Web Mining and Recommender Systems

Classification (& Regression Recap)

Learning Goals

In this section we want to:

- Explore techniques for **classification**
- Try some simple solutions, and see why they might fail
- Explore more complex solutions, and their advantages and disadvantages
- Understand the relationship between classification and regression
- Examine how we can reliably **evaluate** classifiers under different conditions

Recap...

Previously we started looking at
supervised learning problems

$$f(\text{data}) \overset{?}{\rightarrow} \text{labels}$$

Recap...

We studied **linear regression**, in order to learn linear relationships between features and parameters to predict **real-valued** outputs

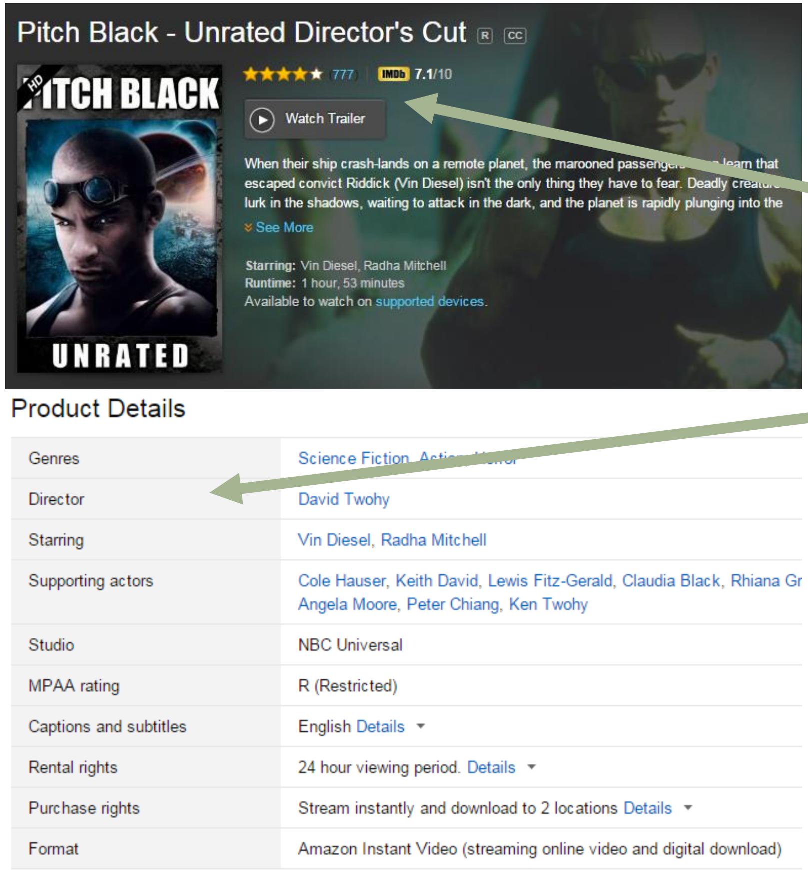
$$X\theta = y$$

matrix of features
(data)

unknowns
(which features are relevant)

vector of outputs
(labels)

Recap...



ratings

features

$$f(\text{user features}, \text{movie features}) \xrightarrow{?} \text{star rating}$$

Four important ideas:

1) Regression can be cast in terms of **maximizing a likelihood**

$$y_i = X_i \cdot \theta + \mathcal{N}(0, \sigma^2)$$

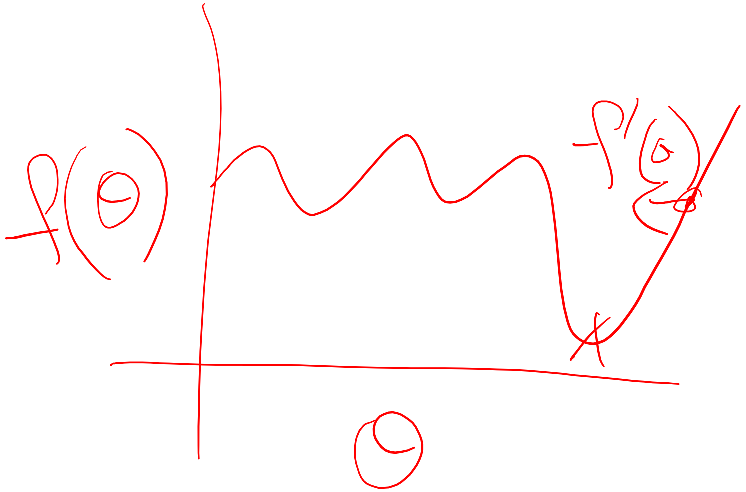
$$\max_\theta P_\theta(Y|X) = \min_\theta \frac{1}{N} \sum_i (y_i - X_i \cdot \theta)^2$$

Four important ideas:

2) Gradient descent for model optimization

1. Initialize θ at random
2. While (not converged) do

$$\theta := \theta - \alpha f'(\theta)$$



Four important ideas:

3) Regularization & Occam's razor

Regularization is the process of penalizing model complexity during training

$$\arg\min_{\theta} = \frac{1}{N}\|y - X\theta\|_2^2 + \lambda\|\theta\|_2^2$$

How much should we trade-off accuracy versus complexity?

Four important ideas:

4) Regularization pipeline

1. Training set – select model parameters
2. Validation set – to choose amongst models (i.e., hyperparameters)
3. Test set – just for testing!

[train | valid | test]

fit θ | choose λ | only use once!

Model selection

A **validation set** is constructed to "tune" the model's parameters

- Training set: used to **optimize the model's parameters**
- Test set: used to report how well we expect the model to perform on **unseen data**
- Validation set: used to **tune** any model parameters that are not directly optimized

A few "theorems" about training, validation, and test sets

- The training error **increases** as lambda **increases**
- The validation and test error are at least as large as the training error (assuming infinitely large random partitions)
- The validation/test error will usually have a "sweet spot" between under- and over-fitting

Up next...

How can we predict **binary** or **categorical** variables?

$$f(\text{data}) \overset{?}{\rightarrow} \text{labels}$$

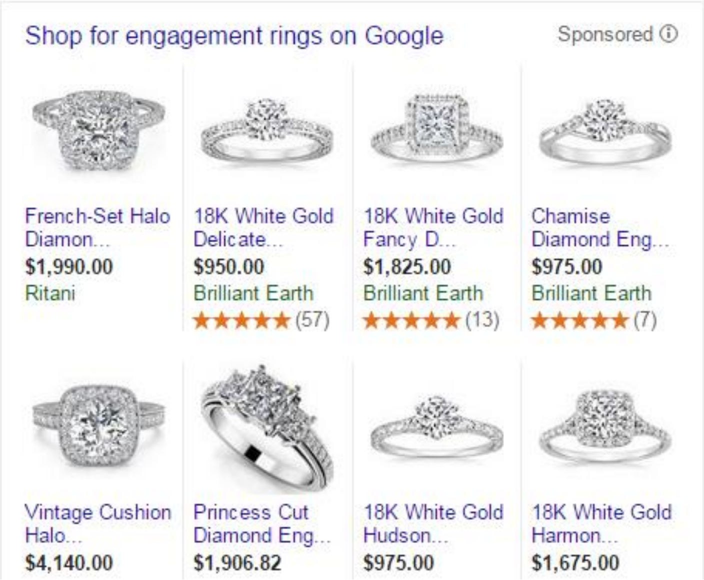
{0,1}, {True, False}

{1, ... , N}

Up next...

Will I **purchase** this product?

(yes)



Will I **click on** this ad?

(no)

Up next…

What animal appears in this image?

(mandarin duck)

Up next...

What are the **categories** of the item being described?

(book, fiction, philosophical fiction)

From Booklist

Houellebecq's deeply philosophical novel is about an alienated young man searching for happiness in the computer age. Bored with the world and too weary to try to adapt to the foibles of friends and coworkers, he retreats into himself, descending into depression while attempting to analyze the passions of the people around him. Houellebecq uses his nameless narrator as a vehicle for extended exploration into the meanings and manifestations of love and desire in human interactions. Ironically, as the narrator attempts to define love in increasingly abstract terms, he becomes less and less capable of experiencing that which he is so desperate to understand. Intelligent and well written, the short novel is a thought-provoking inspection of a generation's confusion about all things sexual. Houellebecq captures precisely the cynical disillusionment of disaffected youth. *Bonnie Johnston --This text refers to an out of print or unavailable edition of this title.*

Up next...

We'll attempt to build **classifiers** that make decisions according to rules of the form

$$y_i = \begin{cases} 1 & \text{if } X_i \cdot \theta > 0 \\ 0 & \text{otherwise} \end{cases}$$

Up later…

1. Naïve Bayes

Assumes an **independence** relationship between the features and the class label and "learns" a simple model by counting

2. Logistic regression

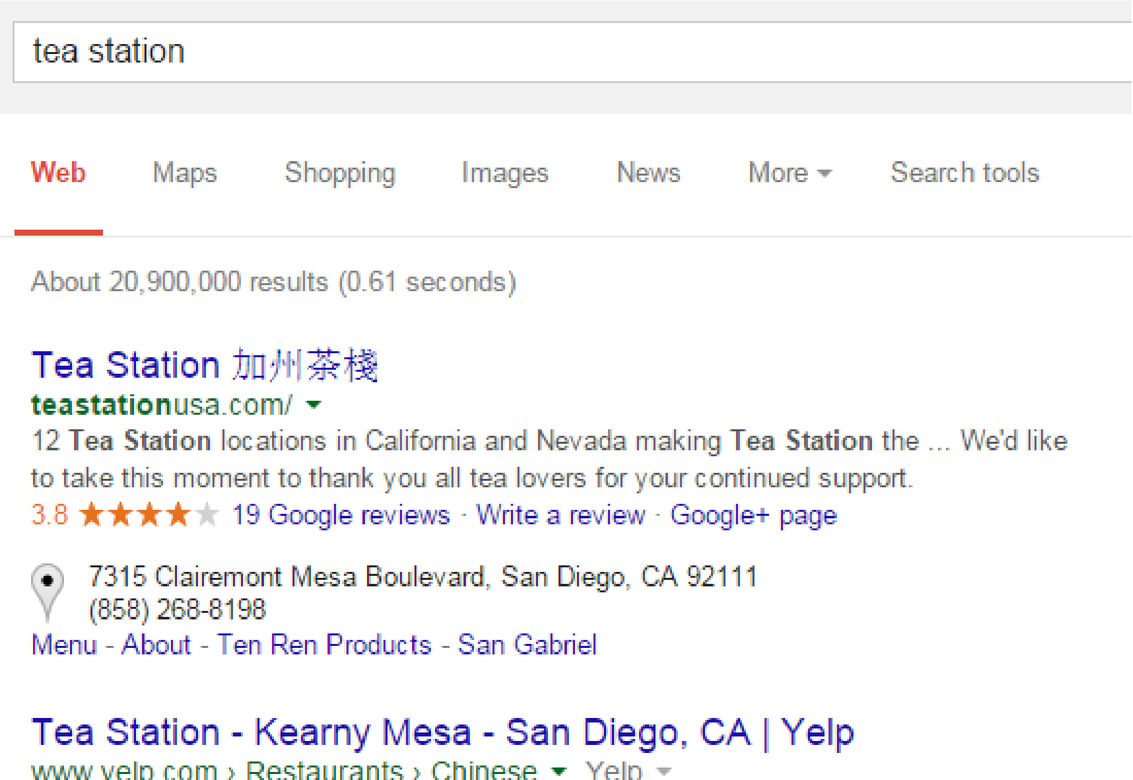
Adapts the **regression** approaches we saw last week to binary problems

3. Support Vector Machines

Learns to classify items by finding a hyperplane that separates them

Up later...

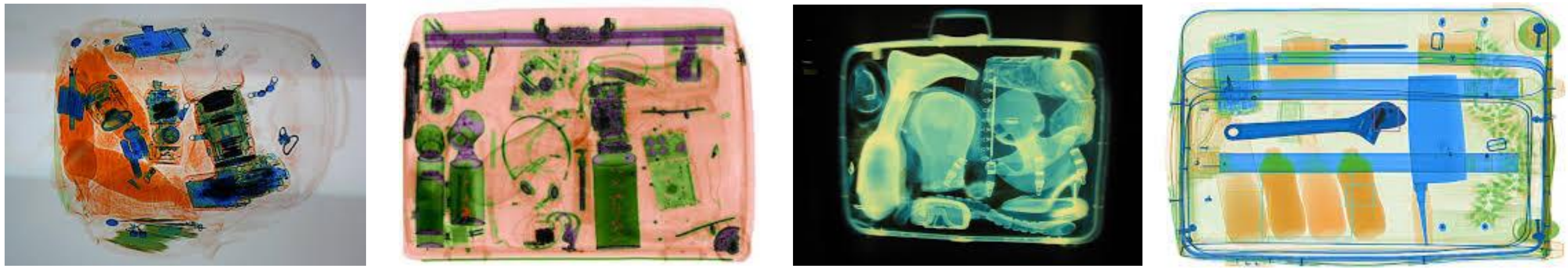
Ranking results in order of how likely they are to be relevant



Up later…

Evaluating classifiers

- False positives are nuisances but false negatives are disastrous (or vice versa)
- Some classes are very rare
- When we only care about the "most confident" predictions



e.g. which of these bags contains a weapon?

Web Mining and Recommender Systems

Classification: Naïve Bayes

Learning Goals

- Introduce the **Naïve Bayes** classifier
- We study Naïve Bayes largely to learn about the complications involved in building classifiers

Naïve Bayes

We want to associate a probability with a label and its negation:

$$p(label|data)$$

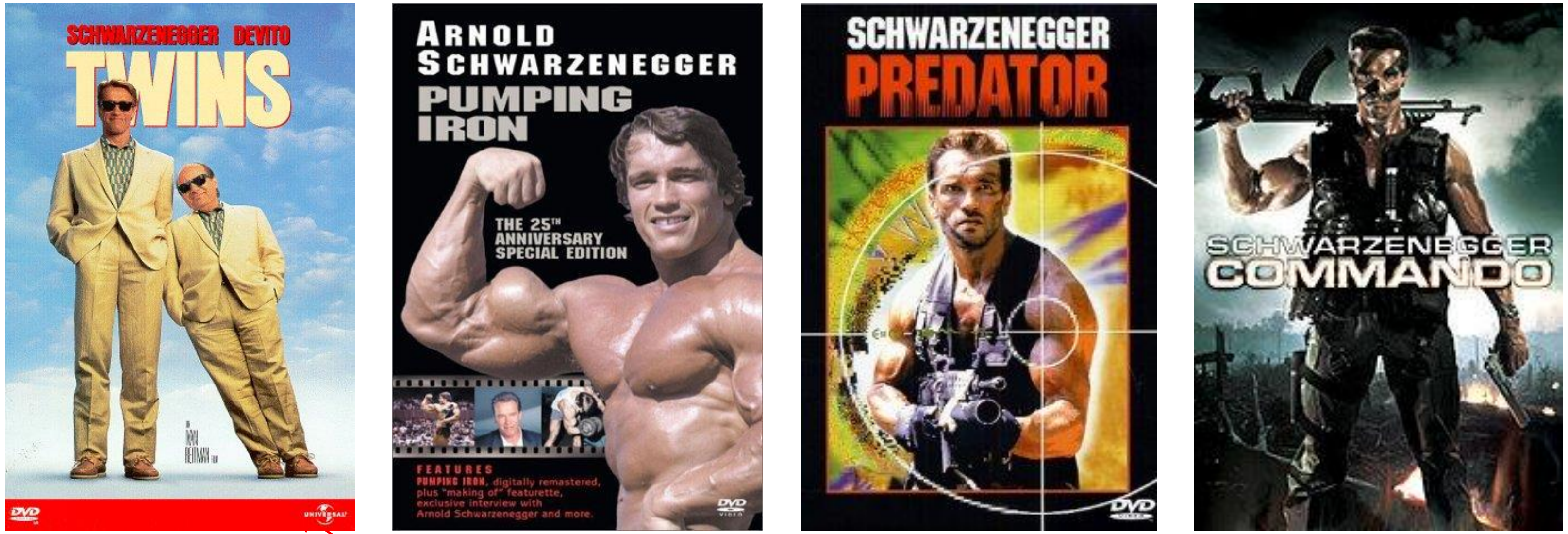
$$p(\neg label|data)$$

(classify according to whichever probability is greater than 0.5)

Q: How far can we get just by counting?

