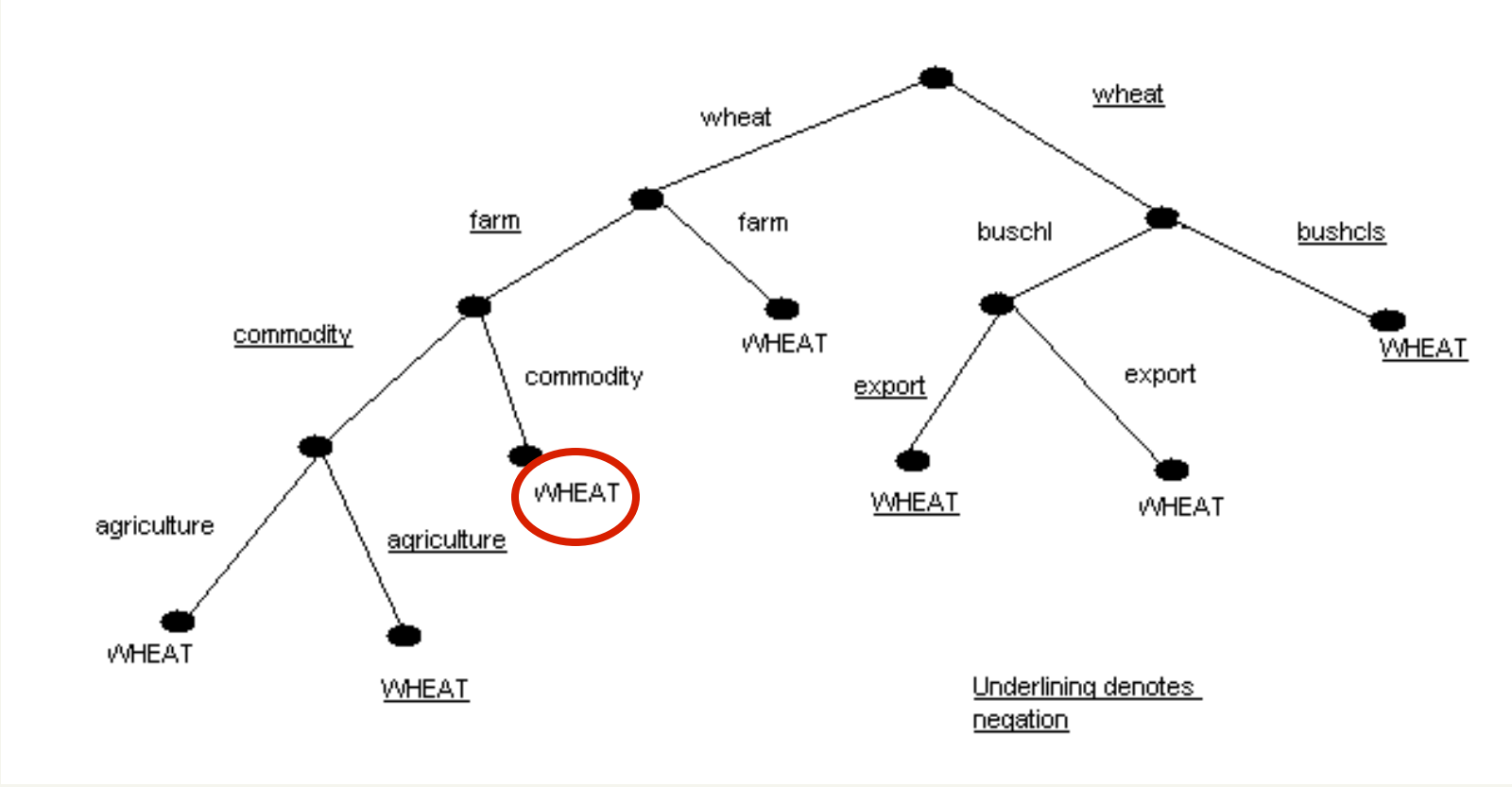
Decision trees

- Classifier categorizes a document by descending tree following tests to leaf
- The label of the leaf node is then assigned to the document



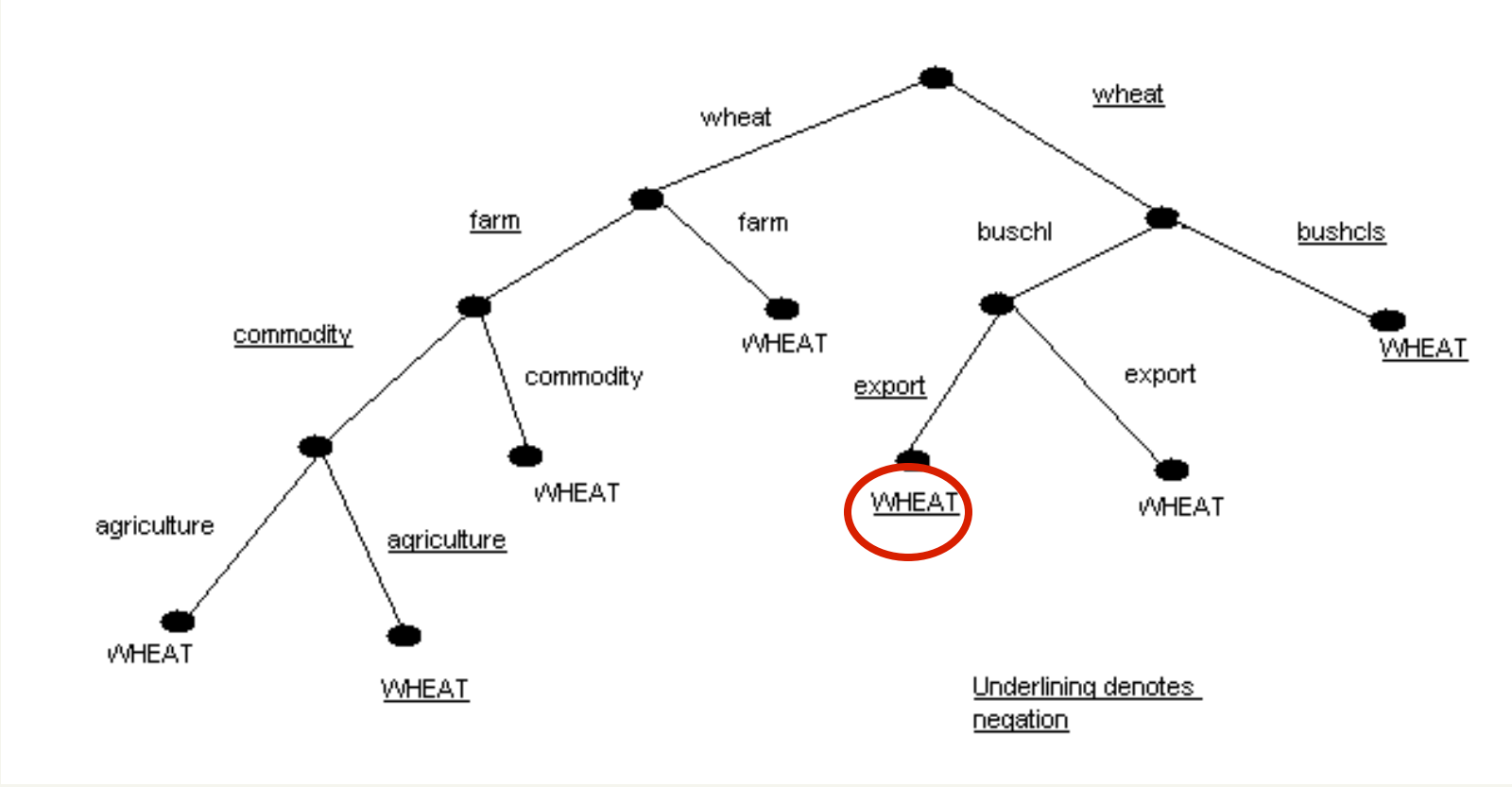
Decision trees

wheat, not(farm), commodity, not(agriculture)?



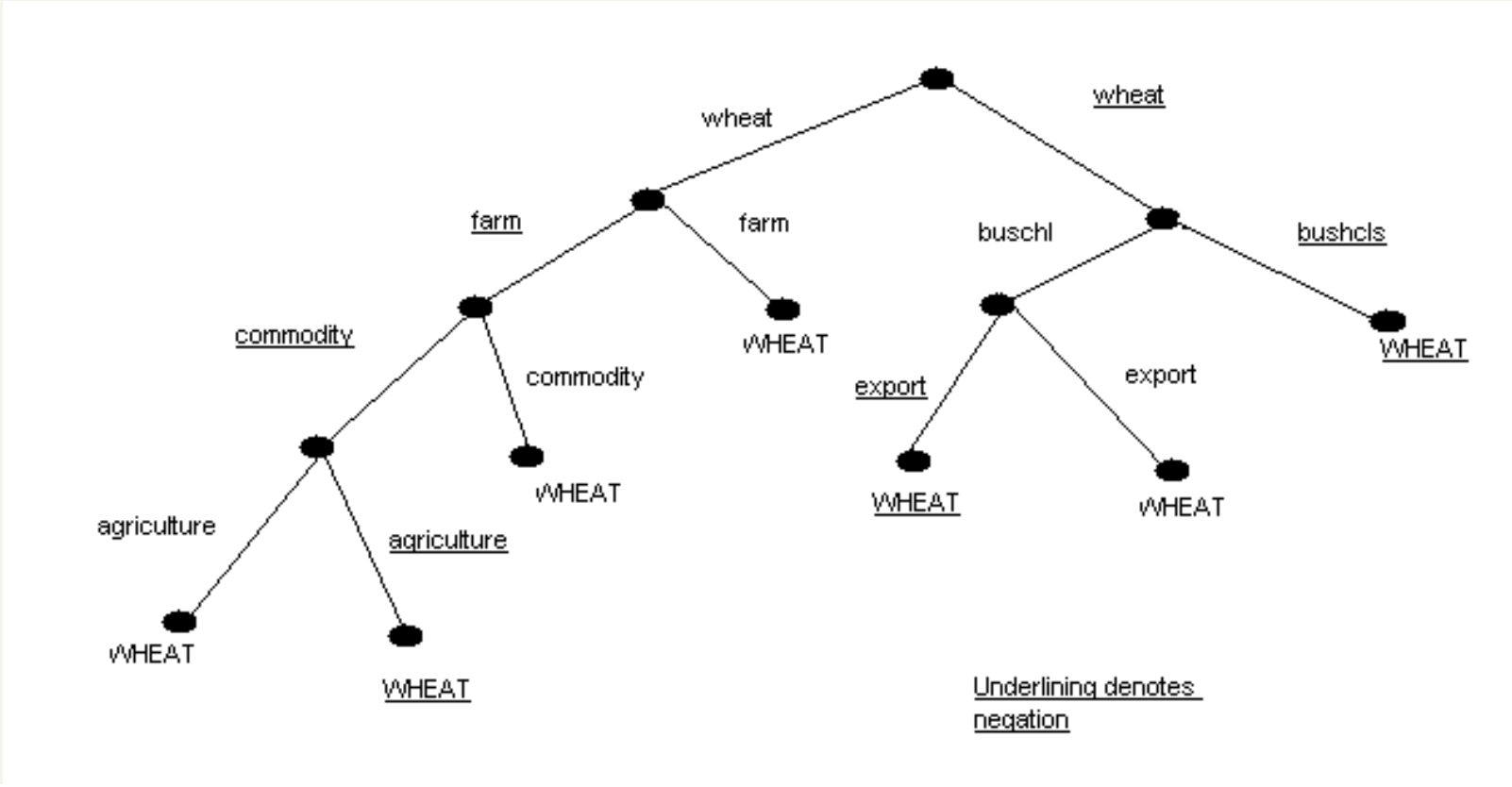
Decision trees

not(wheat), not(farm), commodity, export, buschi?