Naïve Bayes

e.g. p(movie is "action" | schwarzenegger in cast)



$p(b|a) = \frac{p(a,b)}{p(a)}$

Just count!

#films with Arnold = 45

#**action** films with Arnold = 32

$\frac{p(a)}{p(a,b)}$

p(movie is "action" | schwarzenegger in cast) = 32/45

Naïve Bayes

What about:

p(movie is “action” |
schwarzenegger in cast **and**
release year = 2017 **and**
mpaa rating = PG **and**
budget < $1000000
)

#(training) fims with Arnold, released in 2017, rated PG, with a budget below $1M = 0

#(training) action fims with Arnold, released in 2017, rated PG, with a budget below $1M = 0

Naïve Bayes

Q: If we've never seen this combination of features before, what can we conclude about their probability?

A: We need some **simplifying assumption** in order to associate a probability with this feature combination

Naïve Bayes

Naïve Bayes assumes that features are **conditionally independent** given the label

$$(feature_i \perp\!\!\!\perp feature_j | label)$$

Naïve Bayes

$$(feature_i \perp\!\!\!\perp feature_j | label)$$

$$a \perp\!\!\!\perp b \rightarrow p(a, b) = p(a)p(b)$$

$$a \perp\!\!\!\perp b | c \rightarrow p(a, b|c) = p(a|c)p(b|c)$$

---

a = I'm wearing shorts

b = you're wearing shorts

c = It's summer

Conditional independence?

$$(a \perp\!\!\!\perp b | c)$$

(a is conditionally independent of b, given c)

"if you know **c**, then knowing **a** provides no additional information about **b**"

$$(\text{I remembered my umbrella} \perp\!\!\!\perp \text{the streets are wet} \mid \text{it's raining})$$

Naïve Bayes

$$(feature_i \perp\!\!\!\perp feature_j | label)$$

$$\downarrow$$

$$p(feature_i, feature_j | label)$$

$$=$$

$$p(feature_i | label) p(feature_j | label)$$

Naïve Bayes

posterior

prior

likelihood

$$p(label|features) = \frac{p(label)\, p(features|label)}{p(features)}$$

evidence

Naïve Bayes

posterior prior likelihood

$$\underbrace{p(label|features)}_{\text{posterior}} = \frac{\overbrace{p(label)}^{\text{prior}}\,\overbrace{p(features|label)}^{\text{likelihood}}}{\underbrace{p(features)}_{\text{evidence}}}$$

evidence

due to our conditional independence assumption:

$$p(label|features) = \frac{p(label)\prod_i p(feature_i|label)}{p(features)}$$

Naïve Bayes

$$\boxed{p(label|features)} = \frac{p(label)\prod_i p(feature_i|label)}{\underbrace{p(features)}_{?}}$$

$$\boxed{p(\neg label \mid f)} = \frac{p(\neg label) \prod_i p(f_i \mid \neg label)}{p(features)}$$

The denominator doesn't matter, because we really just care about

$$p(label|features) \quad \text{vs.} \quad p(\neg label|features)$$

both of which have the same denominator

Naïve Bayes

$$\frac{p(y)\prod_i p(f_i|y)}{p(\neg y)\prod_i p(f_i|\neg y)} > 1 \; ?$$

The denominator doesn't matter, because we really just care about

$$p(label|features) \quad \text{vs.} \quad p(\neg label|features)$$

both of which have the same denominator

Learning Outcomes

- Introduced the **Naïve Bayes** classifier
- Discussed some of the challenges involved in classifier design

Web Mining and Recommender Systems

Naïve Bayes – Worked Example

Learning Goals

- Attempt to implement and experiment with a Naïve Bayes classifier

Example 1

Amazon editorial descriptions:

Amazon.com Review

For most children, summer vacation is something to look forward to. But not for our 13-year-ol
uncle, and cousin who detest him. The third book in J.K. Rowling's Harry Potter series catapults
Dursleys' dreadful visitor Aunt Marge to inflate like a monstrous balloon and drift up to the ceili
(and from officials at Hogwarts School of Witchcraft and Wizardry who strictly forbid students t
out into the darkness with his heavy trunk and his owl Hedwig.

As it turns out, Harry isn't punished at all for his errant wizardry. Instead he is mysteriously res
triple-decker, violently purple bus to spend the remaining weeks of summer in a friendly inn ca
his third year at Hogwarts explains why the officials let him off easily. It seems that Sirius Blac
loose. Not only that, but he's after Harry Potter. But why? And why do the Dementors, the guar
are unaffected? Once again, Rowling has created a mystery that will have children and adults cl
Fortunately, there are four more in the works. (Ages 9 and older) --*Karin Snelson --This text re*

50k descriptions:

http://jmcauley.ucsd.edu/cse258/data/amazon/book_descriptions_50000.json

Example 1

P(book is a children's book |
"wizard" is mentioned in the description **and**
"witch" is mentioned in the description)

Code available on course webpage

Example 1

Conditional independence assumption:

“if you know **a book is for children**, then knowing that **wizards are mentioned** provides no additional information about whether **witches are mentioned**”

obviously ridiculous

Double-counting

Q: What would happen if we trained two regressors, and attempted to "naively" combine their parameters?

Double-counting

$$\text{height} = 1.2\,\text{weight}$$

$$\text{height} = 15\,\text{shoe size}$$

$$\text{height} = 1.2\,\text{weight} + 15\,\text{shoe size}$$

Double-counting

A: Since both features encode essentially the same information, we'll end up **double-counting** their effect

Learning Outcomes

- Implemented a simple Naïve Bayes classifier, and studied its effectivenes in practice

Web Mining and Recommender Systems

Classification: Logistic Regression

Learning Goals

- Introduce the **logistic regression** classifier
- Show how to design classifiers by maximizing a likelihood function

Logistic regression

Logistic Regression also aims to model

$$p(label|data)$$

By training a classifier of the form

$$y_i = \begin{cases} 1 & \text{if } X_i \cdot \theta > 0 \\ 0 & \text{otherwise} \end{cases}$$

Logistic regression

Previously: regression

$$y_i = X_i \cdot \theta$$

Now: logistic regression

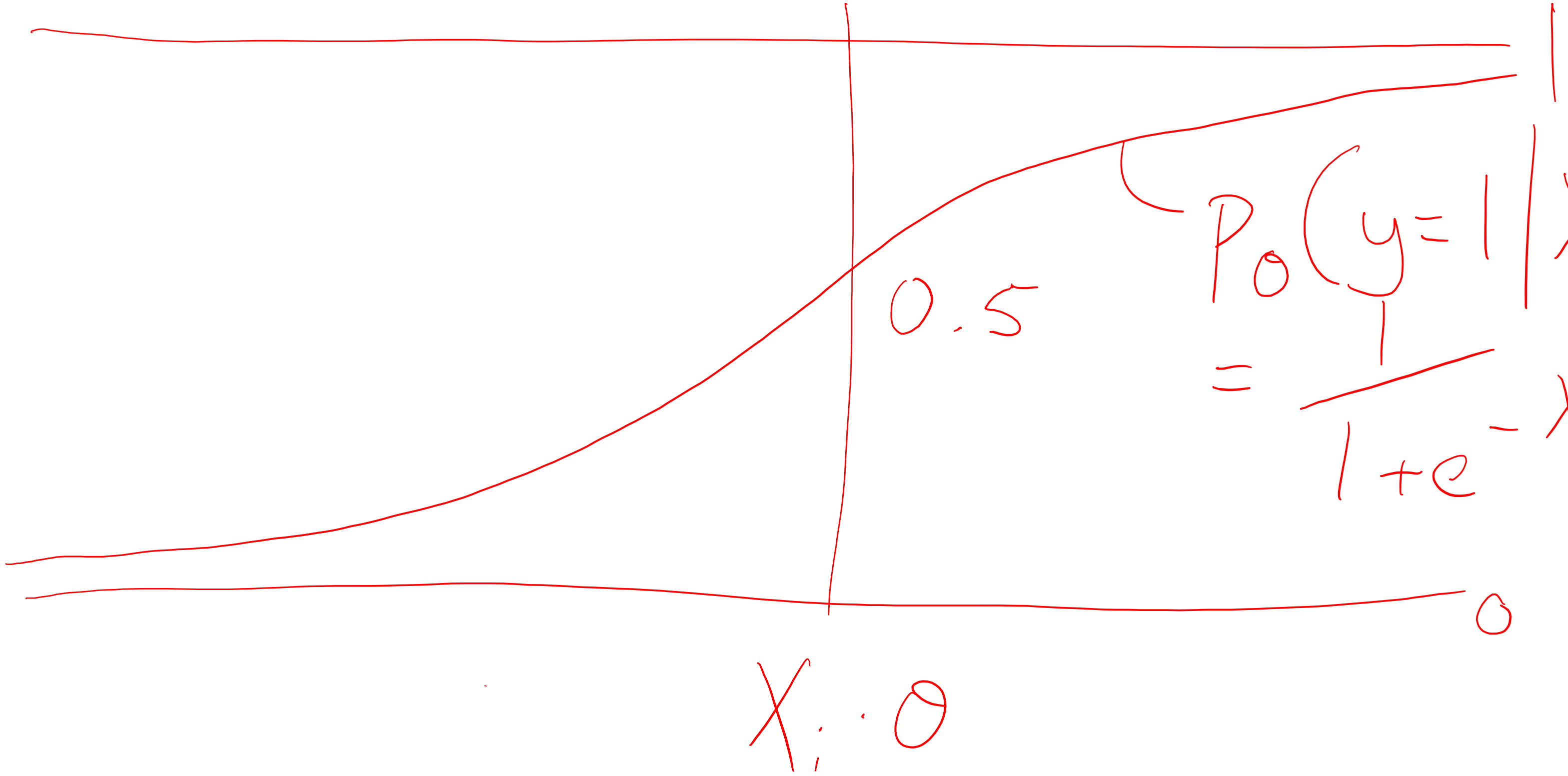
$$y_i = \begin{cases} 1 & \text{if } X_i \cdot \theta > 0 \\ 0 & \text{otherwise} \end{cases}$$

Logistic regression

Q: How to convert a real-valued expression ($X_i \cdot \theta \in \mathbb{R}$) Into a probability ($p_\theta(y_i|X_i) \in [0, 1]$)

Logistic regression

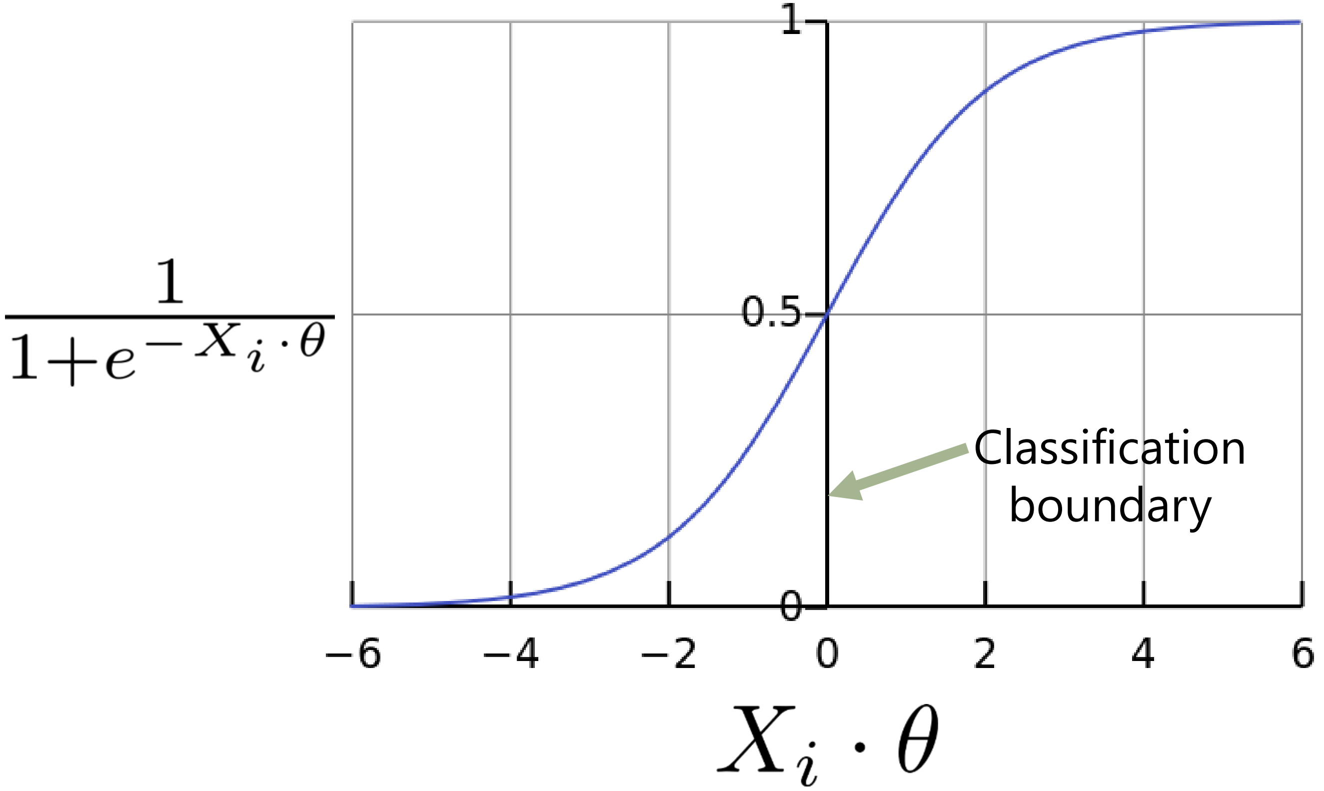
A: sigmoid function: $\sigma(t) = \frac{1}{1+e^{-t}}$



$P_\theta(y=1|X) = \frac{1}{1+e^{-X_i \cdot \theta}}$

Logistic regression

A: sigmoid function: $\sigma(t) = \frac{1}{1+e^{-t}}$



Logistic regression

Training:

$X_i \cdot \theta$ should be maximized when y_i is positive and minimized when y_i is negative

$$\arg\max_\theta \prod_{y_i=1} P_\theta(y_i|X_i) \prod_{y_i=0}\left(1 - P_\theta(y_i=1|X_i)\right)$$

Logistic regression

Training:

$X_i \cdot \theta$ should be maximized when y_i is positive and minimized when y_i is negative

$$\arg\max_\theta \prod_i \delta(y_i = 1) p_\theta(y_i|X_i) + \delta(y_i = 0)(1 - p_\theta(y_i|X_i))$$

$\delta(\text{arg})$ = 1 if the argument is true, = 0 otherwise

Logistic regression

How to optimize?

$$L_\theta(y|X) = \prod_{y_i=1} p_\theta(y_i|X_i) \prod_{y_i=0} (1 - p_\theta(y_i|X_i))$$

- Take logarithm
- **Subtract** regularizer
- Compute gradient
- Solve using gradient **ascent**

Logistic regression

$$L_\theta(y|X) = \prod_{y_i=1} p_\theta(y_i|X_i) \prod_{y_i=0}(1 - p_\theta(y_i|X_i))$$

$$\ell_\theta(y|X) = \sum_{y_i=1} \log\left(\frac{1}{1+e^{-X_i\cdot\theta}}\right) + \sum_{y_i=0} \log\left(\frac{e^{-X_i\cdot\theta}}{1+e^{-X_i\cdot\theta}}\right)$$

$$= \sum_i -\log\left(1+e^{-X_i\cdot\theta}\right) + \sum_{y_i=0} -X_i\cdot\theta - \lambda\sum_k \theta_k^2$$

Logistic regression

$$l_\theta(y|X) = \sum_i -\log(1 + e^{-X_i \cdot \theta}) + \sum_{y_i=0} -X_i \cdot \theta - \lambda \|\theta\|_2^2$$

$$\frac{\partial l}{\partial \theta_k} = \sum_i \frac{x_{ik} e^{-X_i \cdot \theta}}{1 + e^{-X_i \cdot \theta}} + \sum_{y_i=0} -x_{ik} - 2\lambda\theta_k$$

$$= \sum_i x_{ik}\left(1 - \sigma(X_i \cdot \theta)\right) + \sum_{y_i=0} -x_{ik} - 2\lambda\theta_k$$

Logistic regression

Log-likelihood:

$$l_\theta(y|X) = \sum_i -\log(1 + e^{-X_i \cdot \theta}) + \sum_{y_i=0} -X_i \cdot \theta - \lambda \|\theta\|_2^2$$

Derivative:

$$\frac{\partial l}{\partial \theta_k} = \sum_i X_{ik}(1 - \sigma(X_i \cdot \theta)) + \sum_{y_i=0} -X_{ik} - 2\lambda\theta_k$$

Learning Outcomes

- Introduced the logistic regression classifier
- Further studied **gradient descent** (really *ascent*) here as a means of model fitting

References

Further reading:

- On Discriminative vs. Generative classifiers: A comparison of logistic regression and naïve Bayes (Ng & Jordan '01)
- Boyd-Fletcher-Goldfarb-Shanno algorithm (BFGS)

Web Mining and Recommender Systems

Classification: Support Vector Machines

Learning Goals

- Introduce the **Support Vector Machine** classifier
- Study some of the underlying tradeoffs made by different classification approaches

So far we've seen...

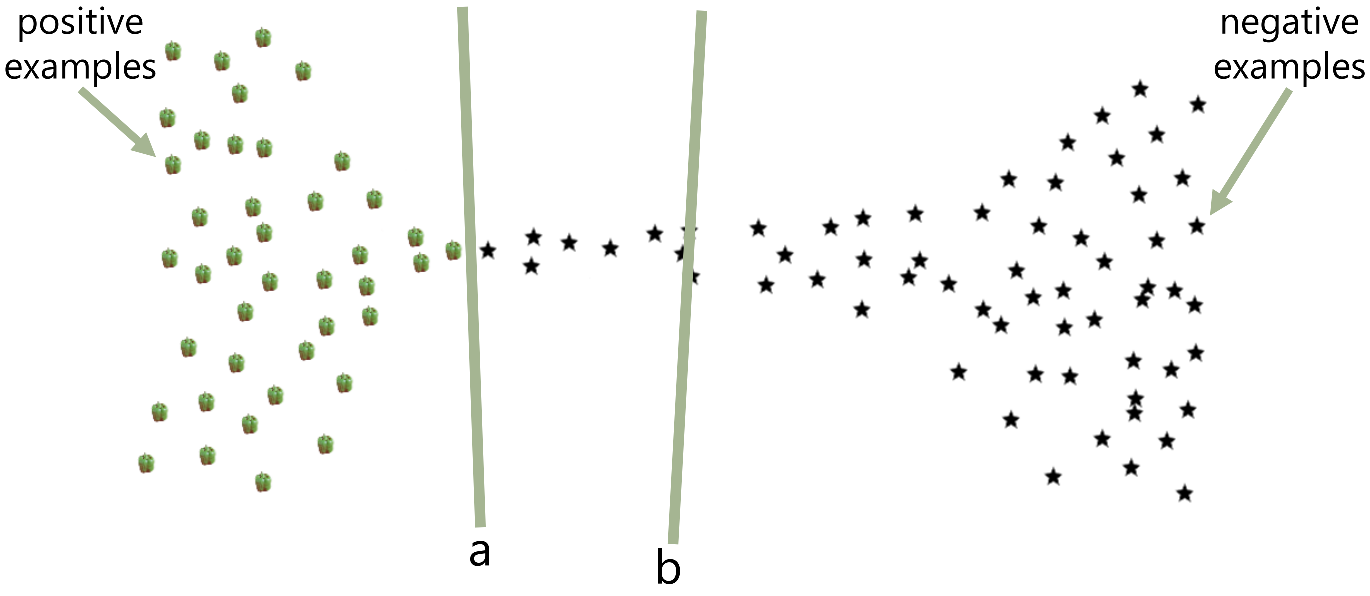
So far we've looked at **logistic regression,** which is a classification model of the form:

$$y_i = \begin{cases} 1 & \text{if } X_i \cdot \theta > 0 \\ 0 & \text{otherwise} \end{cases}$$

- In order to do so, we made certain **modeling assumptions,** but there are many different models that rely on different assumptions
  - Next we'll look at another such model

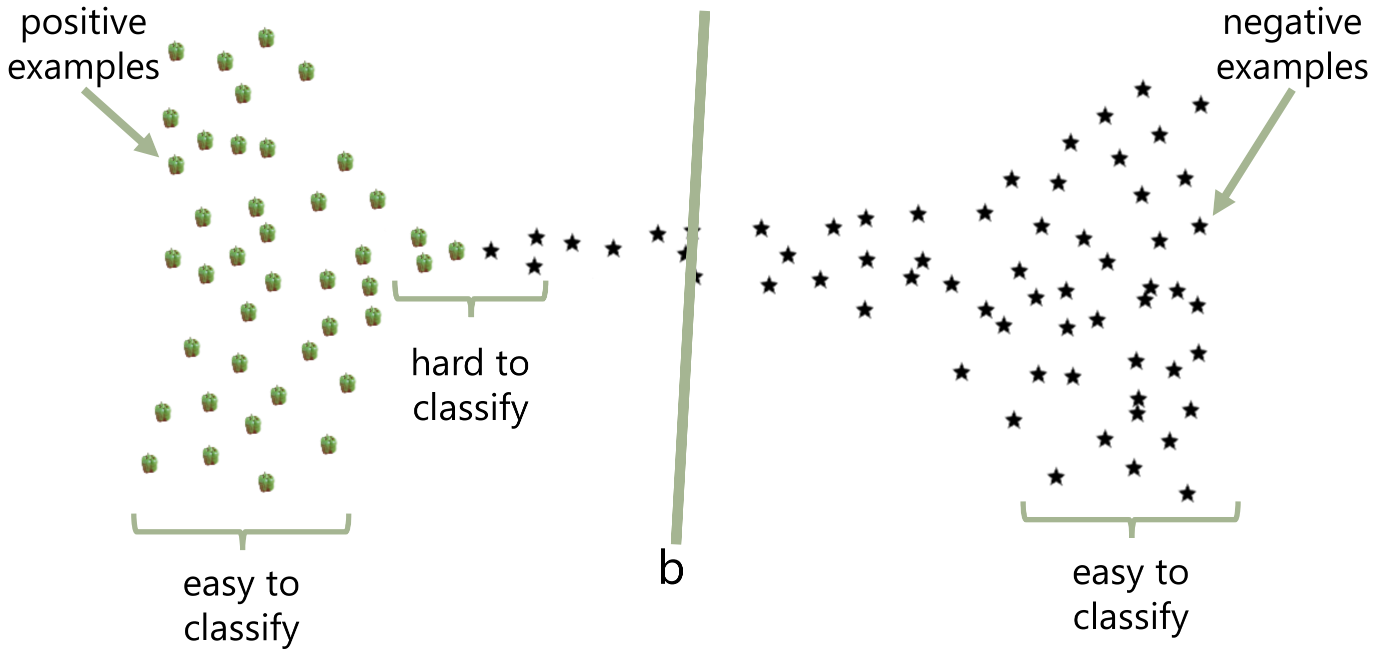
(Rough) Motivation: SVMs vs Logistic regression

Q: Where would a logistic regressor place the decision boundary for these features?



SVMs vs Logistic regression

Q: Where would a logistic regressor place the decision boundary for these features?

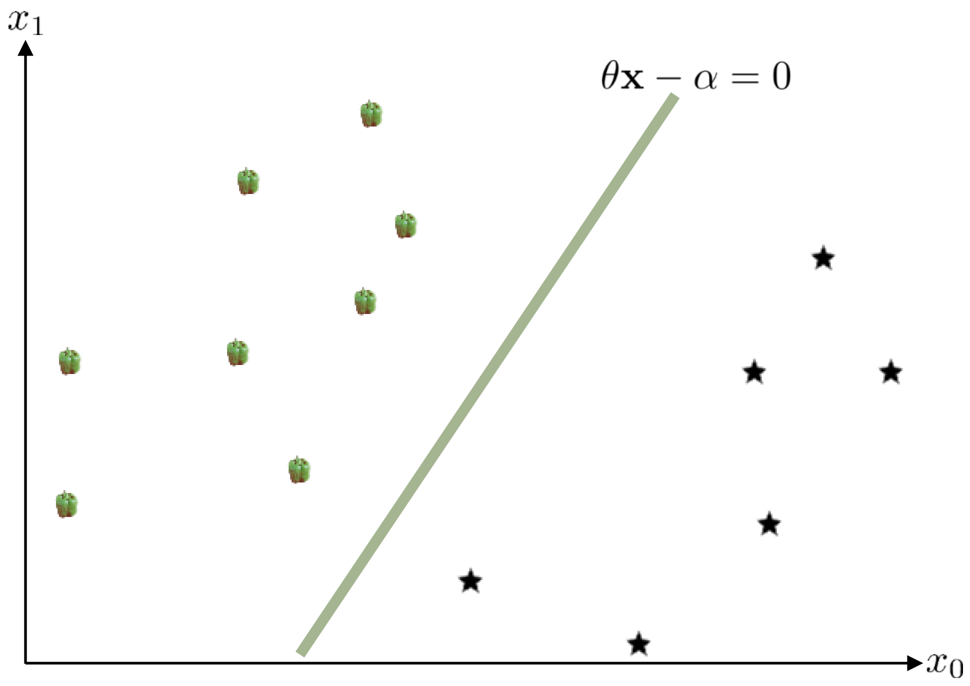


SVMs vs Logistic regression

- Logistic regressors don’t optimize the number of “mistakes”
- No special attention is paid to the “difficult” instances – every instance influences the model
- But “easy” instances can affect the model (and in a bad way!)
- How can we develop a classifier that optimizes the number of mislabeled examples?

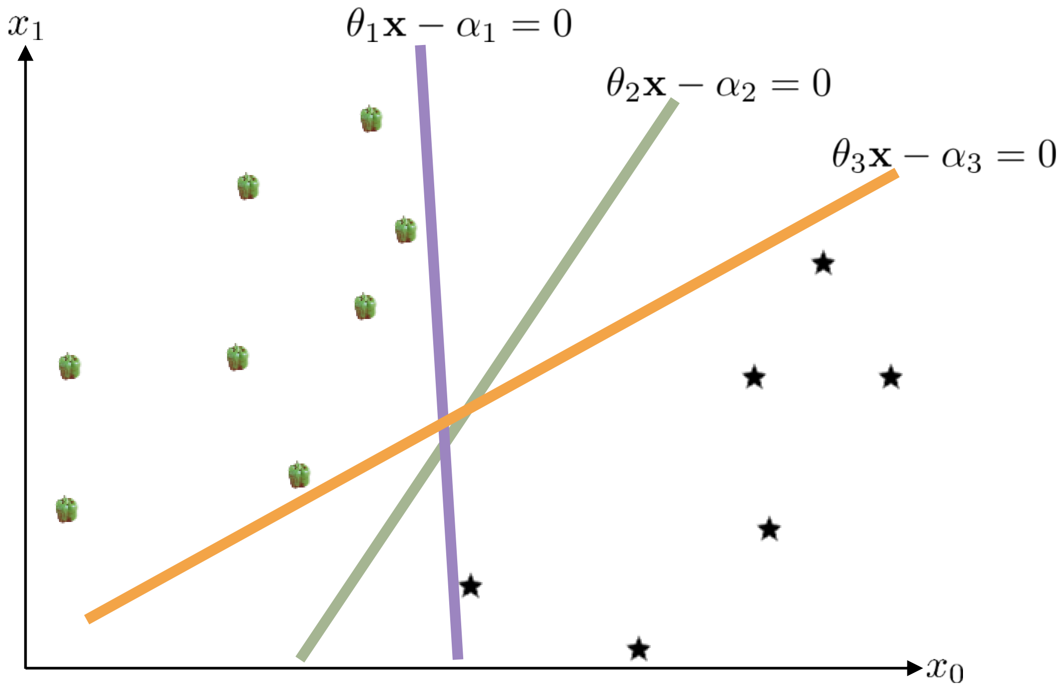
Support Vector Machines: Basic idea

A classifier can be defined by the hyperplane (line) $\theta \mathbf{x} - \alpha = 0$