Decision trees

- Most decision trees are binary trees
- DT make good use of a few high-leverage features
- Linear or non-linear classifier?



Decision Tree Learning

- Learn a sequence of tests on features, typically using top-down, greedy search
  - Choose the unused feature with highest Information Gain/mutual information with the class

$$I(C,F) = \sum_{c \in C} \sum_{f \in F} p(c,f) \log\left(\frac{p(c,f)}{p(c)p(f)} \right)$$

When will this be large?

Decision Tree Learning

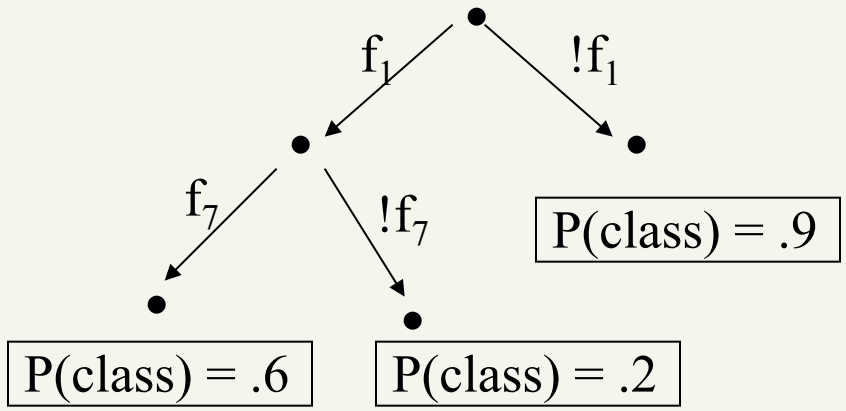
Measure of how much information two variables share

if p(c,f) is high when both p(c) and p(f) are high and vice versa, then high mutual information

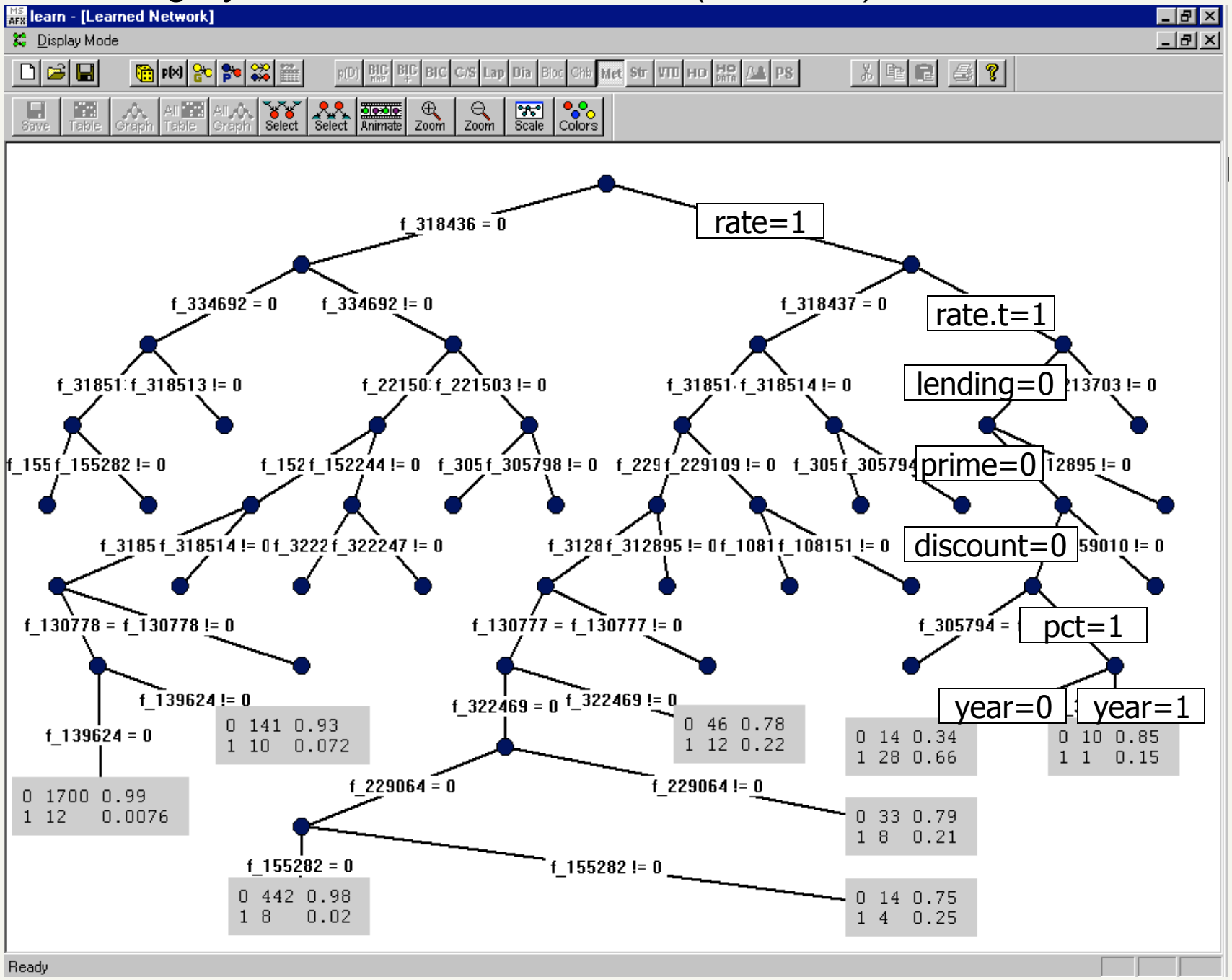
$$I(C,F) = \sum_{c \in C} \sum_{f \in F} p(c,f) \log\left(\frac{p(c,f)}{p(c)\,p(f)} \right)$$

Decision Tree Learning

- Pick one feature at each step and split the tree
- Eventually, stop splitting and calculate the probability for each class based on the training examples that satisfy the chain of constraints
- Key challenge is when to stop splitting



Category: “interest” – Dumais et al. (Microsoft) Decision Tree

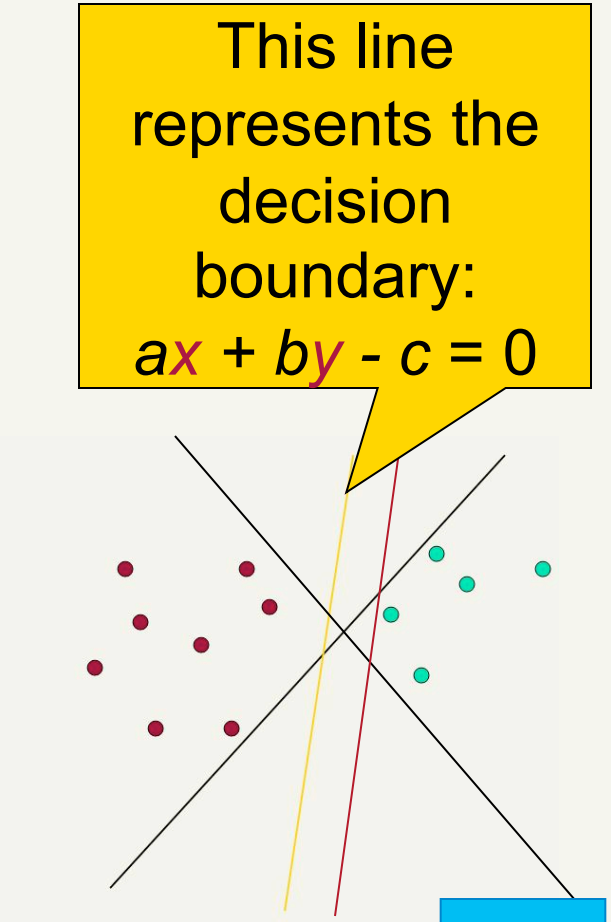


Decision trees: The good and the bad

- Good
  - Easy to understand/interpret (set of rules/ decisions)
  - Reasonable performance
  - Non-linear decision boundary
- Bad
  - Pruning: when do we stop splitting (overfitting)
  - Problems with large numbers of features/sparse data
  - Doesn’t handle data with complex feature interactions well
  - Not generally the best performing methods

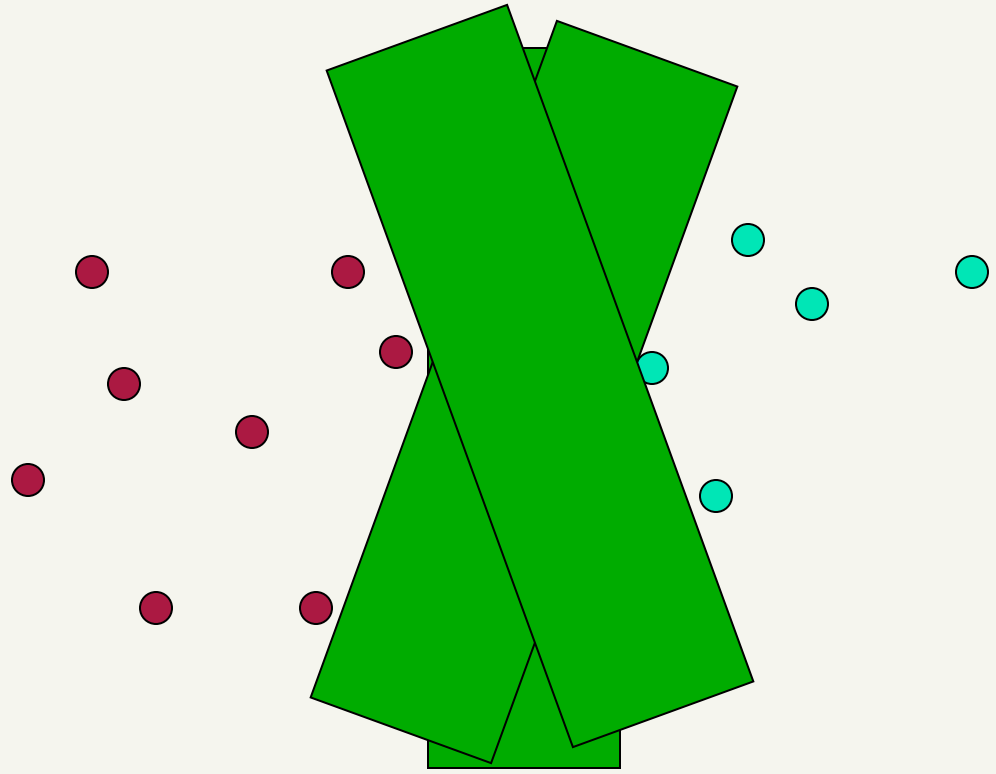
Linear classifiers: Which Hyperplane?