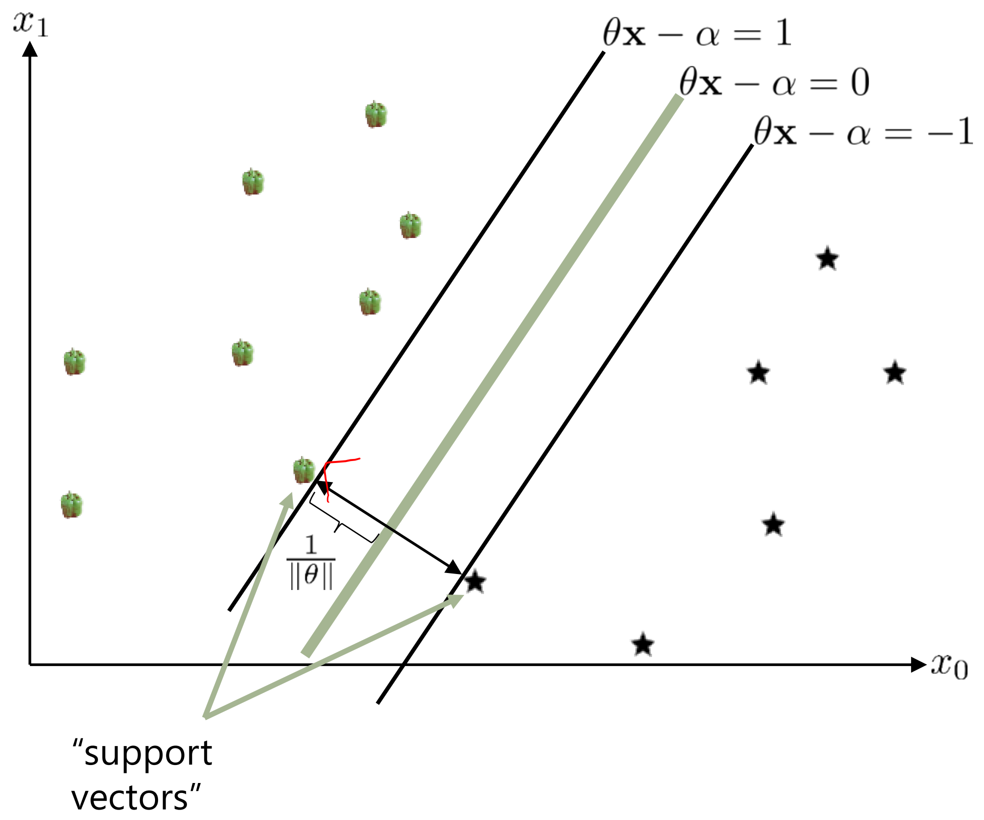


Support Vector Machines: Basic idea

Observation: Not all classifiers are equally good



Support Vector Machines



- An SVM seeks the classifier (in this case a line) that is **furthest from the nearest points**
- This can be written in terms of a specific optimization problem:

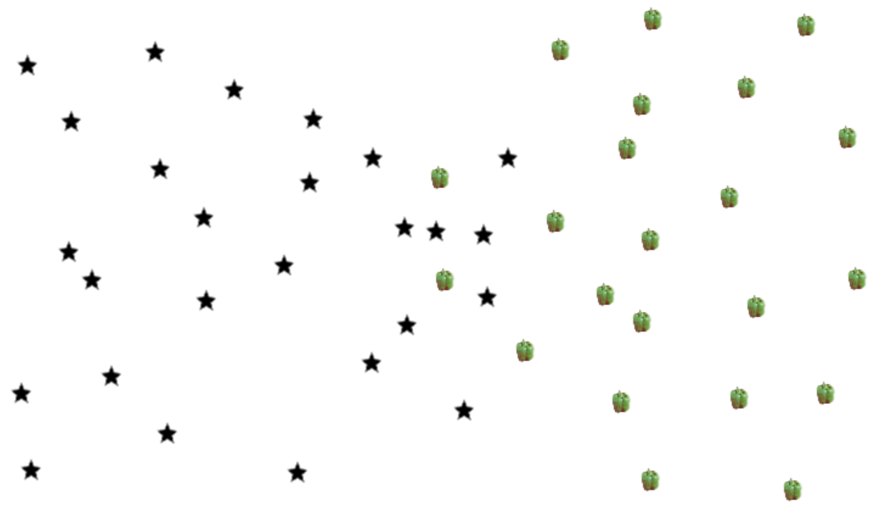
$$\arg\min_{\theta,\alpha} \frac{1}{2}||\theta||_2^2$$

such that

$$\forall_i y_i(\theta \cdot X_i - \alpha) \geq 1$$

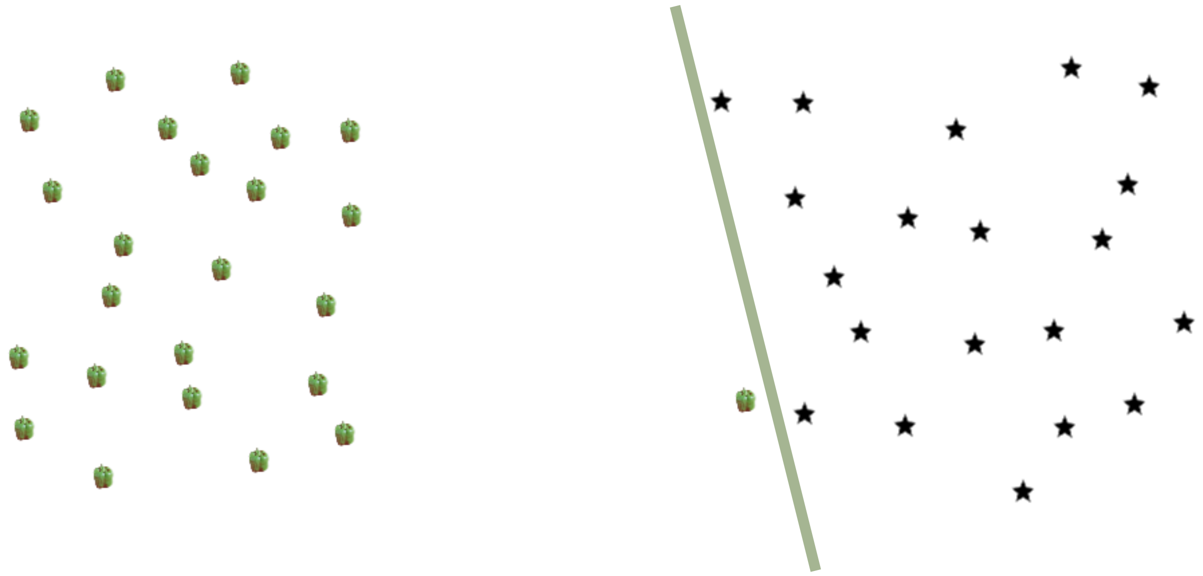
Support Vector Machines

But: is finding such a separating hyperplane even possible?



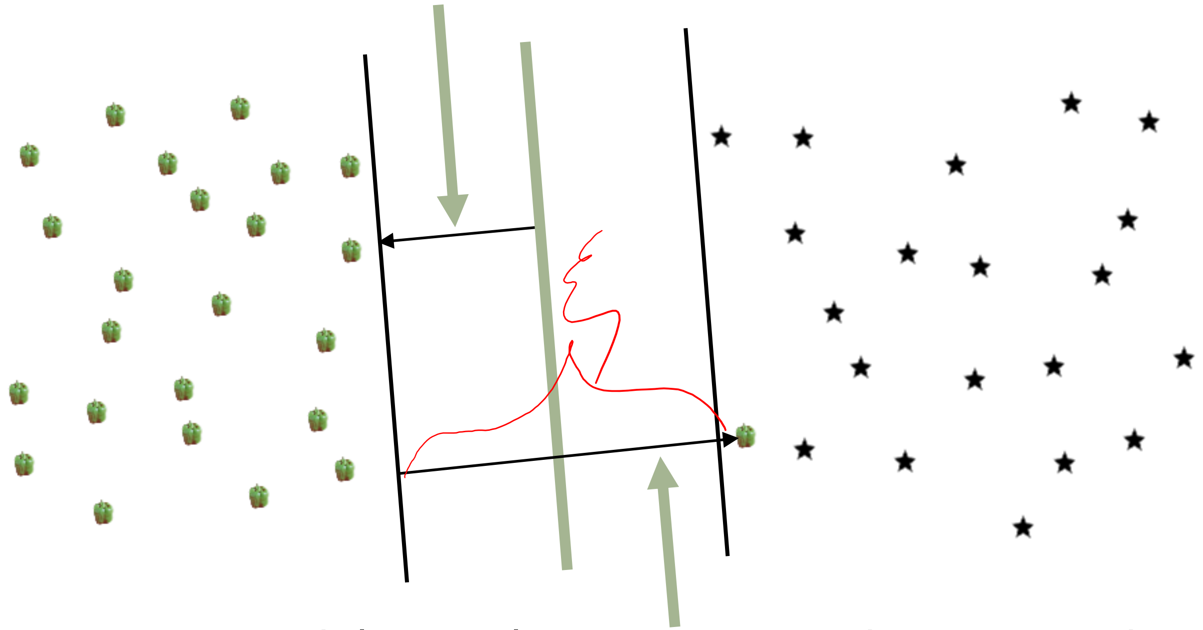
Support Vector Machines

Or: is it actually a good idea?



Support Vector Machines

Want the margin to be as wide as possible



While penalizing points on the wrong side of it

Support Vector Machines

Soft-margin formulation:

$$\boxed{\begin{array}{c} \arg\min_{\theta,\alpha,\xi>0} \frac{1}{2}\|\theta\|_2^2 + C\sum_i \xi_i \\[2em] \text{such that} \\[2em] \forall_i y_i(\theta \cdot X_i - \alpha) \geq 1 - \xi_i \end{array}}$$

Summary of Support Vector Machines

- SVMs seek to find a hyperplane (in two dimensions, a line) that optimally separates two classes of points
- The "best" classifier is the one that classifies all points correctly, such that the nearest points are **as far as possible** from the boundary
- If not all points can be correctly classified, a penalty is incurred that is proportional to **how badly the points are misclassified** (i.e., their distance from this hyperplane)

Learning Outcomes

- Introduced a different type of classifier that seeks to minimize the number of mistakes made more directly

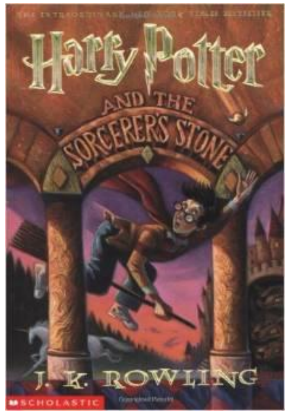
Web Mining and Recommender Systems

Classification – Worked example

Learning Goals

- Work through a simple example of classification
- Introduce some of the difficulties in evaluating classifiers

Judging a book by its cover



[0.723845, 0.153926, 0.757238, 0.983643, …]

$\underbrace{\hspace{10em}}_{\text{4096-dimensional image features}}$

Images features are available for each book on
http://cseweb.ucsd.edu/classes/fa19/cse258-a/data/book_images_5000.json

http://caffe.berkeleyvision.org/

Judging a book by its cover

Example: train a classifier to predict whether a book is a children's book from its cover art

(code available on course webpage)

Judging a book by its cover

- The number of errors we made was extremely low, yet our classifier doesn’t seem to be very good – why?
(stay tuned!)

Web Mining and Recommender Systems

Classifiers: Summary

Learning Goals

- Summarize some of the differences between each of the classification schemes we have seen

Previously...

How can we predict **binary** or **categorical** variables?

$$f(\text{data}) \xrightarrow{?} \text{labels}$$

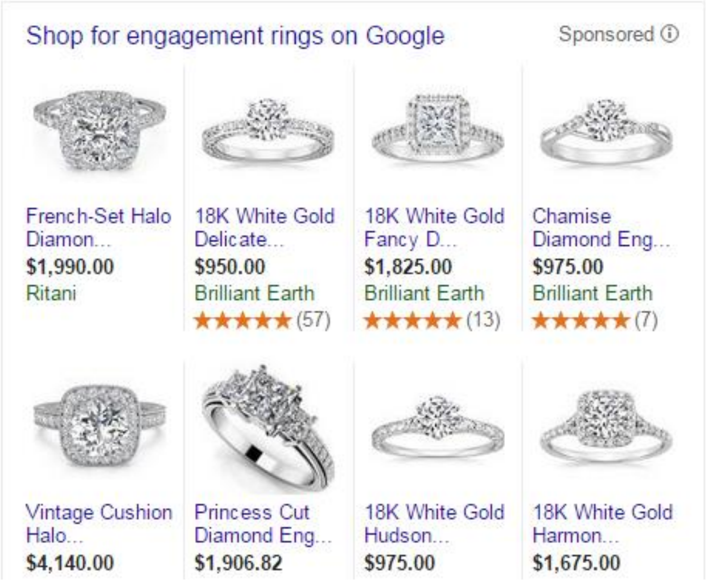
{0,1}, {True, False}

{1, ... , N}

Previously...

Will I **purchase** this product?

(yes)



Will I **click on** this ad?

(no)

Previously...

- **Naïve Bayes**
  - Probabilistic model (fits $p(label|data)$)
  - Makes a conditional independence assumption of the form $(feature_i \perp\!\!\!\perp feature_j | label)$ allowing us to define the model by computing $p(feature_i | label)$ for each feature
  - Simple to compute just by counting
- **Logistic Regression**
  - Fixes the "double counting" problem present in naïve Bayes
- **SVMs**
  - Non-probabilistic: optimizes the classification error rather than the likelihood

1) Naïve Bayes

posterior   prior   likelihood

$$\underbrace{p(label|features)}_{\text{posterior}} = \frac{\overbrace{p(label)}^{\text{prior}}\,\overbrace{p(features|label)}^{\text{likelihood}}}{\underbrace{p(features)}_{\text{evidence}}}$$

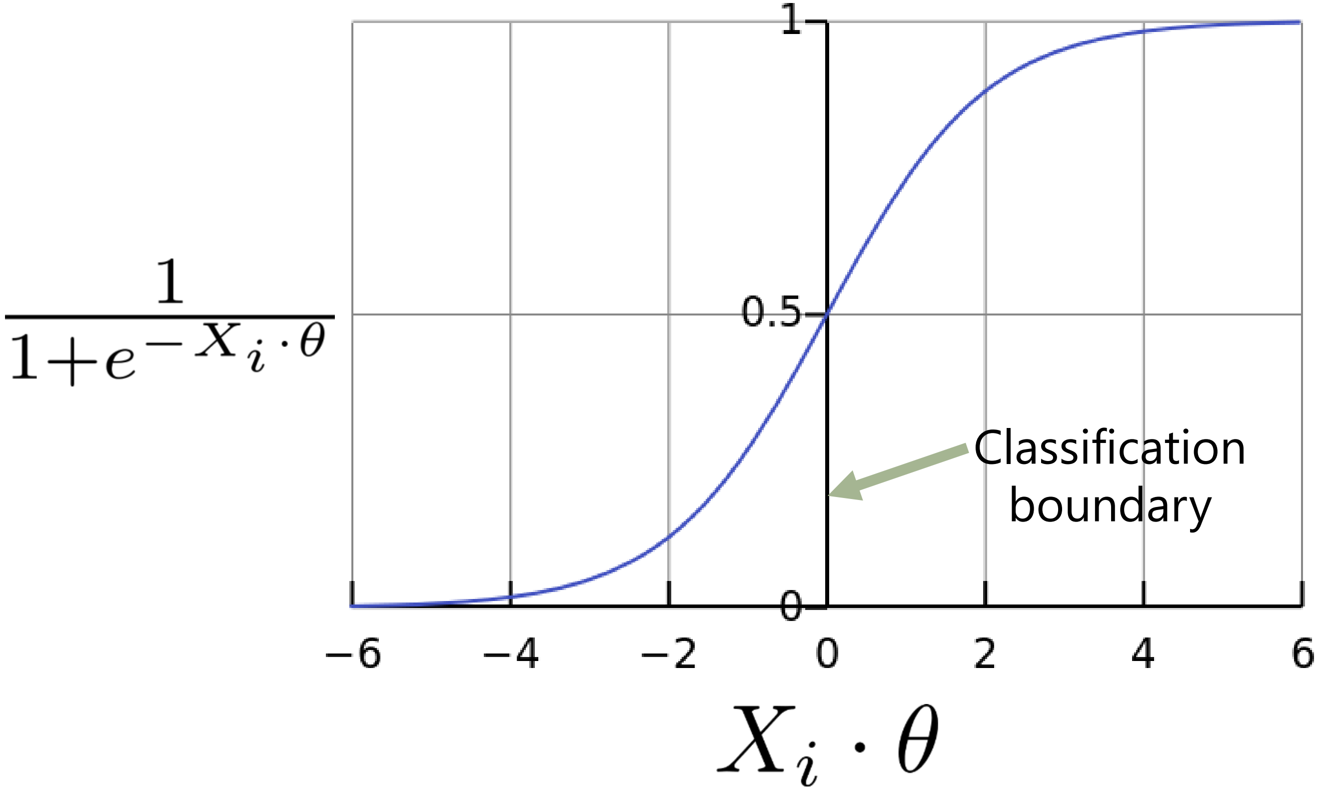
evidence

due to our conditional independence assumption:

$$p(label|features) = \frac{p(label)\prod_i p(feature_i|label)}{p(features)}$$

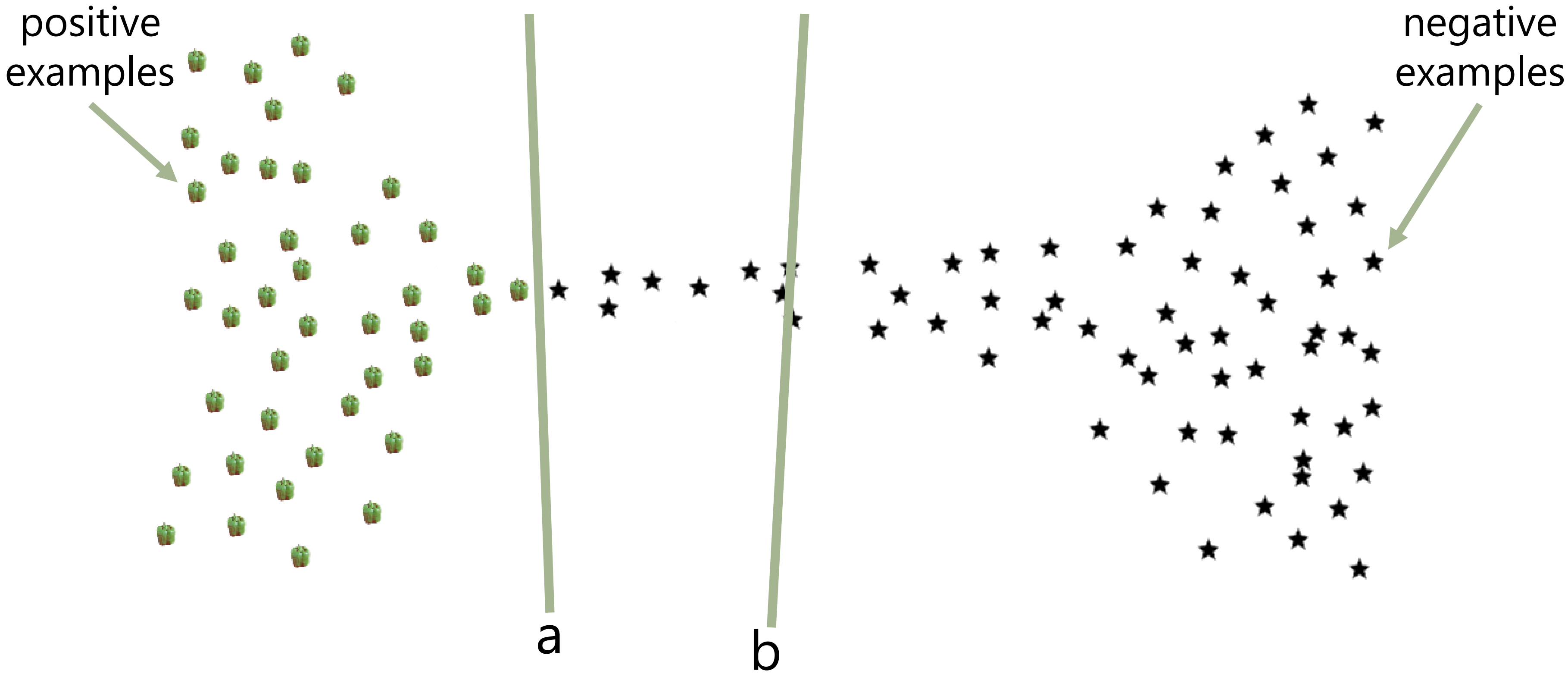
2) logistic regression

sigmoid function: $\sigma(t) = \frac{1}{1+e^{-t}}$



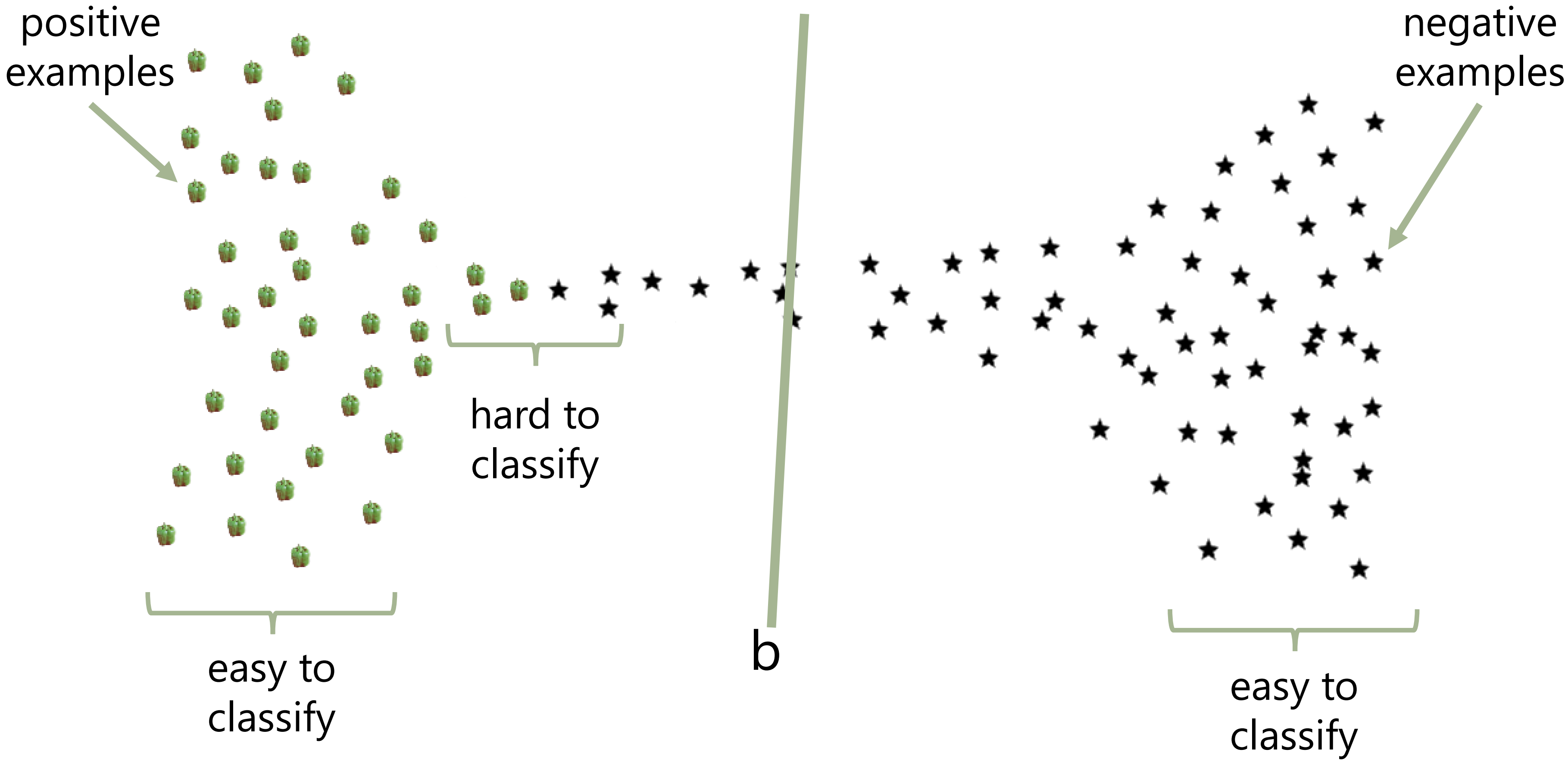
Logistic regression

Q: Where would a logistic regressor place the decision boundary for these features?



Logistic regression

Q: Where would a logistic regressor place the decision boundary for these features?

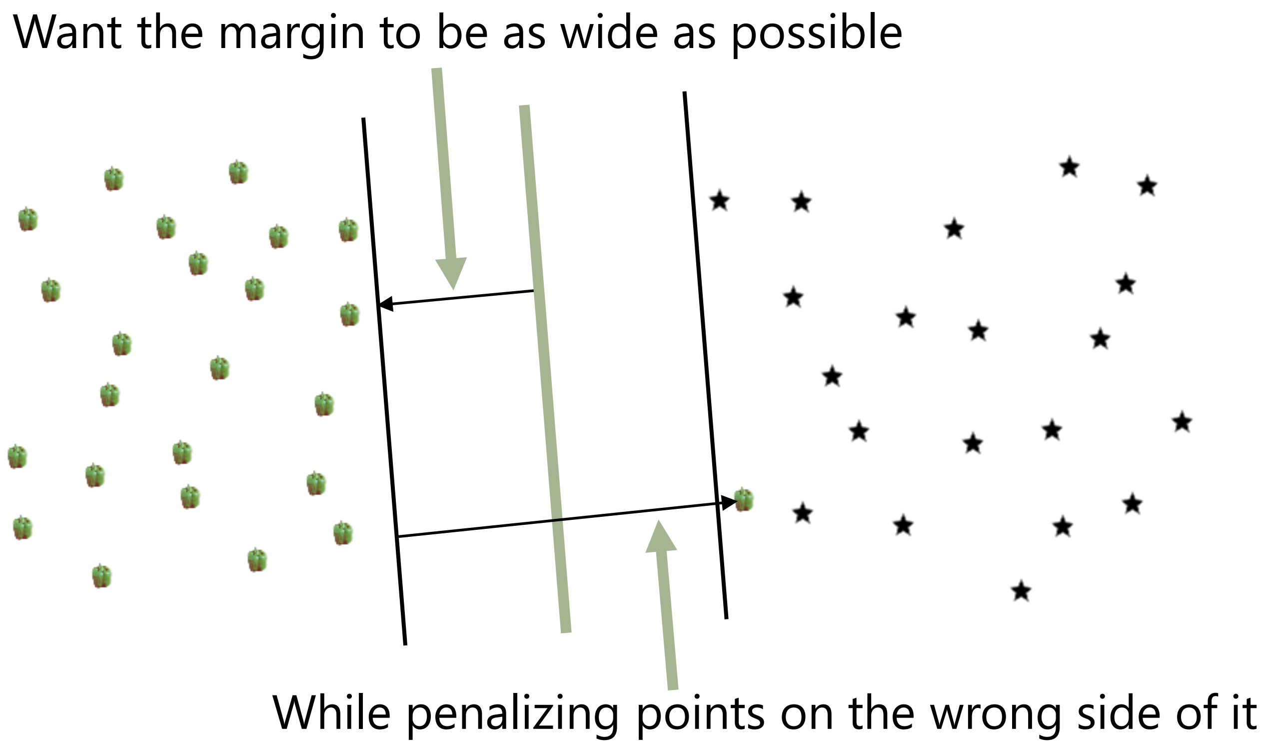


Logistic regression

- Logistic regressors don’t optimize the number of “mistakes”
- No special attention is paid to the “difficult” instances – every instance influences the model
- But “easy” instances can affect the model (and in a bad way!)
- How can we develop a classifier that optimizes the number of mislabeled examples?

3) Support Vector Machines

Can we train a classifier that optimizes the **number of mistakes,** rather than maximizing a probability?



Pros/cons

- **Naïve Bayes**

  ++ Easiest to implement, most efficient to “train”
  ++ If we have a process that generates feature that *are* independent given the label, it’s a very sensible idea
  -- Otherwise it suffers from a “double-counting” issue

- **Logistic Regression**

  ++ Fixes the “double counting” problem present in naïve Bayes
  -- More expensive to train

- **SVMs**

  ++ Non-probabilistic: optimizes the classification error rather than the likelihood
  -- More expensive to train

Summary

- **Naïve Bayes**
  - Probabilistic model (fits $p(label|data)$)
  - Makes a conditional independence assumption of the form $(feature_i \perp\!\!\!\perp feature_j|label)$ allowing us to define the model by computing $p(feature_i|label)$ for each feature
  - Simple to compute just by counting
- **Logistic Regression**
  - Fixes the "double counting" problem present in naïve Bayes
- **SVMs**
  - Non-probabilistic: optimizes the classification error rather than the likelihood

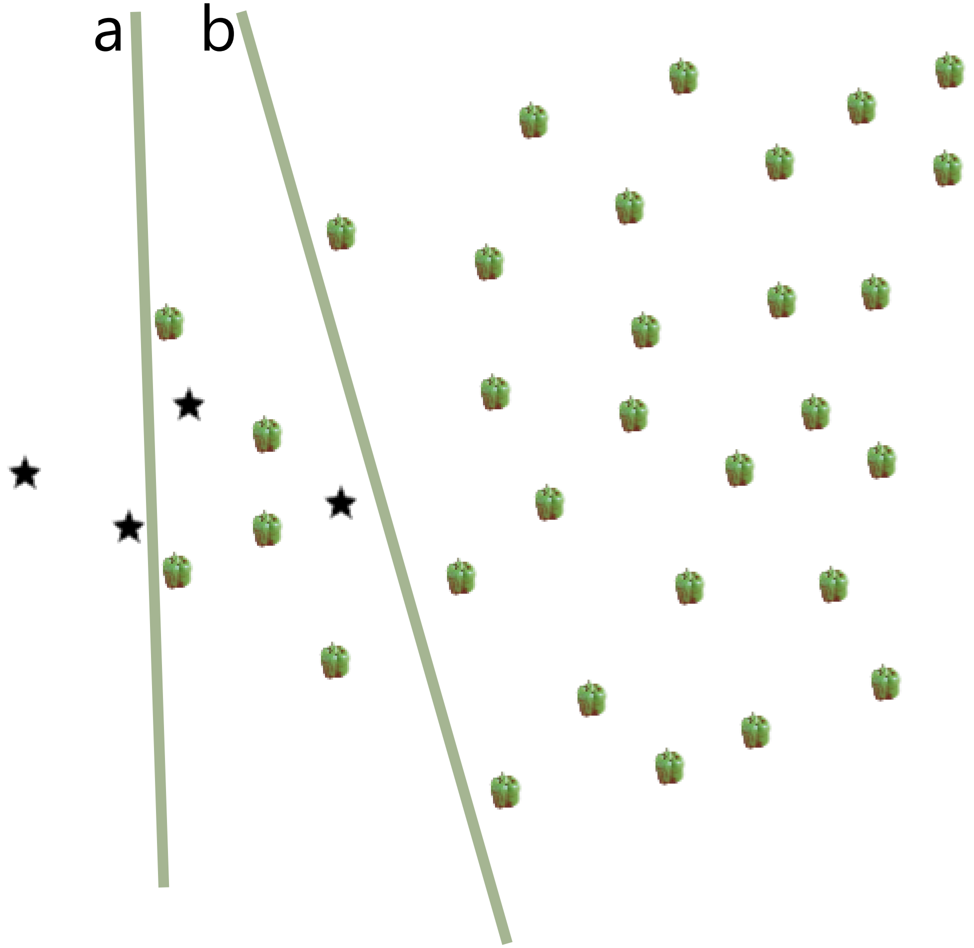
Web Mining and Recommender Systems

Evaluating classifiers

Learning Goals

- Discuss several schemes for evaluating classifiers under different conditions

Which of these classifiers is best?