- Lots of possible solutions for *a,b,c.*
- Support Vector Machine (SVM) finds an optimal solution
  - Maximizes the distance between the hyperplane and the "difficult points" close to decision boundary



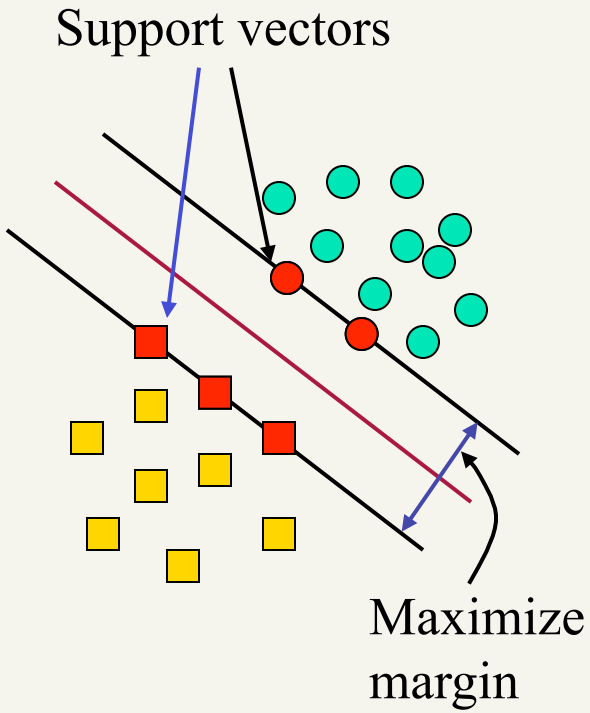
Another intuition

- If you have to place a fat separator between classes, you have less choices, and so the capacity of the model has been decreased



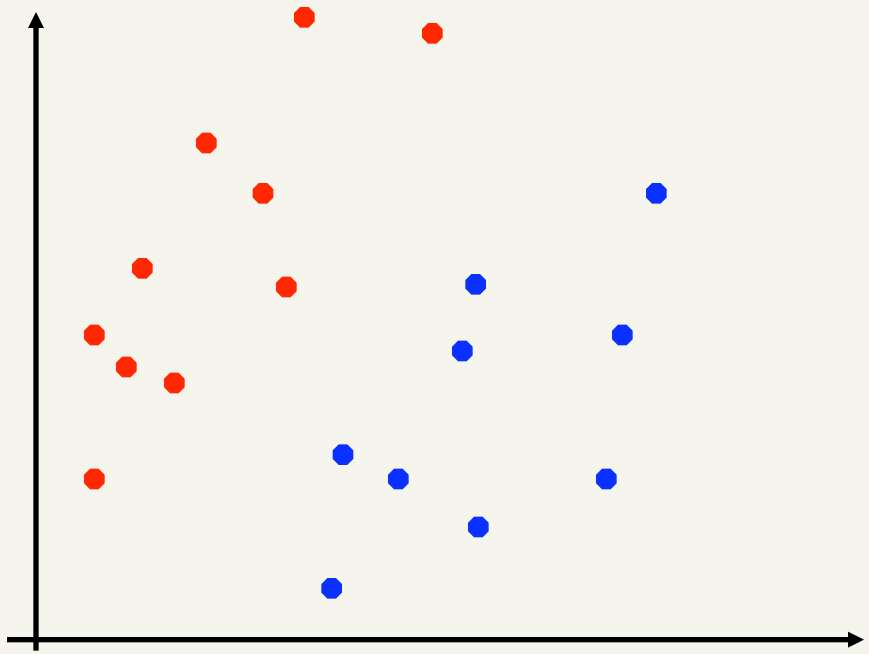
Support Vector Machine (SVM)

- SVMs maximize the *margin* around the separating hyperplane.
  - A.k.a. large margin classifiers
- The decision function is fully specified by a subset of training samples, *the support vectors*.
- Solving SVMs is a *quadratic programming* problem
- Seen by many as the most successful current text classification method*