\# mistakes

a = 2

b = 5

Which of these classifiers is best?

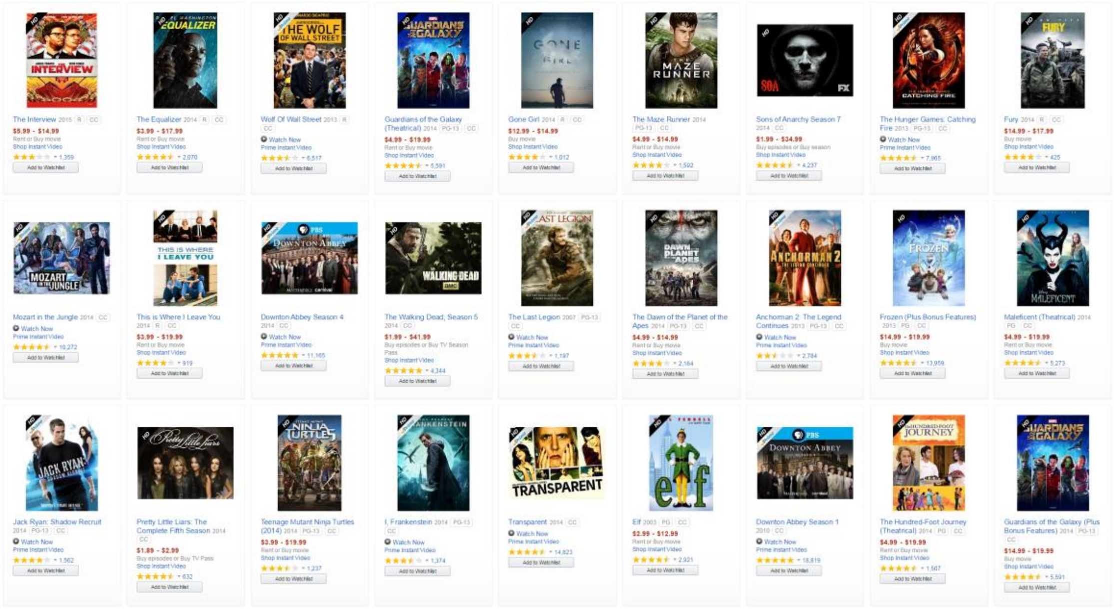
The solution which minimizes the
#errors may not be the best one

Which of these classifiers is best?

1. When data are highly imbalanced

If there are far fewer positive examples than negative examples we may want to assign additional weight to negative instances (or vice versa)

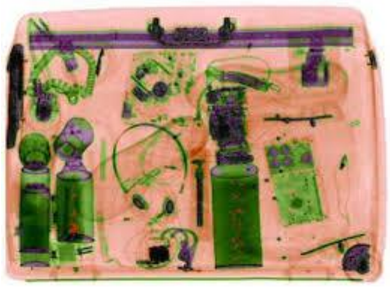
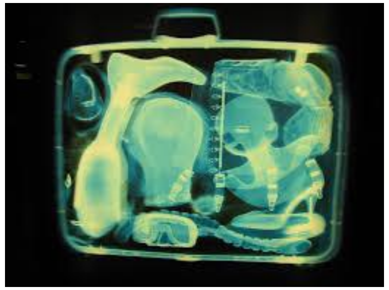
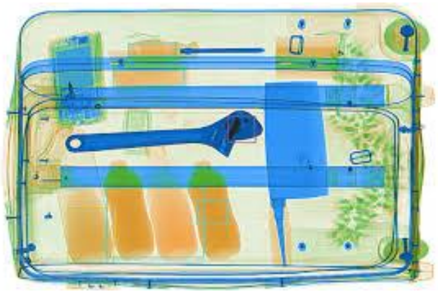
e.g. will I purchase a product? If I purchase 0.00001% of products, then a classifier which just predicts "no" everywhere is 99.99999% accurate, but not very useful



Which of these classifiers is best?

2. When mistakes are more costly in one direction

False positives are nuisances but false negatives are disastrous (or vice versa)

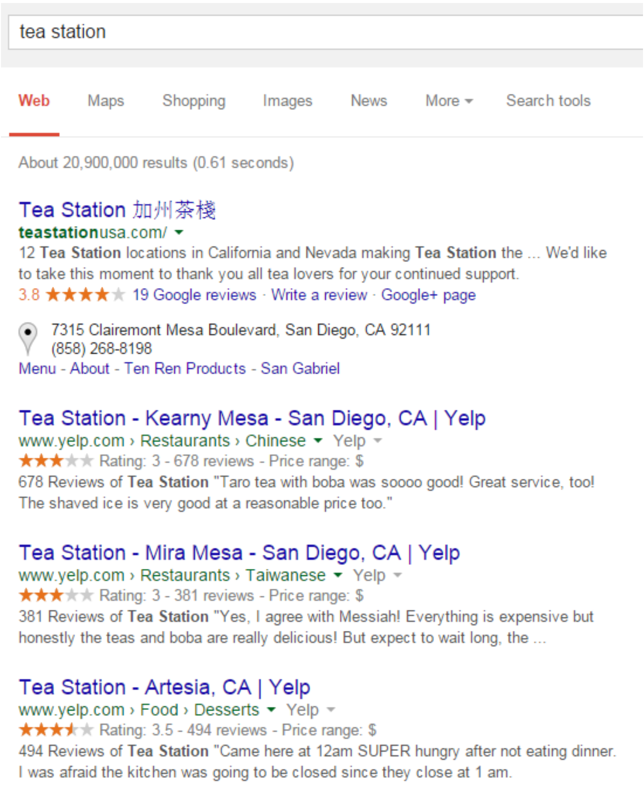





e.g. which of these bags contains a weapon?

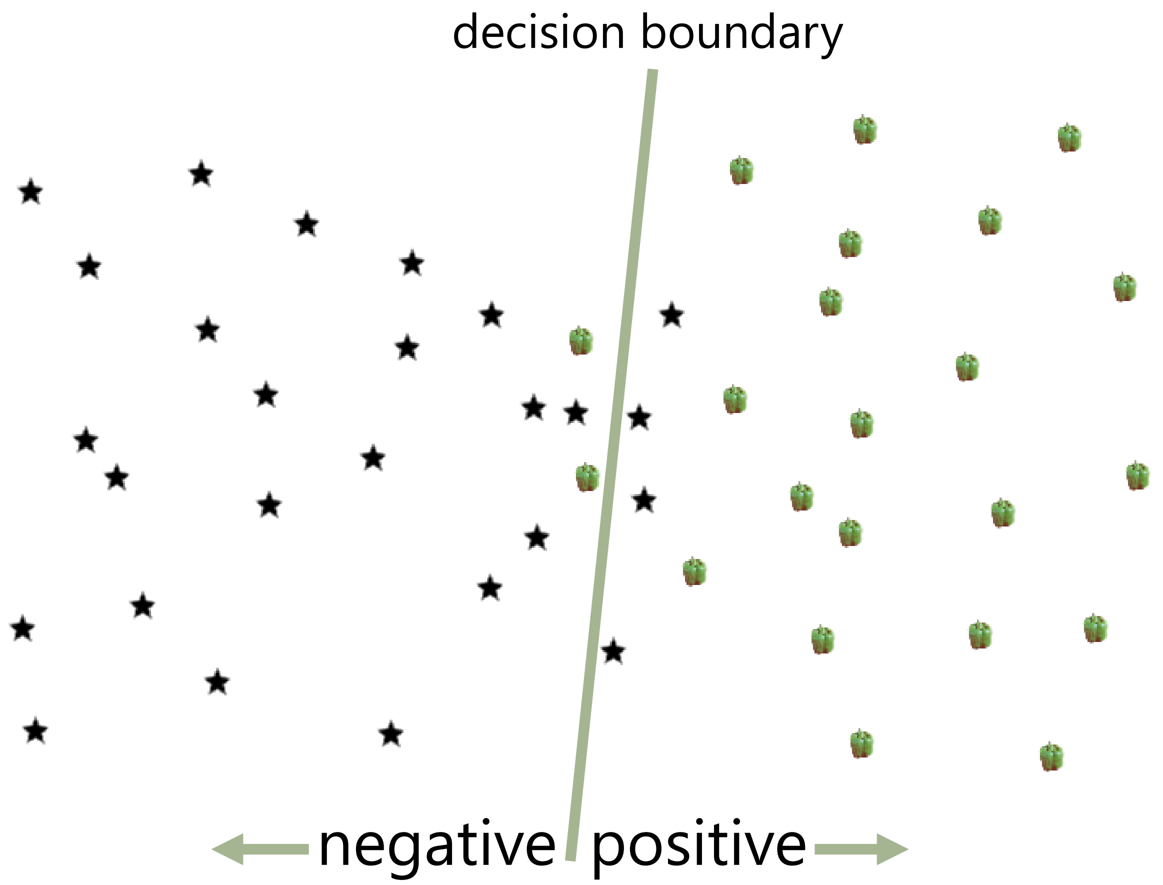
Which of these classifiers is best?

3. When we only care about the "most confident" predictions

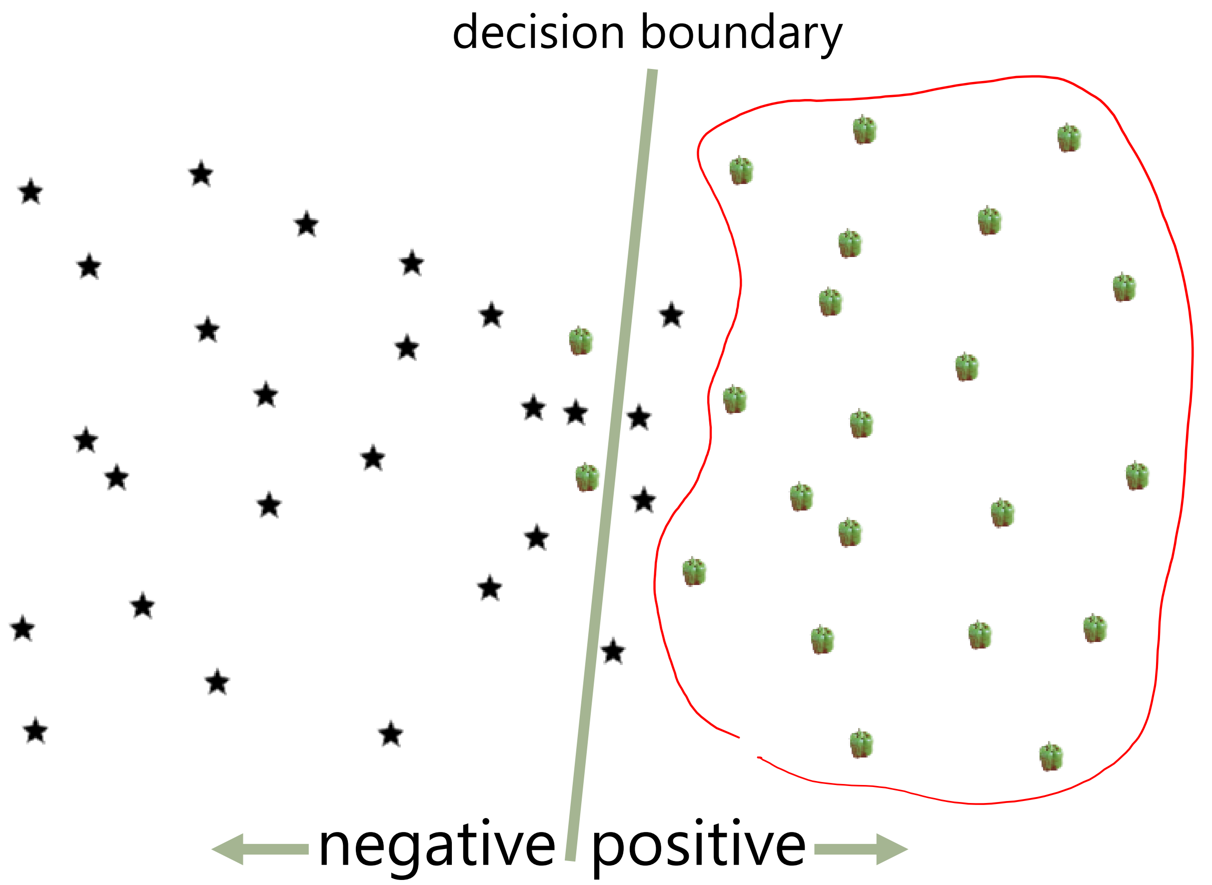
e.g. does a relevant result appear among the first page of results?



Evaluating classifiers

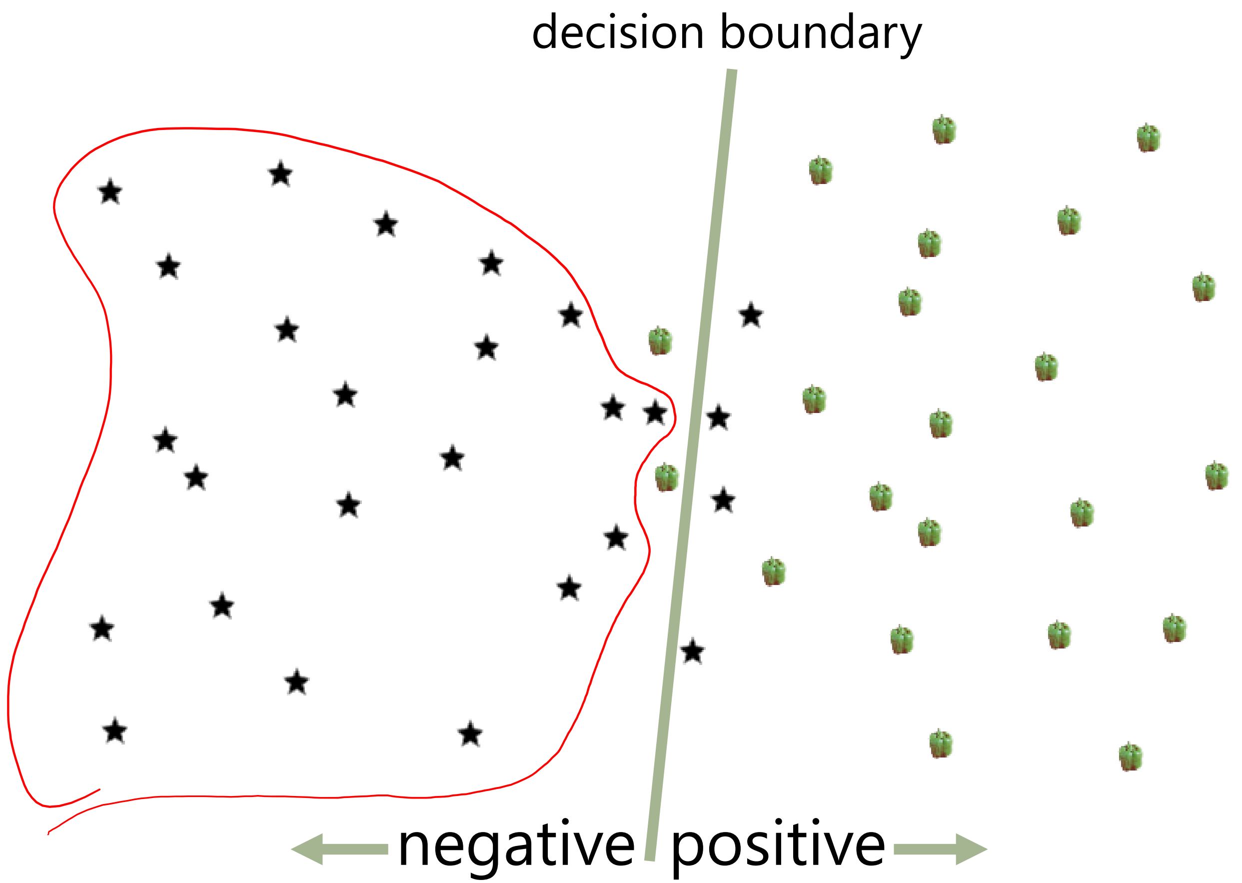


Evaluating classifiers



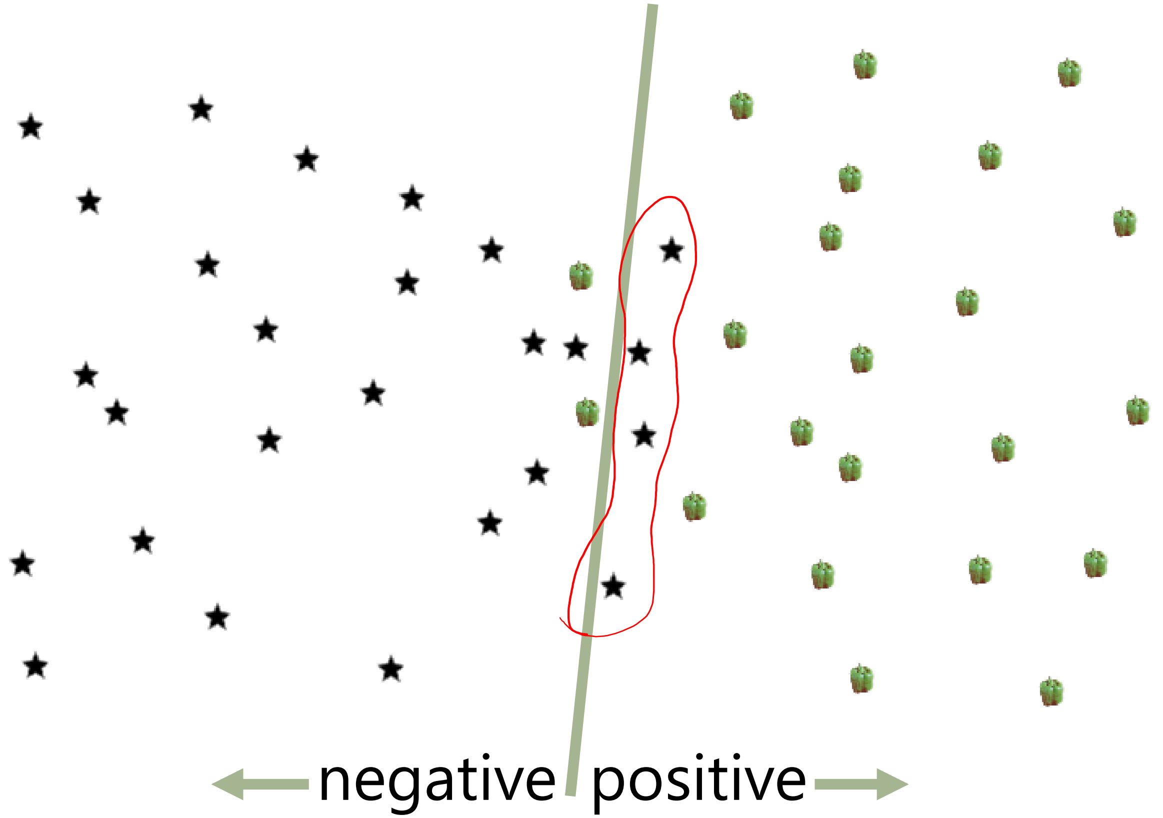
TP (true positive): Labeled as T , predicted as T

Evaluating classifiers



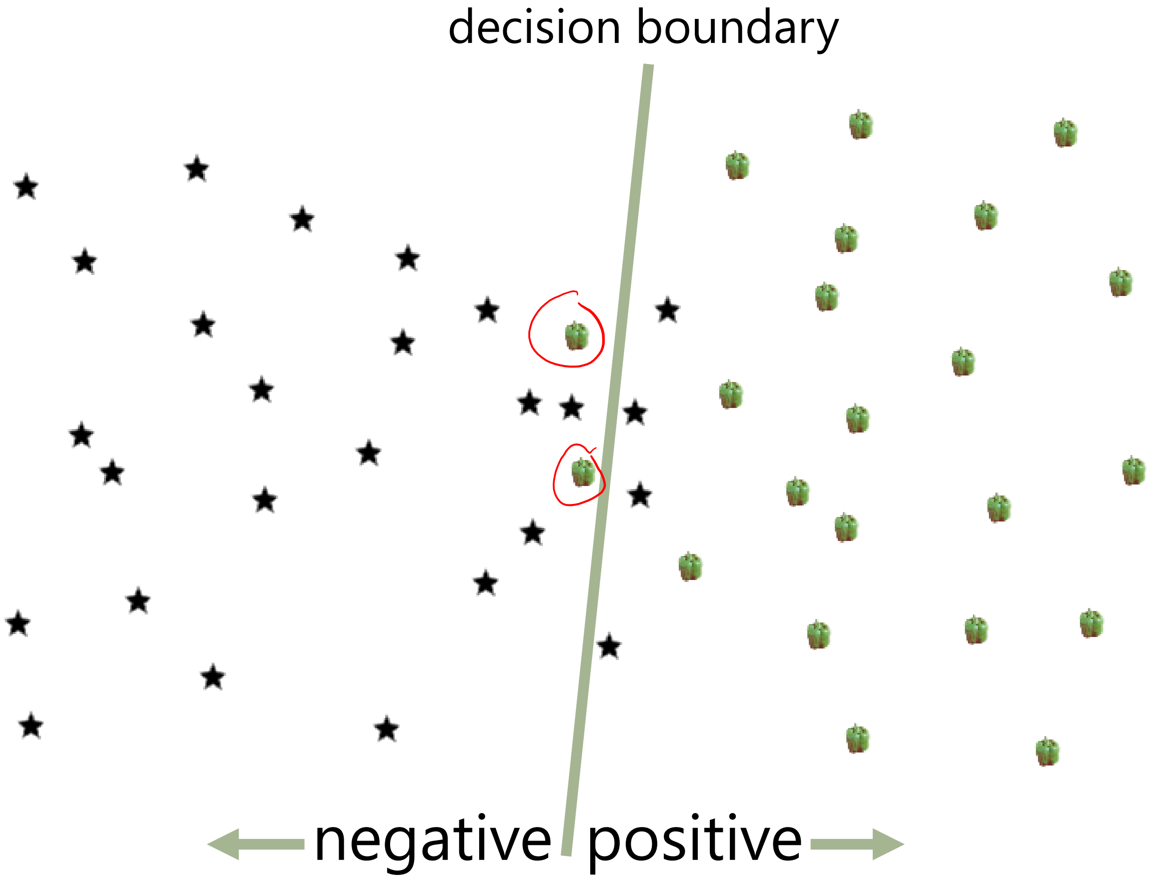
TN (true negative): Labeled as F, predicted as F

Evaluating classifiers



FP (false positive): Labeled as F, predicted as T

Evaluating classifiers



FN (false negative): Labeled as T , predicted as F

Evaluating classifiers

		Label	
		true	false
Prediction	true	true positive	false positive
	false	false negative	true negative

Classification accuracy = correct predictions / #predictions

$= (TP + TN) / (TP + TN + FP + FN)$

Error rate = incorrect predictions / #predictions

$= (FP + FN) / (TP + TN + FP + FN)$

Evaluating classifiers

		Label: true	Label: false
Prediction	true	true positive	false positive
	false	false negative	true negative

True positive rate (**TPR**) = true positives / #labeled positive

= TP/(TP+FN)

True negative rate (**TNR**) = true negatives / #labeled negative

= TN/(TN+FP)

Evaluating classifiers

		Label	
		true	false
Prediction	true	true positive	false positive
	false	false negative	true negative

Balanced Error Rate (BER) = ½ (FPR + FNR)

= ½ for a random/naïve classifier, 0 for a perfect classifier

$= 1 - \frac{1}{2}(TPR + TNR)$

Evaluating classifiers

e.g.

```
         y = [  1,  -1,   1,   1,  1, -1,  1,  1,  -1,  1]
Confidence = [1.3,-0.2,-0.1,-0.4,1.4,0.1,0.8,0.6,-0.8,1.0]
```

$\checkmark$ TP TN FN FN TP FP TP TP TN TP

$X_i \cdot \theta$

TP = 5 TN = 2

FP = 1 FN = 2

$TPR = 5/7$ $TNR = 2/3$

$$BER = 1 - \frac{1}{2}\left(\frac{5}{7} + \frac{2}{3}\right)$$

Evaluating classifiers

How to optimize a balanced error measure:

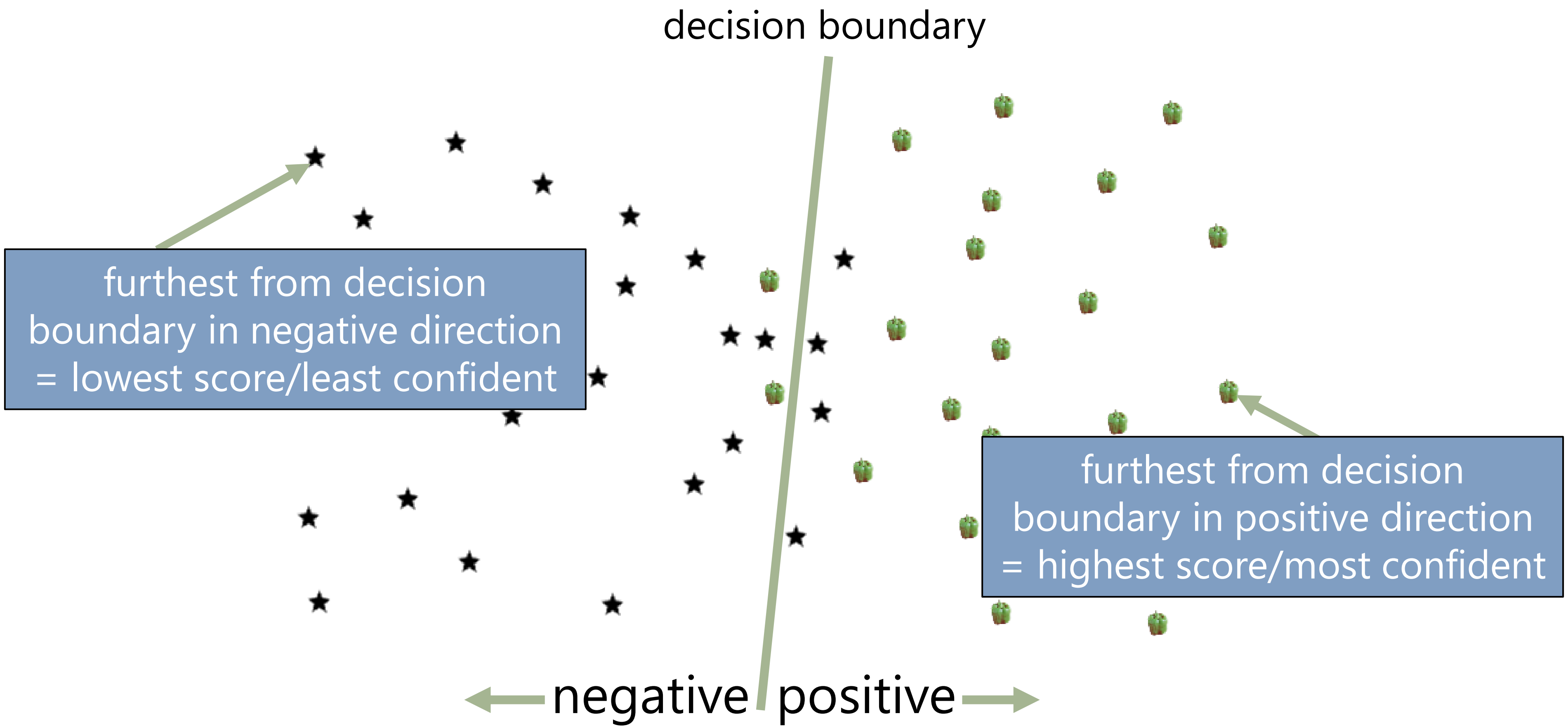
$$L_\theta(y|X) = \prod_{y_i=1} p_\theta(y_i|X_i) \prod_{y_i=0} (1 - p_\theta(y_i|X_i))$$

$$\ell_\theta = \frac{N}{2|y_i=1|} \sum_{y_i=1} \log \sigma(X_i \cdot \theta)$$

$$+ \frac{N}{2|y_i=0|} \sum_{y_i=0} \log(1 - \sigma(X_i \cdot \theta))$$

Evaluating classifiers – ranking

The classifiers we've seen can associate **scores** with each prediction



Evaluating classifiers – ranking

The classifiers we've seen can associate **scores** with each prediction

- In ranking settings, the actual labels assigned to the points (i.e., which side of the decision boundary they lie on) **don't matter**
- All that matters is that positively labeled points tend to be at **higher ranks** than negative ones

Evaluating classifiers – ranking

The classifiers we've seen can associate **scores** with each prediction

- For naïve Bayes, the "score" is the ratio between an item having a positive or negative class
- For logistic regression, the "score" is just the probability associated with the label being 1
- For Support Vector Machines, the score is the distance of the item from the decision boundary (together with the sign indicating what side it's on)

Evaluating classifiers – ranking

The classifiers we've seen can associate **scores** with each prediction

e.g.

y = `[  1,  -1,   1,   1,  1, -1,  1,  1,  -1,  1]`
Confidence = `[1.3,-0.2,-0.1,-0.4,1.4,0.1,0.8,0.6,-0.8,1.0]`

Sort **both** according to confidence:

[1 1 1 1 1 -1 1 -1 1 -1]
[1.4, 1.3, 1.0, 0.8, 0.6, 0.1, -0.1, -0.2, -0.4, -0.8]

Evaluating classifiers – ranking

The classifiers we've seen can associate **scores** with each prediction

Labels sorted by confidence:

$$\underbrace{[1, 1, 1, 1, 1, -1,}_{}\ 1, -1, 1, -1]$$

Suppose we have a fixed budget (say, six) of items that we can return
(e.g. we have space for six results in an interface)

- Total number of **relevant** items = 7
- Number of items we returned = 6
- Number of **relevant items** we returned = 5

Evaluating classifiers – ranking

The classifiers we've seen can associate **scores** with each prediction

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

"fraction of retrieved documents that are relevant"

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

"fraction of relevant documents that were retrieved"

Evaluating classifiers – ranking

The classifiers we've seen can associate **scores** with each prediction

$\text{precision}@k$ = precision when we have a budget of *k* retrieved documents

e.g.