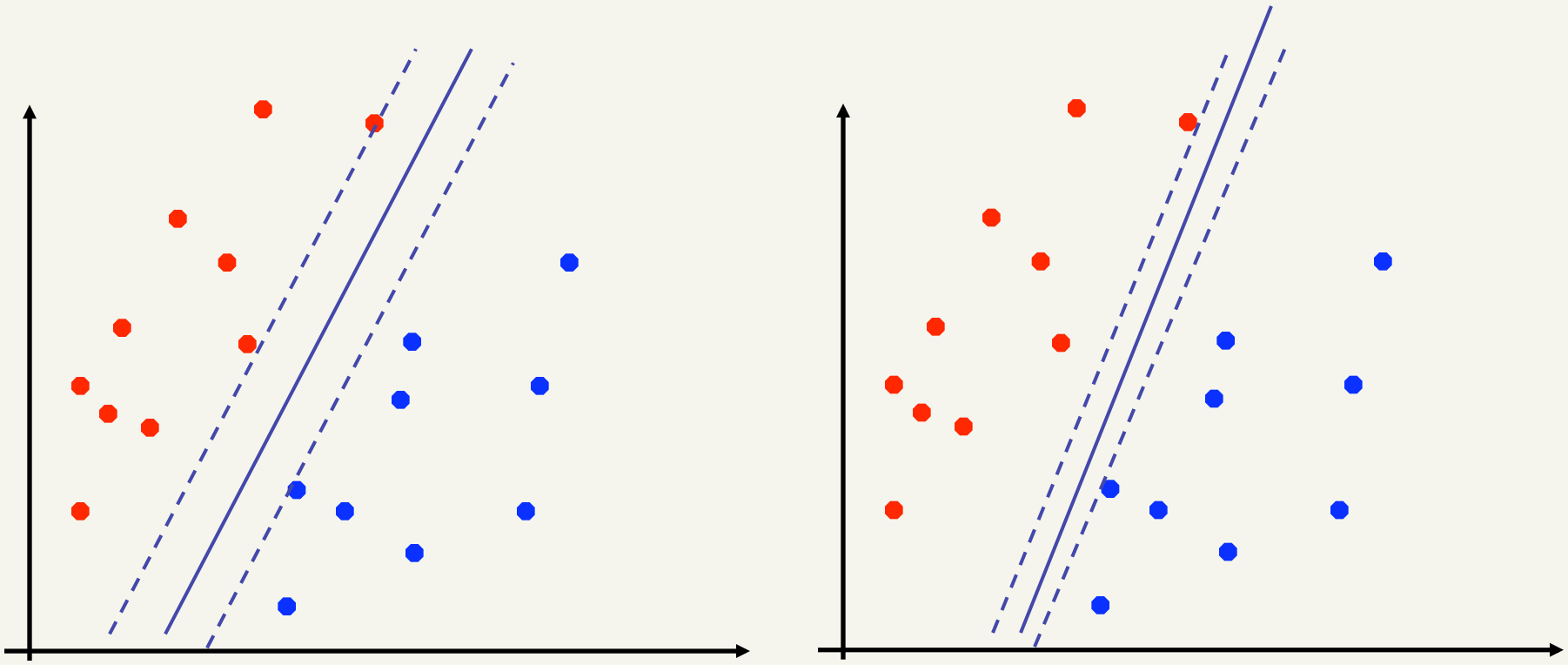


*but other discriminative methods often perform very similarly

Margin maximization

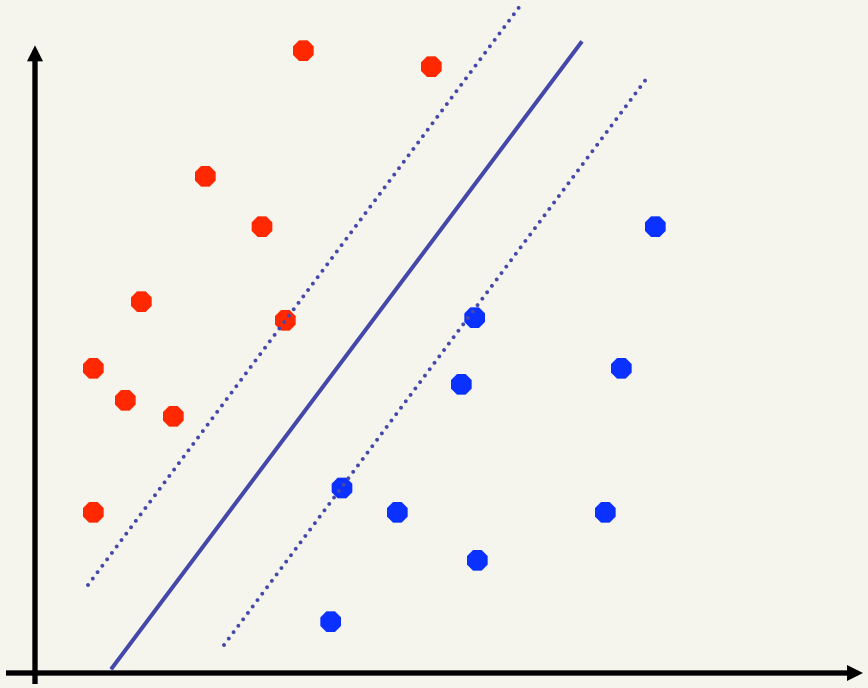


Margin maximization



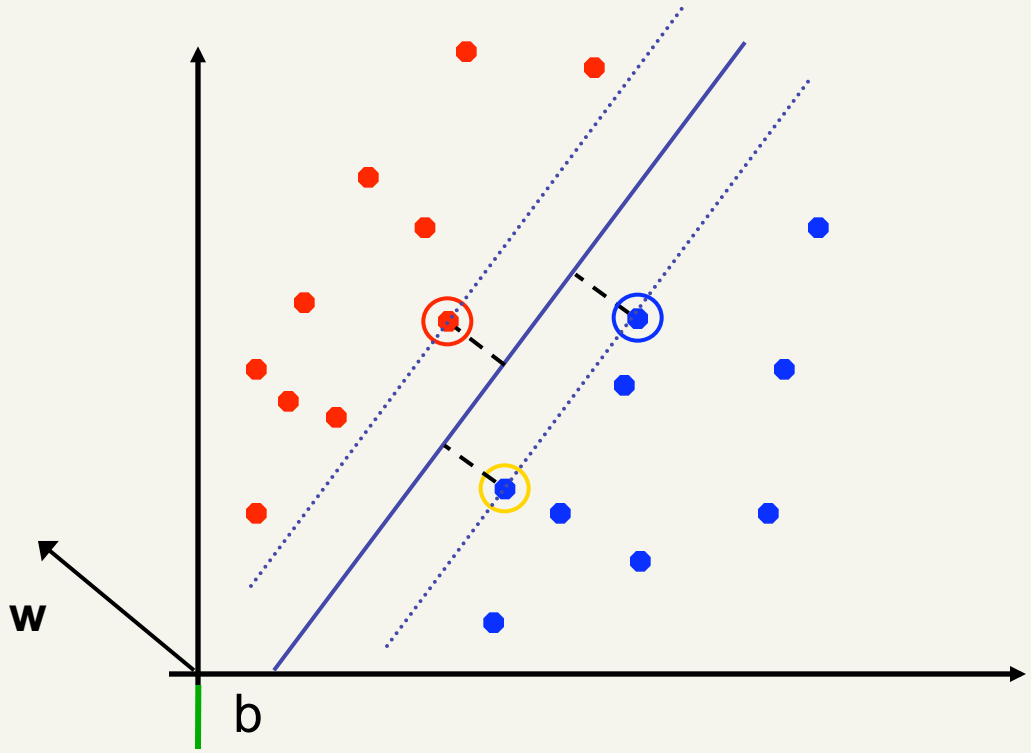
Measuring the margin

What defines a hyperplane?



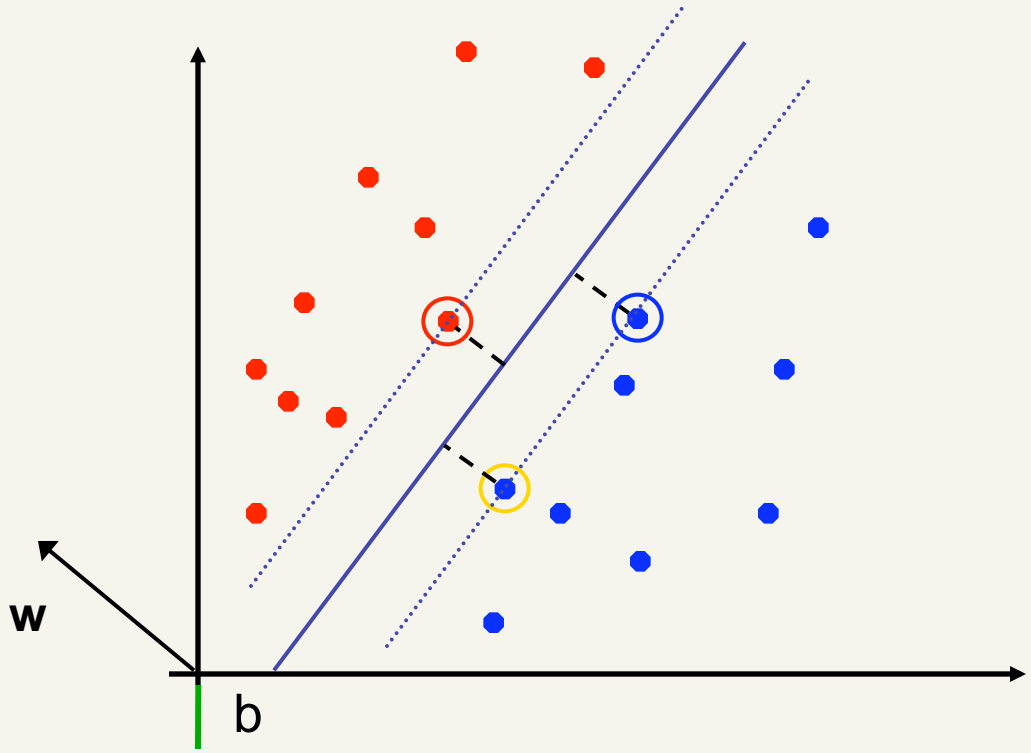
Measuring the margin

The *support vectors* define the hyperplane and the margin



Measuring the margin

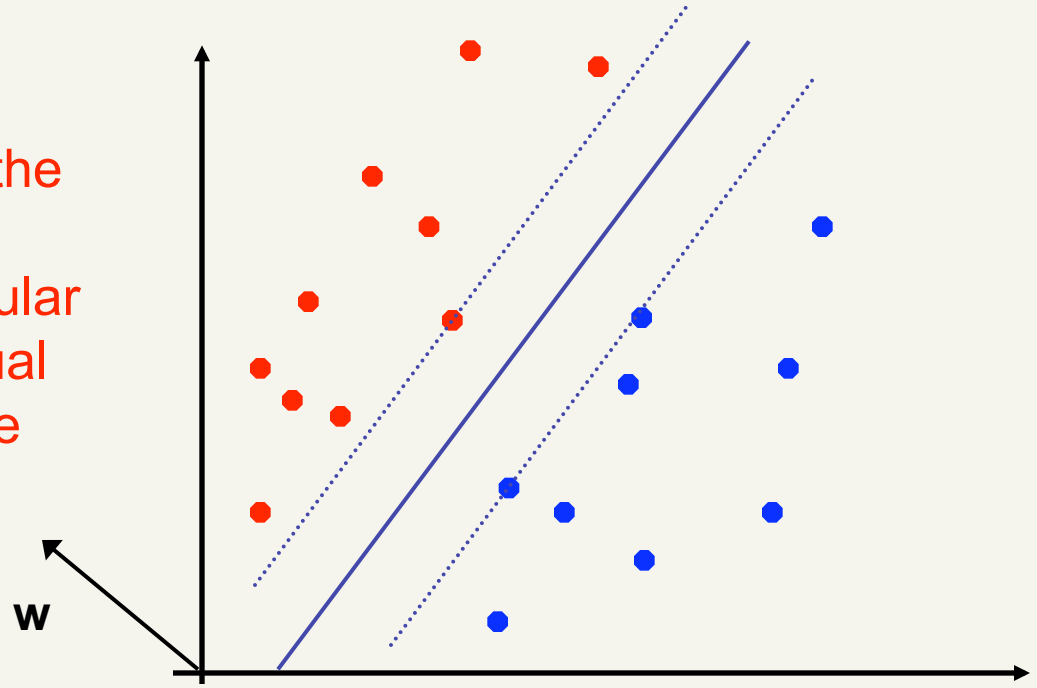
In an n-dimensional space, how can we represent this hyperplane?



Measuring the margin

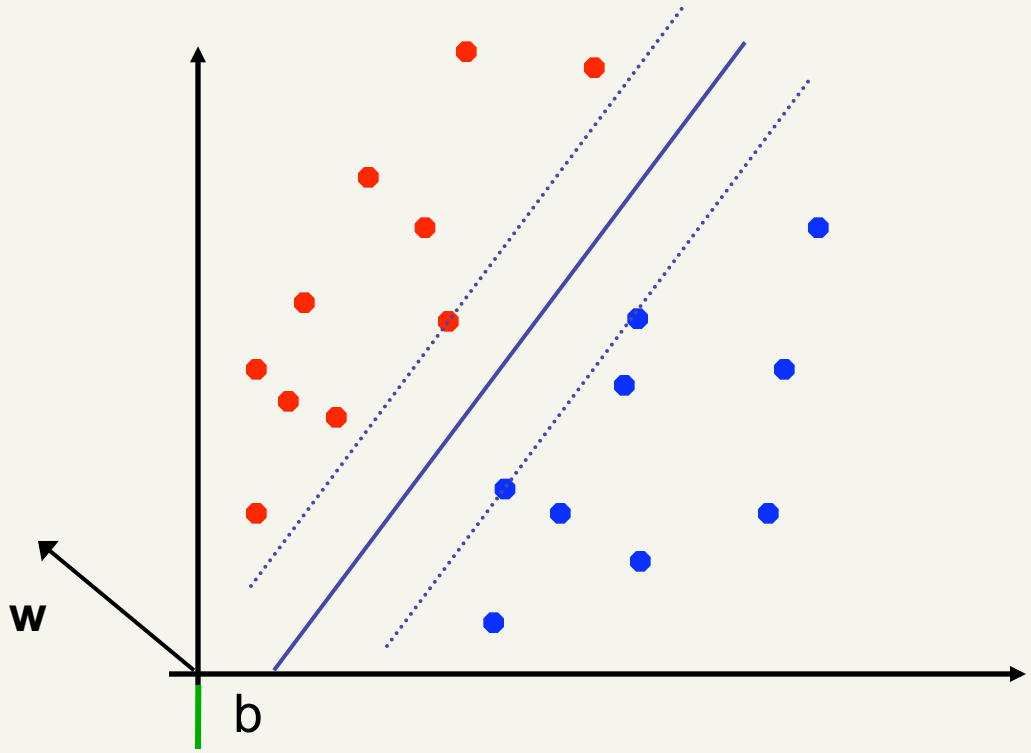
An n dimensional normal vector, **w** and?

Note that the vector is perpendicular to the actual hyperplane



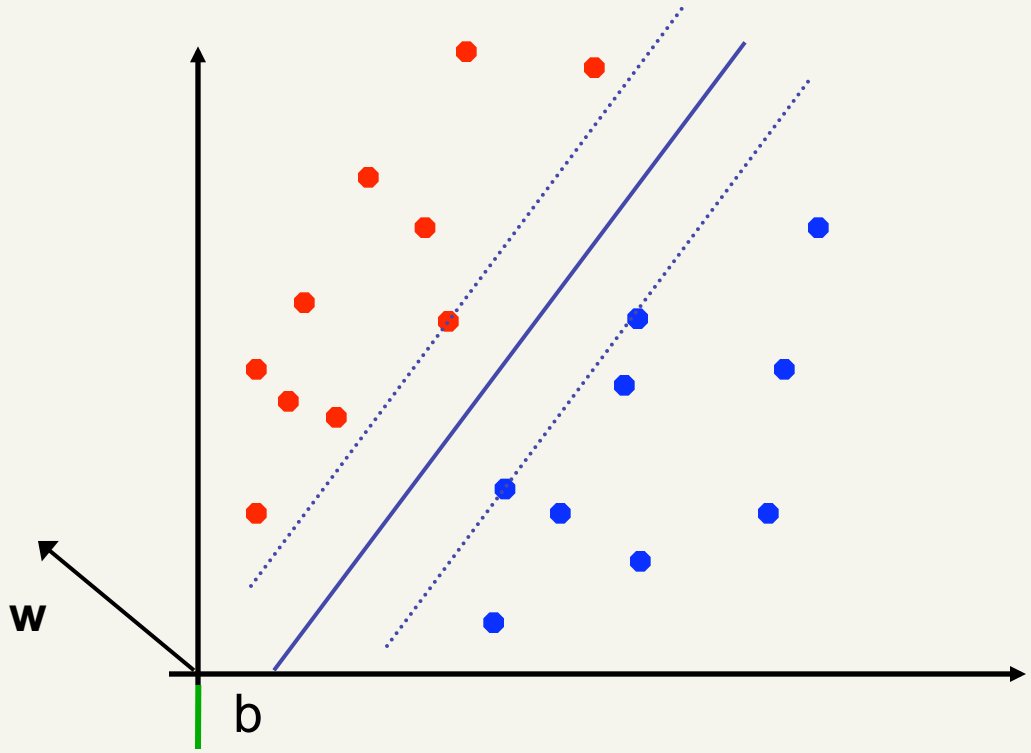
Measuring the margin

An n dimensional vector, **w** and an offset, b



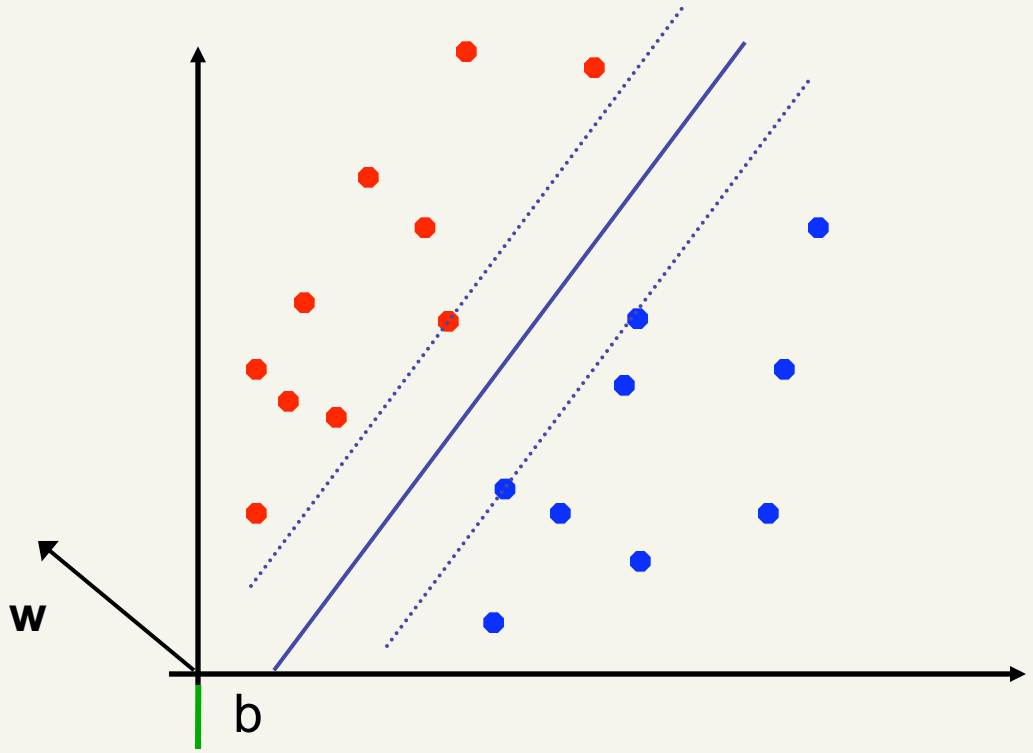
Measuring the margin

How do we classify points given this information?



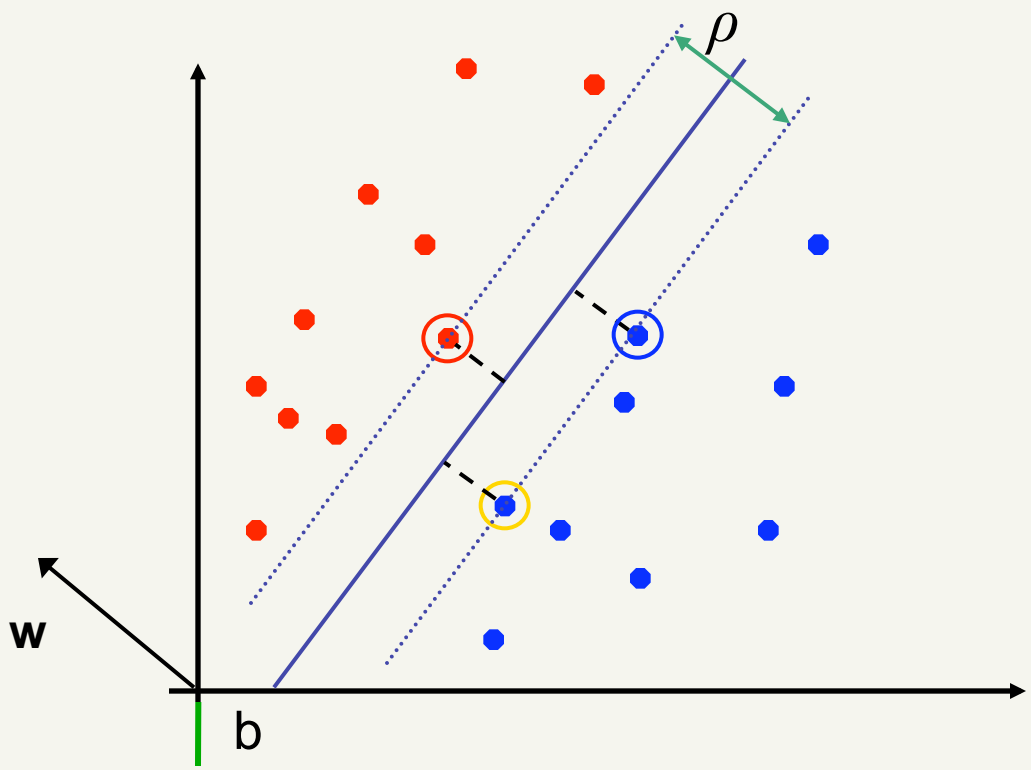
Measuring the margin

$$f(\mathbf{x}_i) = \text{sign}(\mathbf{w}^T\mathbf{x}_i + b)$$



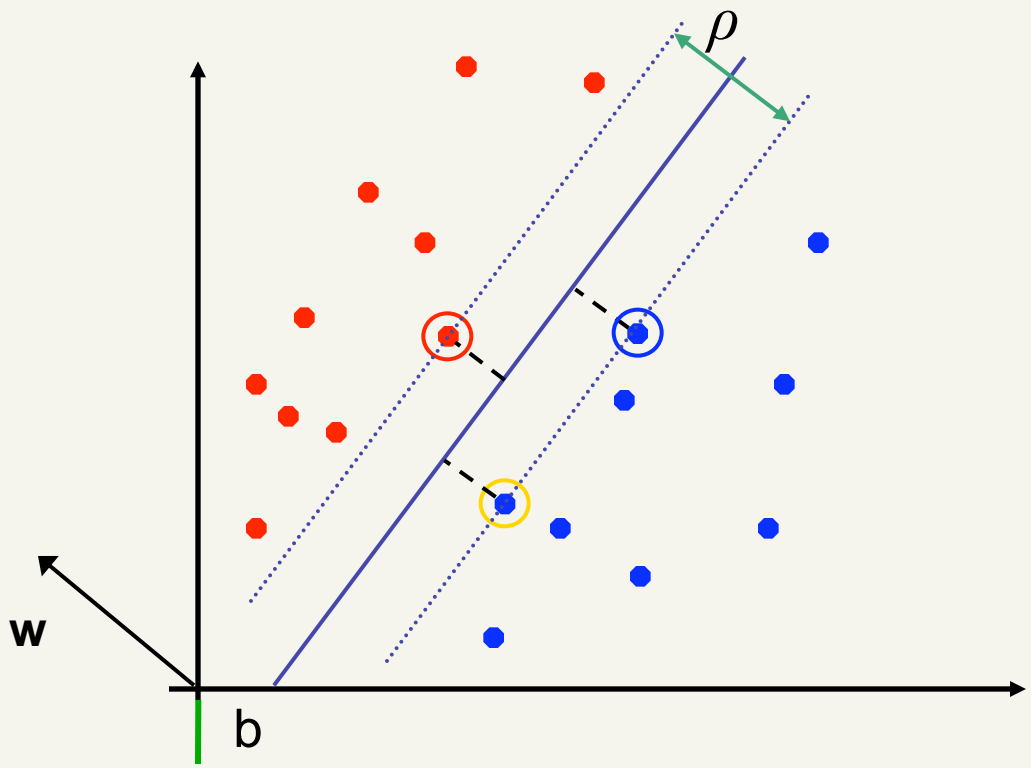
Measuring the margin

How can we calculate margin?



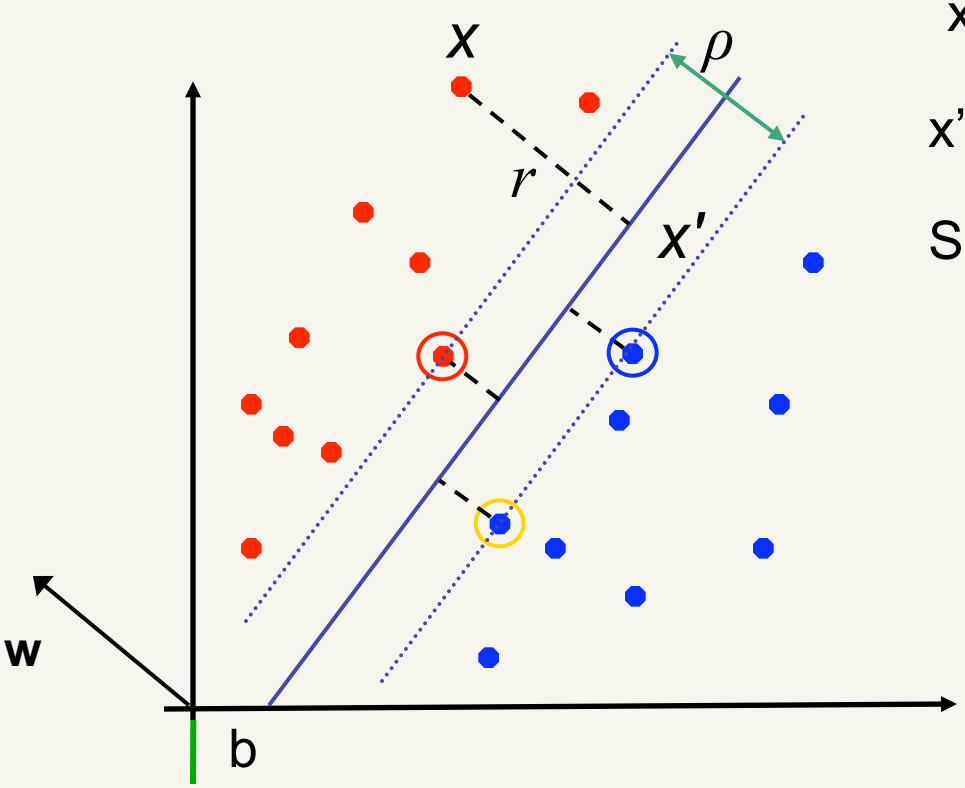
Measuring the margin

Minimum of the distance from the hyperplane to any point(s) (specifically the support vectors)



Measuring the margin

Want to calculate r



x' – x is perpendicular to hyperplane

w/|w| is the unit vector in direction of w

x' = x – rw/|w|

x' satisfies $w^T x' + b = 0$ because it's on w^T

So $w^T(x - rw/|w|) + b = 0$

$w^T x - w^T rw/|w| + b = 0$

$w^T x - w^T rw|w|/|w||w| + b = 0$

$w^T x - w^T rw|w|/w^T w + b = 0$

$w^T x - r|w| + b = 0$

$|w| = \text{sqrt}(w^T w)$

$$r = y\frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$$

Linear SVM Mathematically

The linearly separable case

- Assume that all data is at least distance 1 from the hyperplane, then the following two constraints follow for a training set $\{(\mathbf{x_i}, y_i)\}$

$$\mathbf{w^T x_i} + b \geq 1 \quad \text{if } y_i = 1$$

$$\mathbf{w^T x_i} + b \leq -1 \quad \text{if } y_i = -1$$

- For support vectors, the inequality becomes an equality
- Then, since each example's distance from the hyperplane is

$$r = y\frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$$

- The margin is:

$$\rho = \frac{2}{\|\mathbf{w}\|}$$

Linear SVMs Mathematically (cont.)

- Then we can formulate the *quadratic optimization problem:*

Find $\mathbf{w}$ and b such that

$\rho = \frac{2}{\|\mathbf{w}\|}$ is maximized; and for all $\{(\mathbf{x_i}, y_i)\}$

$\mathbf{w^T x_i} + b \geq 1$ if $y_i = 1$; $\mathbf{w^T x_i} + b \leq -1$ if $y_i = -1$

- A better formulation (min **||w||** = max 1/ **||w||**):

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi(w)} = \mathbf{w}^{\mathrm{T}}\mathbf{w}$ is minimized;

and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i (\mathbf{w^T x_i} + b) \geq 1$

Solving the Optimization Problem

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \mathbf{w}^T\mathbf{w}$ is minimized;
and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- This is now optimizing a *quadratic* function subject to *linear* constraints
- Quadratic optimization problems are a well-known class of mathematical programming problem, and many (intricate) algorithms exist for solving them (with many special ones built for SVMs)
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:

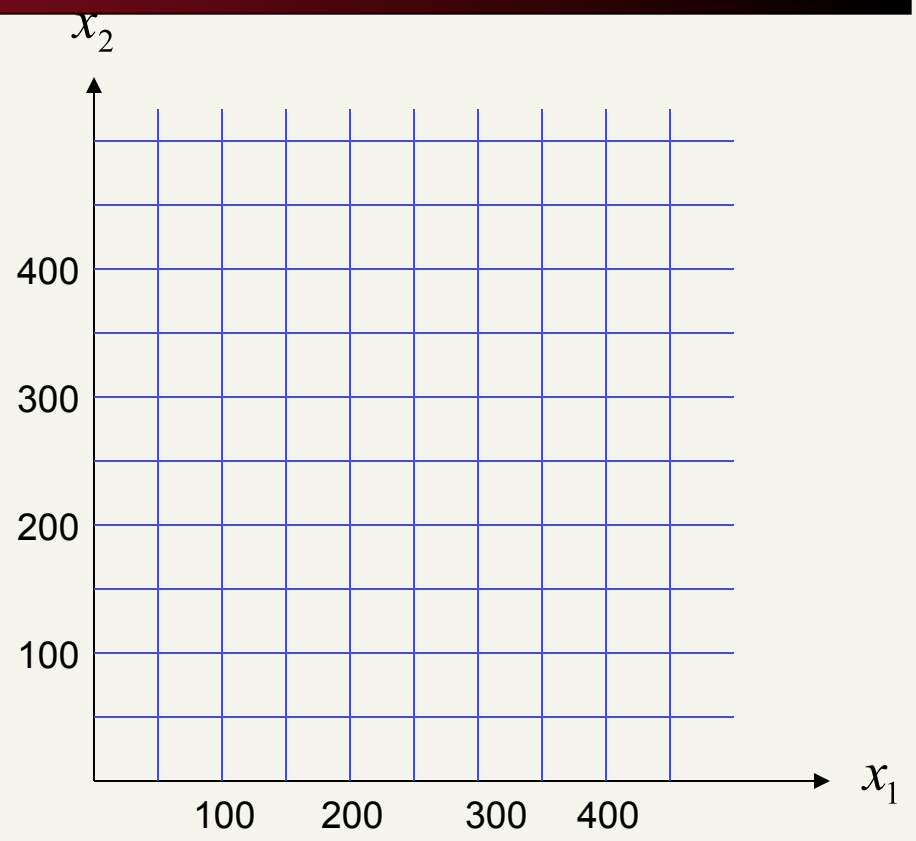
Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

An LP example

maximize $x_1 + 6x_2$

subject to

$$\begin{aligned} x_1 &\leq 200 \\ x_2 &\leq 300 \\ x_1 + x_2 &\leq 400 \\ x_1, x_2 &\geq 0 \end{aligned}$$

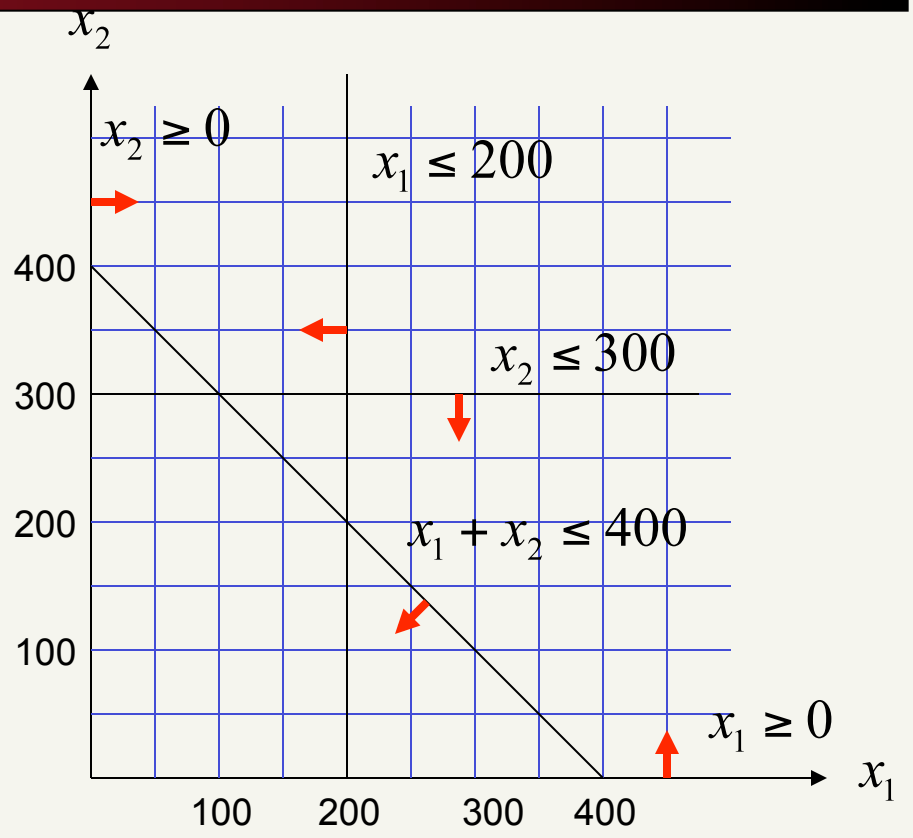


An LP example

maximize $x_1 + 6x_2$

subject to

$$\begin{aligned} x_1 &\leq 200 \\ x_2 &\leq 300 \\ x_1 + x_2 &\leq 400 \\ x_1, x_2 &\geq 0 \end{aligned}$$



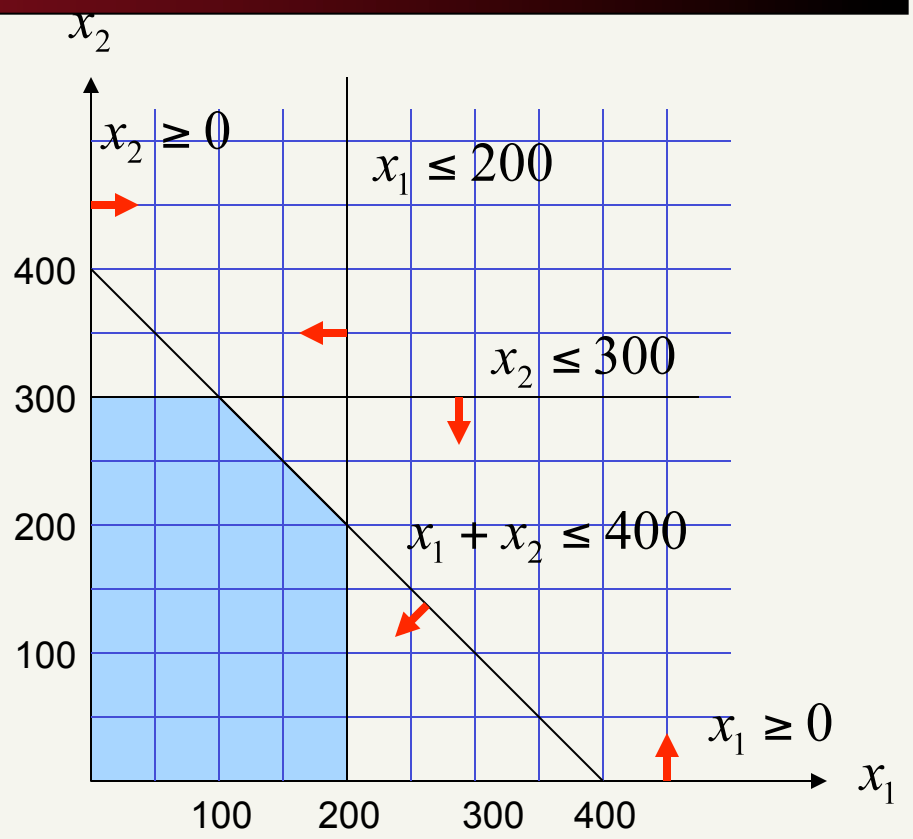
Where is the feasibility region?

An LP example

maximize $x_1 + 6x_2$

subject to

$$\begin{aligned} x_1 &\leq 200 \\ x_2 &\leq 300 \\ x_1 + x_2 &\leq 400 \\ x_1, x_2 &\geq 0 \end{aligned}$$

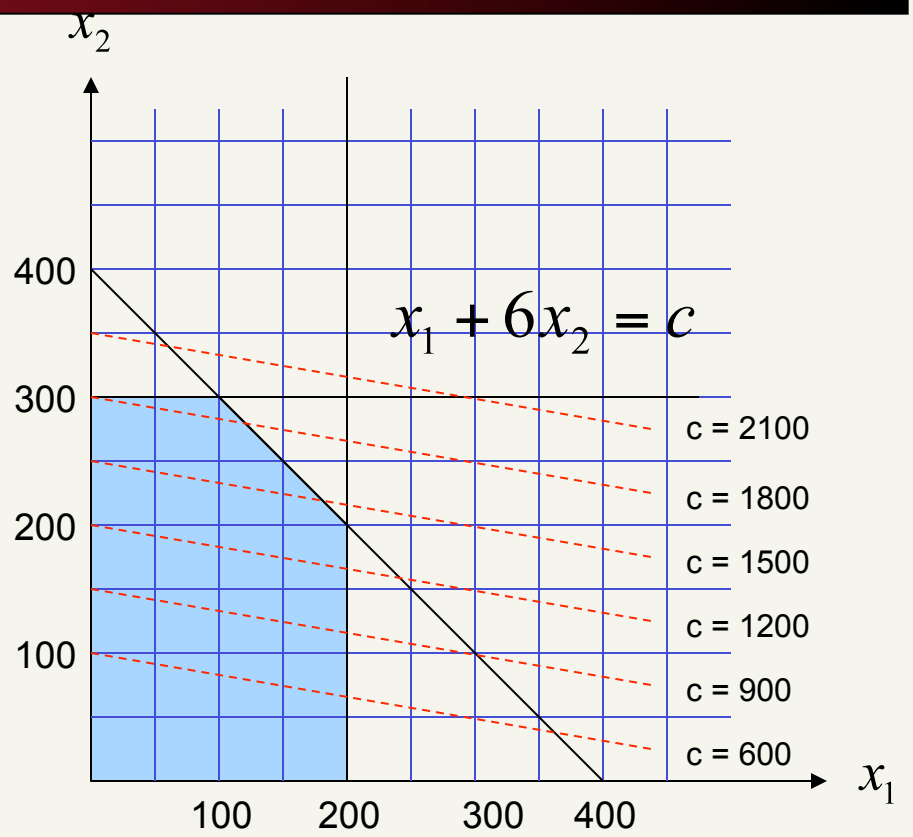


An LP example

maximize $x_1 + 6x_2$

subject to

$$\begin{aligned} x_1 &\leq 200 \\ x_2 &\leq 300 \\ x_1 + x_2 &\leq 400 \\ x_1, x_2 &\geq 0 \end{aligned}$$

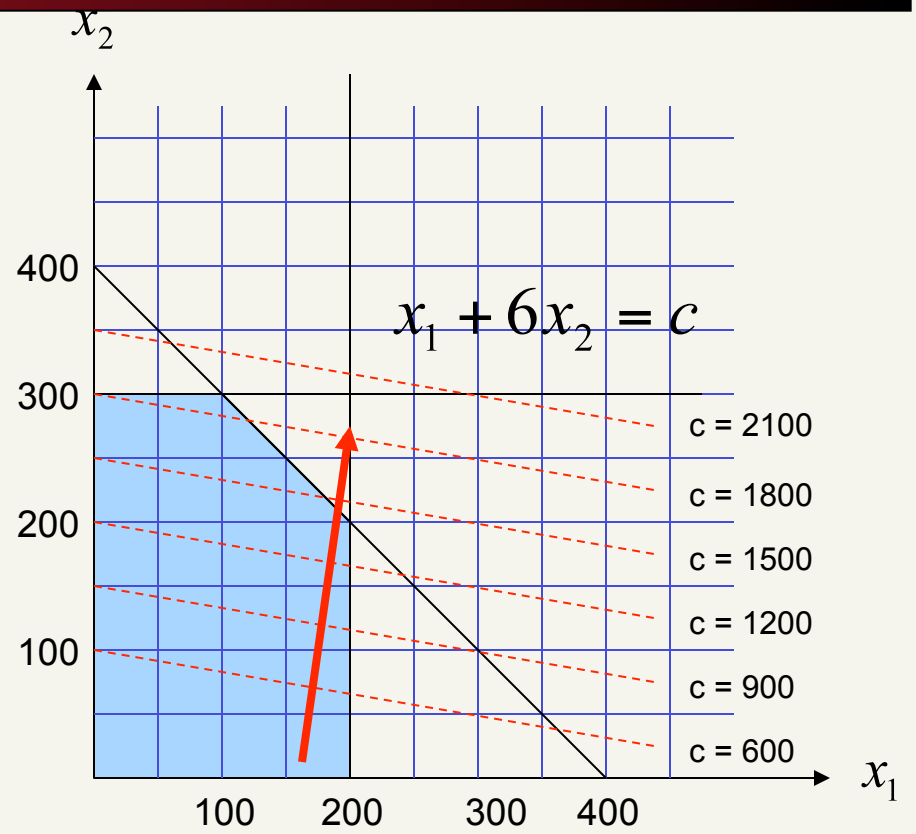


An LP example

maximize $x_1 + 6x_2$

subject to

$$\begin{aligned} x_1 &\leq 200 \\ x_2 &\leq 300 \\ x_1 + x_2 &\leq 400 \\ x_1, x_2 &\geq 0 \end{aligned}$$



to maximize, move as far in this direction as the constraints allow

The Optimization Problem Solution

- The solution has the form:

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x_i} \qquad b = y_k - \mathbf{w}^T \mathbf{x_k} \text{ for any } \mathbf{x_k} \text{ such that } \alpha_k \neq 0$$

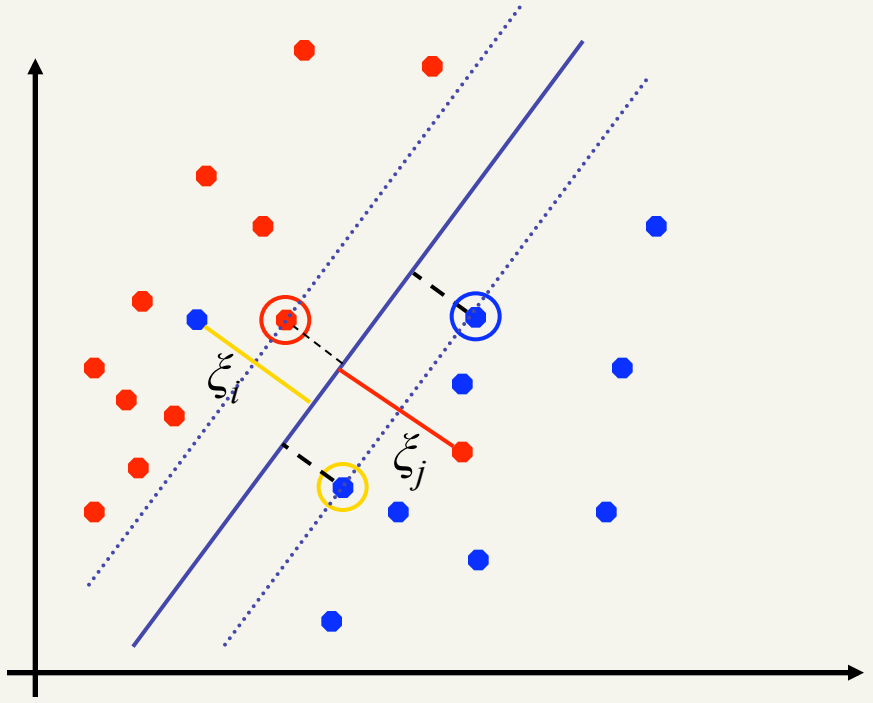
- Each non-zero α_i indicates that corresponding $\mathbf{x_i}$ is a support vector.
- Then the classifying function will have the form:

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x_i}^T \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point **x** and the support vectors $\mathbf{x_i}$ – we will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x_i}^T \mathbf{x_j}$ between all pairs of training points.

Soft Margin Classification

- If the training data is not linearly separable, *slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples.
- Allow some errors
  - Let some points be moved to where they belong, at a cost
- Still, try to minimize training set errors, and to place hyperplane “far” from each class (large margin)



Soft Margin Classification Mathematically

- The old formulation:

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w^T x_i} + b) \geq 1$

- The new formulation incorporating slack variables:

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w^T x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- Parameter C can be viewed as a way to control overfitting – a regularization term

Linear SVMs: Summary

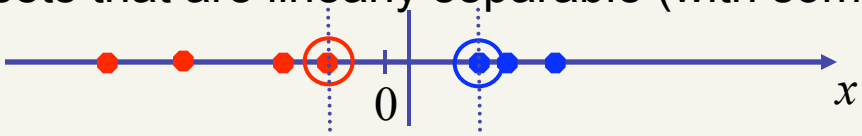
- The classifier is a *separating hyperplane.*
- Most "important" training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\mathbf{x}_i$ are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the dual formulation of the problem and in the solution training points appear only inside inner products:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \le \alpha_i \le C$ for all α_i

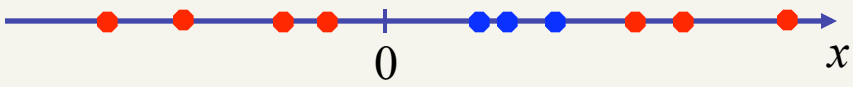
$$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$$

Non-linear SVMs

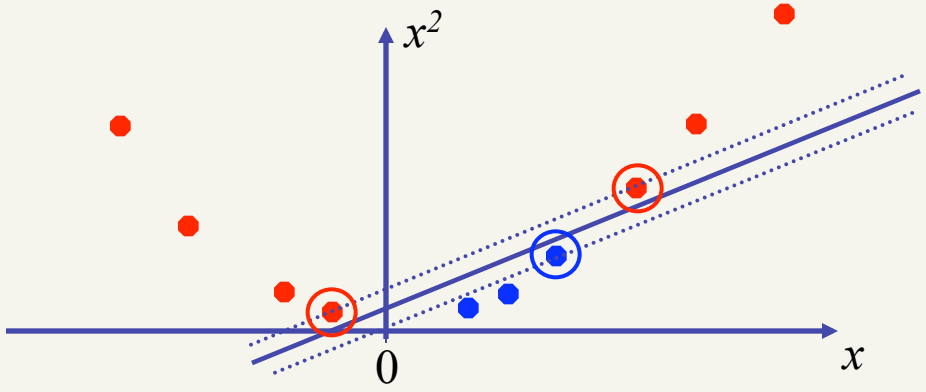
- Datasets that are linearly separable (with some noise) work out great:



- But what are we going to do if the dataset is just too hard?

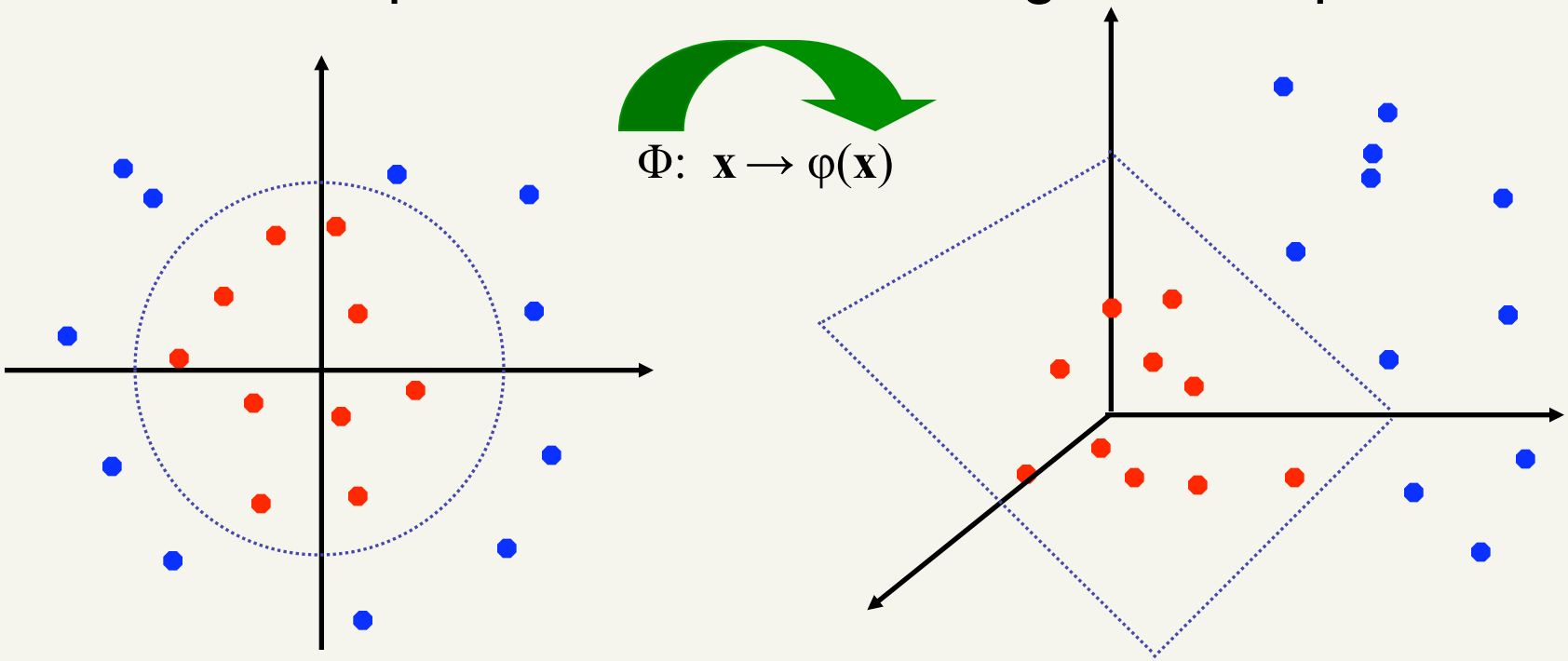


- How about … mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



The “Kernel Trick”

- The linear classifier relies on an inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^T\mathbf{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.

Kernels

- Why use kernels?
  - Make non-separable problem separable.
  - Map data into better representational space
- Common kernels
  - Linear
  - Polynomial $K(x,z) = (1+x^Tz)^d$
    - Gives feature conjunctions
  - Radial basis function (infinite dimensional space)

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2}$$

- Haven't been very useful in text classification

Evaluation: Micro- vs. Macro-Averaging

- If we have more than one class, how do we combine multiple performance measures into one quantity?
  - Macroaveraging: Compute performance for each class, then average
  - Microaveraging: Collect decisions for all classes, compute contingency table, evaluate

- Benefits and drawbacks?

Which classifier do I use for a given text classification problem?

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

Manually written rules

- No training data, adequate editorial staff?
- Never forget the hand-written rules solution!
  - If (wheat or grain) and not (whole or bread) then
    - Categorize as grain
- In practice, rules get a lot bigger than this
  - Can also be phrased using tf or tf.idf weights
- With careful crafting (human tuning on development data) performance is high:
  - Construe: 94% recall, 84% precision over 675 categories (Hayes and Weinstein 1990)
- Amount of work required is huge
  - Estimate 2 days per class … plus maintenance

Very little data?

- If you're just doing supervised classification, you should stick to something with high bias
  - There are theoretical results that Naïve Bayes should do well in such circumstances (Ng and Jordan 2002 NIPS)
- The interesting theoretical answer is to explore semi-supervised training methods:
  - Bootstrapping, EM over unlabeled documents, …
- The practical answer is to get more labeled data as soon as you can
  - How can you insert yourself into a process where humans will be willing to label data for you??

A reasonable amount of data?

- We can use any number of different classifiers
- Roll out the SVM!

- But if you are using an SVM/NB etc., you should probably be prepared with the “hybrid” solution where there is a Boolean overlay
  - Or else to use user-interpretable Boolean-like models like decision trees
  - Users like to hack, and management likes to be able to implement quick fixes immediately

A huge amount of data?

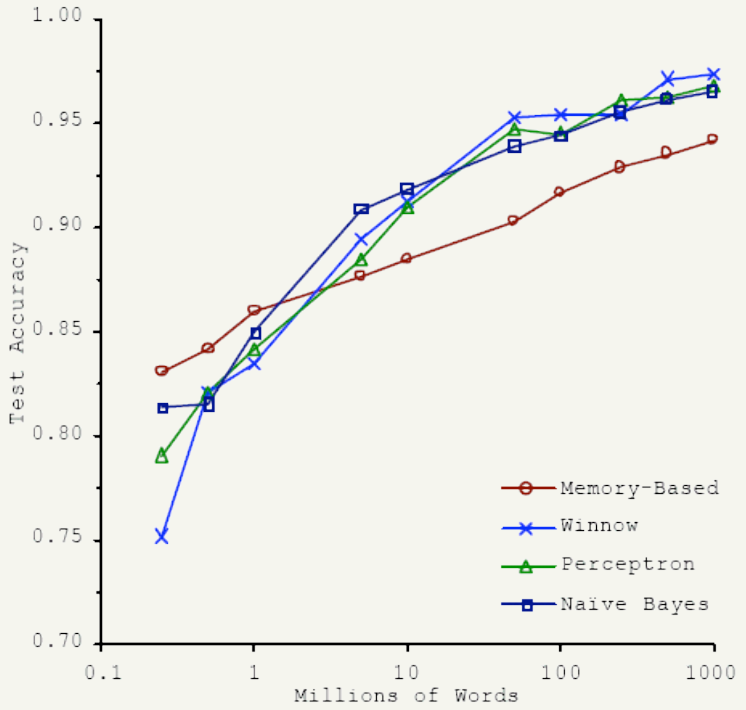
- This is great in theory for doing accurate classification…
- But it could easily mean that expensive methods like SVMs (train time) or kNN (test time) are quite impractical

- Naïve Bayes can come back into its own again!
  - Or other advanced methods with linear training/test complexity like regularized logistic regression (though much more expensive to train)
- When you have lots of data, simple things work well!

A huge amount of data?

- With enough data the choice of classifier may not matter much, and the best choice may be unclear
  - Data: Brill and Banko on context-sensitive spelling correction

- But the fact that you have to keep doubling your data to improve performance is a little unpleasant



The Real World

P. Jackson and I. Moulinier: *Natural Language Processing for Online Applications*

- "There is no question concerning the commercial value of being able to classify documents automatically by content. There are myriad potential applications of such a capability for corporate Intranets, government departments, and Internet publishers"

- "Understanding the data is one of the keys to successful categorization, yet this is an area in which most categorization tool vendors are extremely weak. Many of the 'one size fits all' tools on the market have not been tested on a wide range of content types."

Does putting in “hacks” help?

- You can get a lot of value by differentially weighting contributions from different document zones:
  - Upweighting title words helps (Cohen & Singer 1996)
    - Doubling the weighting on the title words is a good rule of thumb
  - Upweighting the first sentence of each paragraph helps (Murata, 1999)
  - Upweighting sentences that contain title words helps (Ko *et al,* 2002)

Does stemming/lowercasing/… help?

- As always it's hard to tell, and empirical evaluation is normally the gold standard
- But note that the role of tools like stemming is rather different for TextCat vs. IR:
  - For IR, you often want to collapse forms of the verb *oxygenate* and *oxygenation*, since all of those documents will be relevant to a query for *oxygenation*
  - For TextCat, with sufficient training data, stemming *does no good.* It only helps in compensating for data sparseness (which can be severe in TextCat applications). *Overly aggressive stemming can easily degrade performance.*

References

- *IIR* 14
- Fabrizio Sebastiani. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1):1-47, 2002.
- Tom Mitchell, Machine Learning. McGraw-Hill, 1997.
- Yiming Yang & Xin Liu, A re-examination of text categorization methods. *Proceedings of SIGIR*, 1999.
- Evaluating and Optimizing Autonomous Text Classification Systems (1995) David Lewis. Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval
- Trevor Hastie, Robert Tibshirani and Jerome Friedman, *Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer-Verlag, New York.
- Open Calais: Automatic Semantic Tagging
  - Free (but they can keep your data), provided by Thompson/Reuters
- Weka: A data mining software package that includes an implementation of many ML algorithms