- Total number of **relevant** items = 7
- Number of items we returned = 6
- Number of **relevant items** we returned = 5

precision@6 = $\frac{5}{6}$

Evaluating classifiers – ranking

The classifiers we've seen can
associate **scores** with each prediction

$$F_1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

(harmonic mean of precision and recall)

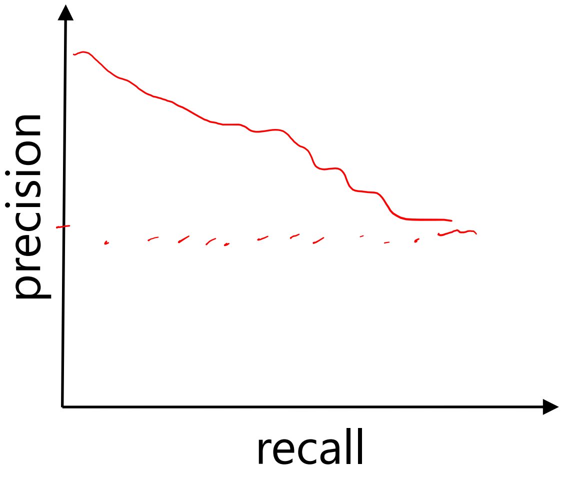
$$F_\beta = (1 + \beta^2) \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\beta^2 \mathrm{precision} + \mathrm{recall}}$$

(weighted, in case precision is more important
(low beta), or recall is more important (high beta))

Precision/recall curves

How does our classifier behave as we "increase the budget" of the number retrieved items?

- For budgets of size 1 to N, compute the precision and recall
- Plot the precision against the recall



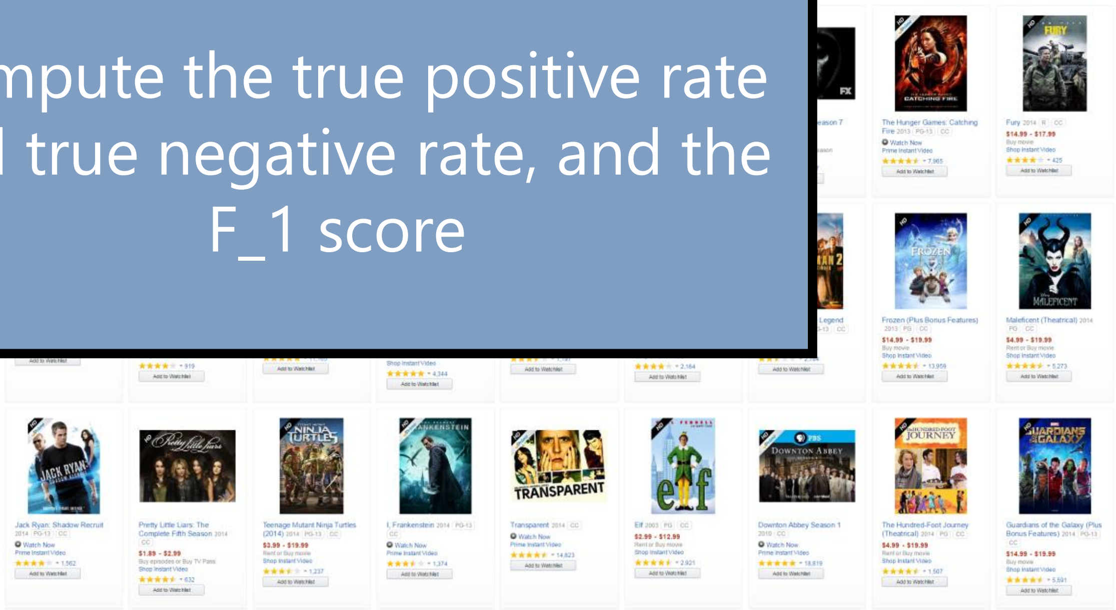
Summary

1. When data are highly imbalanced

If there are far fewer positive examples than negative examples we may want to assign additional weight to negative instances (or vice versa)

Compute the true positive rate and true negative rate, and the F_1 score

e.g. will I purchase
product? If I
purchase 0.00001
of products, then a
classifier which jus
predicts "no"
everywhere is
99.99999% accurate,
but not very useful

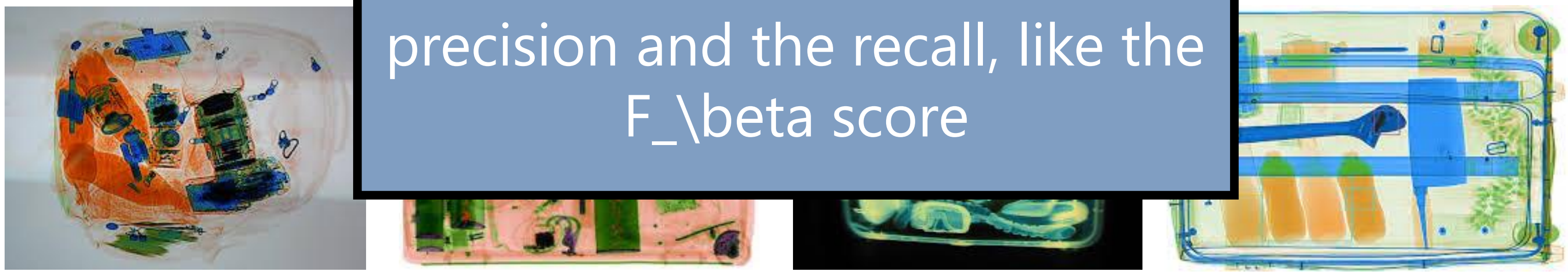


Summary

2. When mistakes are more costly in one direction

False positives are nuisances but false negatives are disastrous (or vice versa)

Compute “weighted” error measures that trade-off the precision and the recall, like the F_\beta score



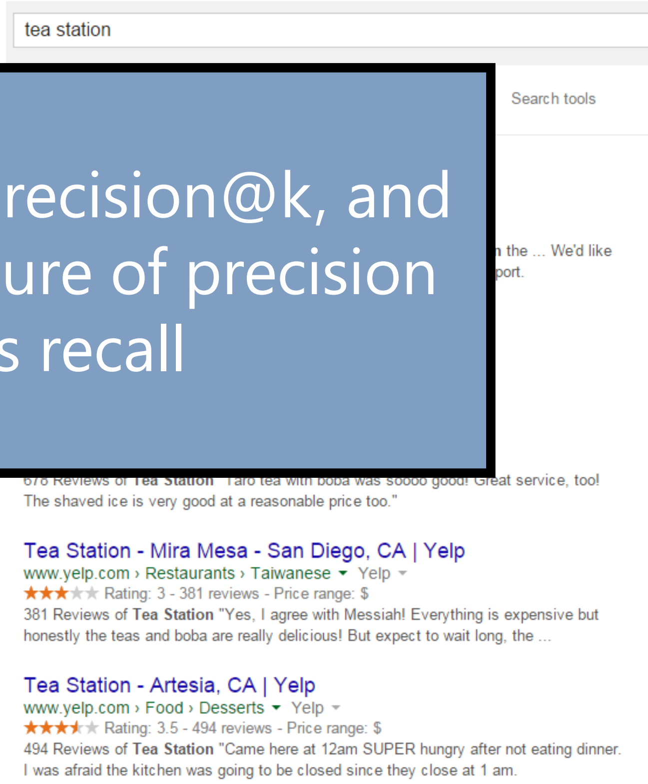
e.g. which of these bags contains a weapon?

Summary

3. When we only care about the "most confident" predictions

e.g. does
result
among
page of results?

Compute the precision@k, and plot the signature of precision versus recall



Learning Outcomes

- Saw several examples of classification evaluation measures
- Introduced the F-score, precision and recall, and Balanced Error Rate (among others)

Web Mining and Recommender Systems

Classifier Evaluation: Worked Example

Learning Goals

- Implement the evaluation metrics from the previous section on real data

Code example: bankruptcy data

We'll look at a simple dataset from the UCI repository:
https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data

```
@relation '5year-weka.filters.unsupervised.instance.SubsetByExpression-Enot ismissing(ATT20)'

@attribute Attr1 numeric
@attribute Attr2 numeric
...
@attribute Attr63 numeric
@attribute Attr64 numeric
@attribute class {0,1}

@data
0.088238,0.55472,0.01134,1.0205,-
66.52,0.34204,0.10949,0.57752,1.0881,0.32036,0.10949,0.1976,0.096885,0.10949,1475.2,0.24742,1.8027,0.10949,0.077287,50.199,
1.1574,0.13523,0.062287,0.41949,0.32036,0.20912,1.0387,0.026093,6.1267,0.37788,0.077287,155.33,2.3498,0.24377,0.13523,1.449
3,571.37,0.32101,0.095457,0.12879,0.11189,0.095457,127.3,77.096,0.45289,0.66883,54.621,0.10746,0.075859,1.0193,0.55407,0.42
557,0.73717,0.73866,15182,0.080955,0.27543,0.91905,0.002024,7.2711,4.7343,142.76,2.5568,3.2597,0
```

Did the company go bankrupt?

Code on course webpage

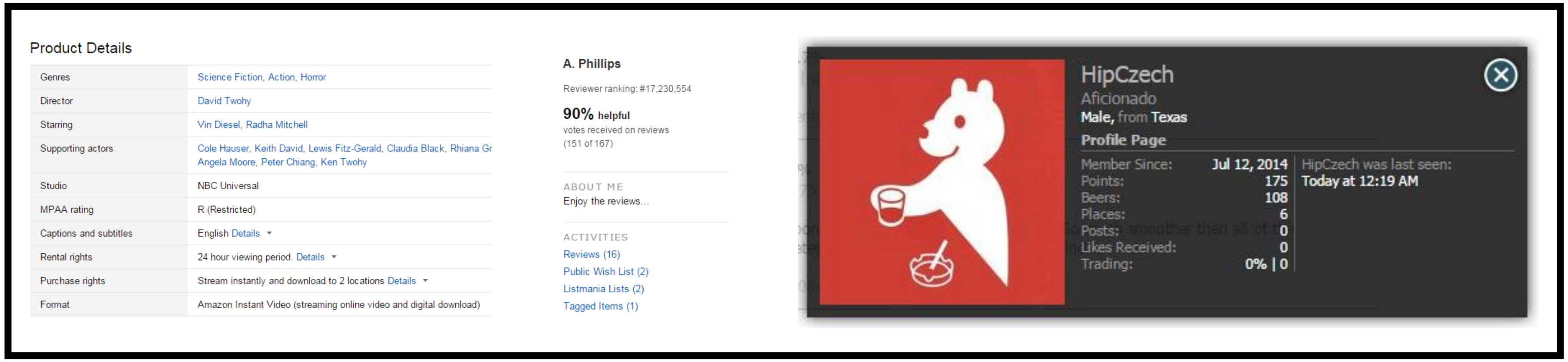
Web Mining and Recommender Systems

Supervised Learning: Summary so far

Learning Goals

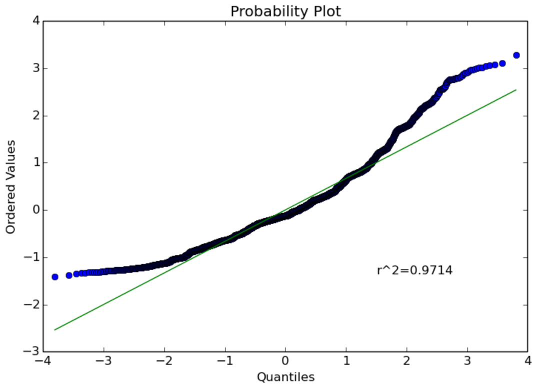
- Summarize our discussion of supervised learning

So far: Regression



How can we use **features** such as product properties and user demographics to make predictions about **real-valued** outcomes (e.g. star ratings)?

How can we prevent our models from **overfitting** by favouring simpler models over more complex ones?



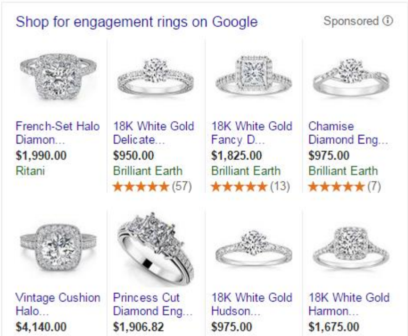
How can we assess our decision to optimize a particular error measure, like the MSE?

So far: Classification

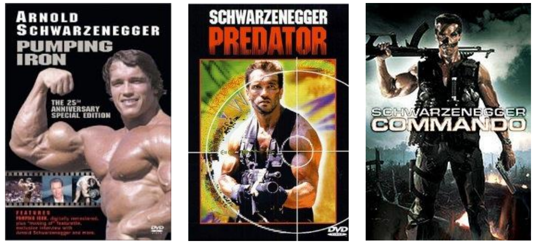
Next we adapted these ideas to **binary** or **multiclass** outputs

What animal is in this image?

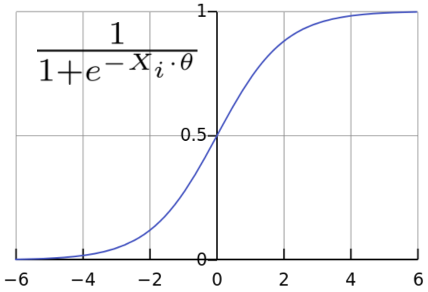
Will I **purchase** this product?



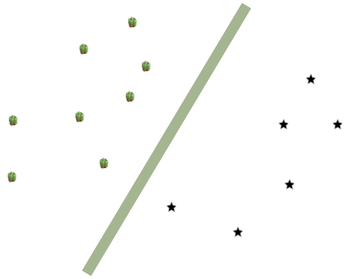
Will I **click on** this ad?



Combining features using naïve Bayes models



Logistic regression



Support vector machines

So far: supervised learning

Given **labeled training data** of the form

$$\{(\text{data}_1, \text{label}_1), \ldots, (\text{data}_n, \text{label}_n)\}$$

Infer the function

$$f(\text{data}) \stackrel{?}{\rightarrow} \text{labels}$$

So far: supervised learning

We've looked at two types of prediction algorithms:

Regression

$$y_i = X_i \cdot \theta$$

Classification

$$y_i = \begin{cases} 1 & \text{if } X_i \cdot \theta > 0 \\ 0 & \text{otherwise} \end{cases}$$

Further Reading

Further reading:

- “Cheat sheet” of performance evaluation measures: http://www.damienfrancois.be/blog/files/modelperfcheatsheet.pdf
- Andrew Zisserman’s SVM slides, focused on computer vision: http://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf