CSE 258 – Lecture 4

Web Mining and Recommender Systems

Evaluating Classifiers

Last lecture...

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

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

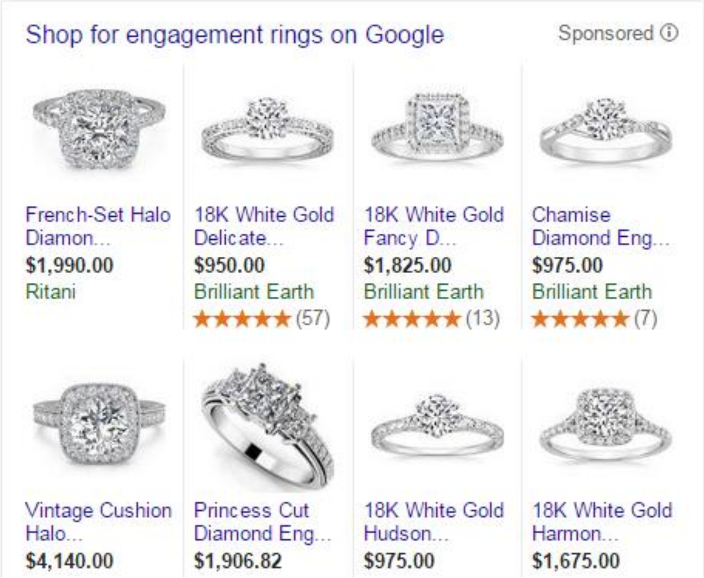
{0,1}, {True, False}

{1, ... , N}

Last lecture...

Will I **purchase** this product?

(yes)



Will I **click on** this ad?

(no)

Last lecture...

- **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

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

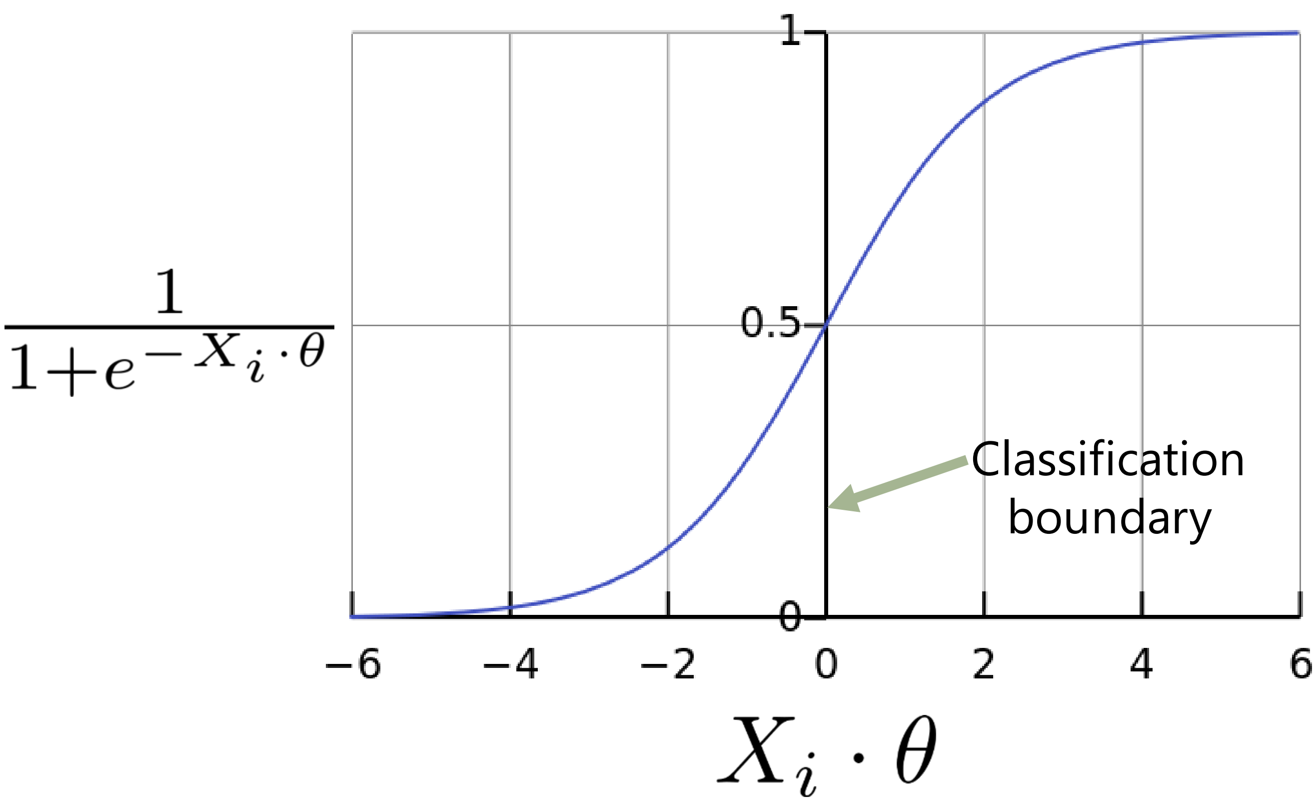
evidence

due to our conditional independence assumption:

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

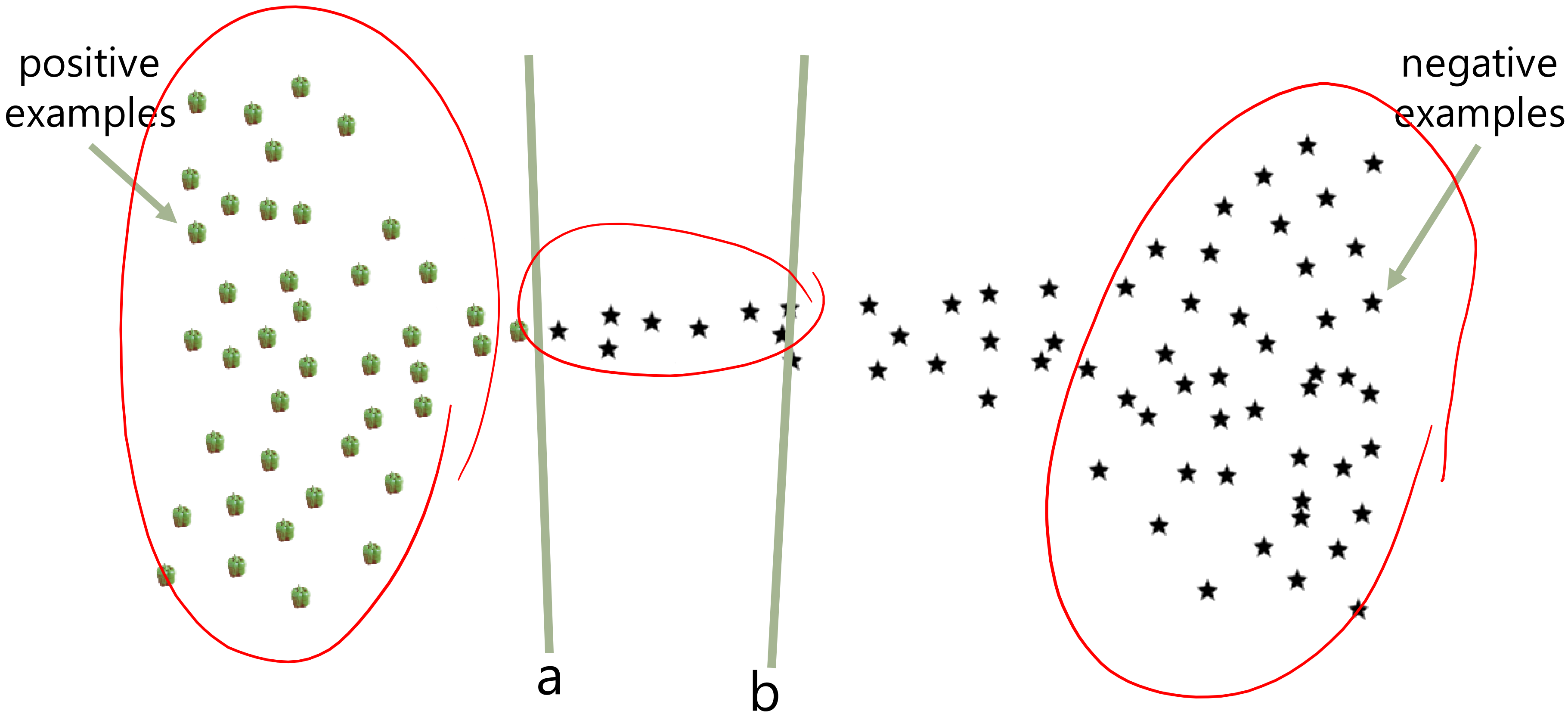
2) logistic regression

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



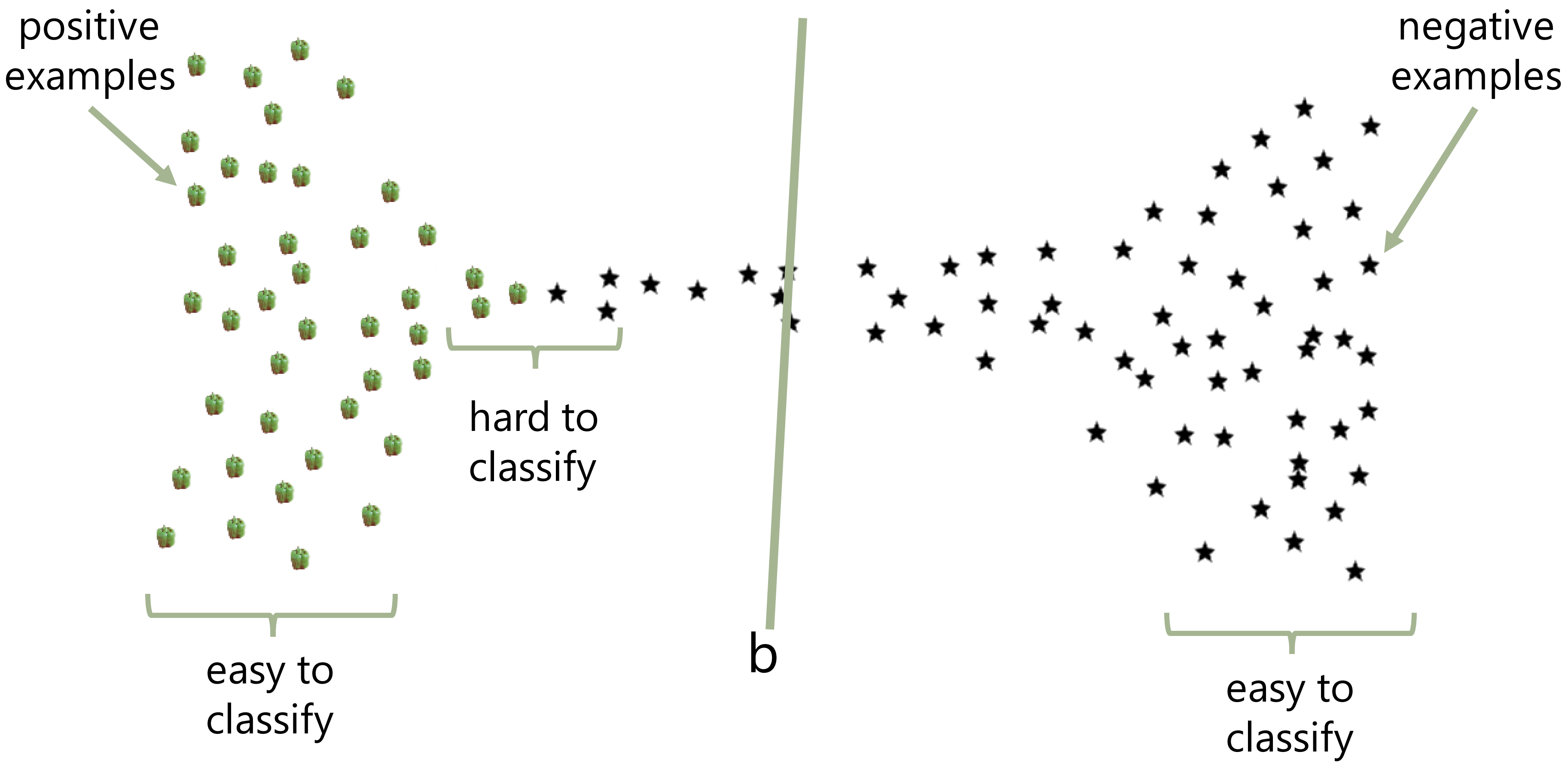
Logistic regression

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?

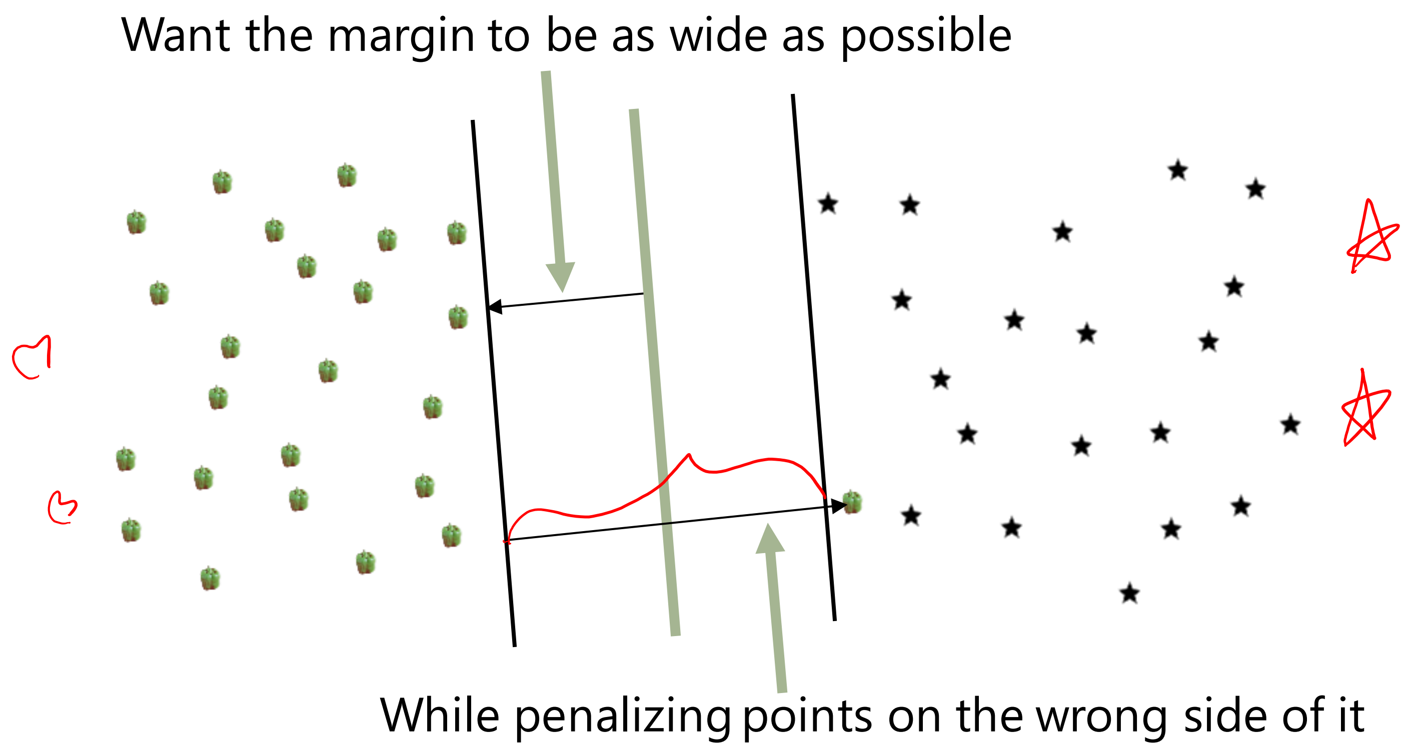


Logistic regression

- 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?



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

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**

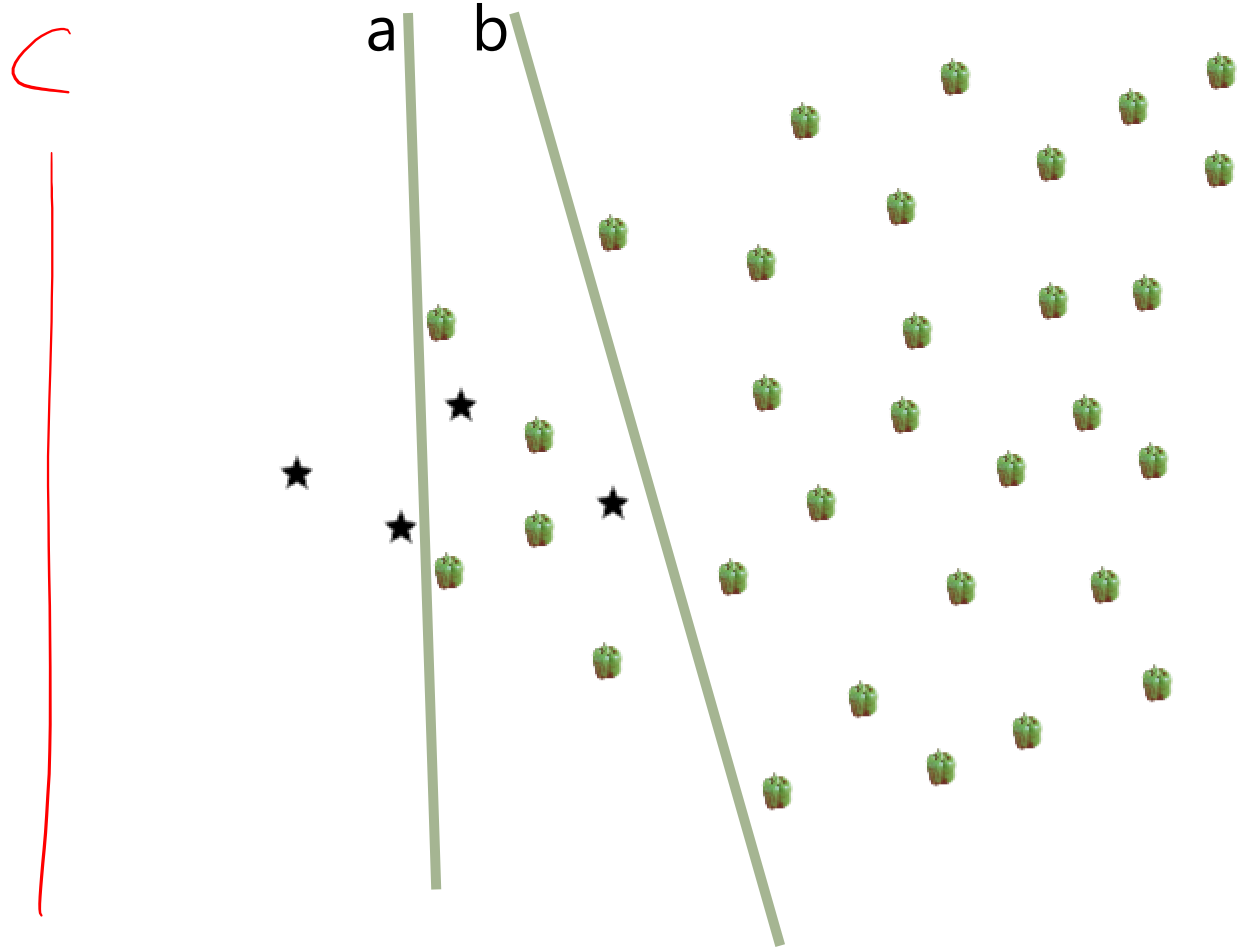
  ++ Non-probabilistic: optimizes the classification error rather than the likelihood
  -- More expensive to train

CSE 258 – Lecture 4

Web Mining and Recommender Systems

Evaluating classifiers

Which of these classifiers is best?



error

a = 2

b = 5

c = 4

Which of these classifiers is best?

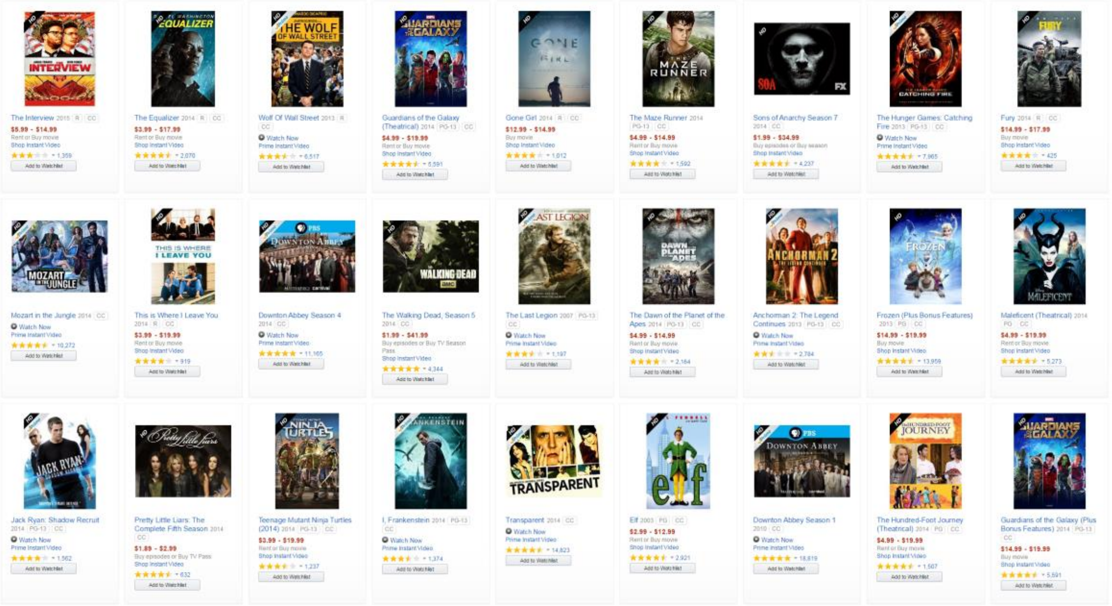
The solution which minimizes the #errors may not be the best one

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)

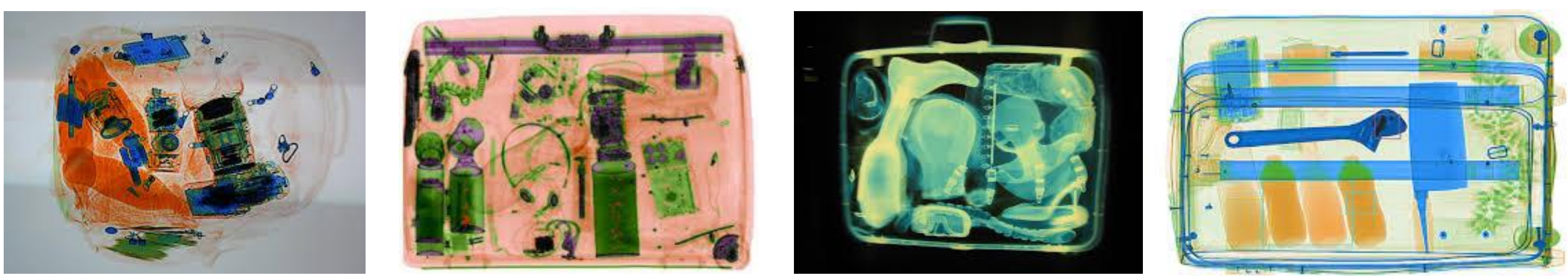
e.g. will I purchase a product? If I purchase 0.00001% of products, then a classifier which just predicts "no" everywhere is 99.99999% accurate, but not very useful



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)

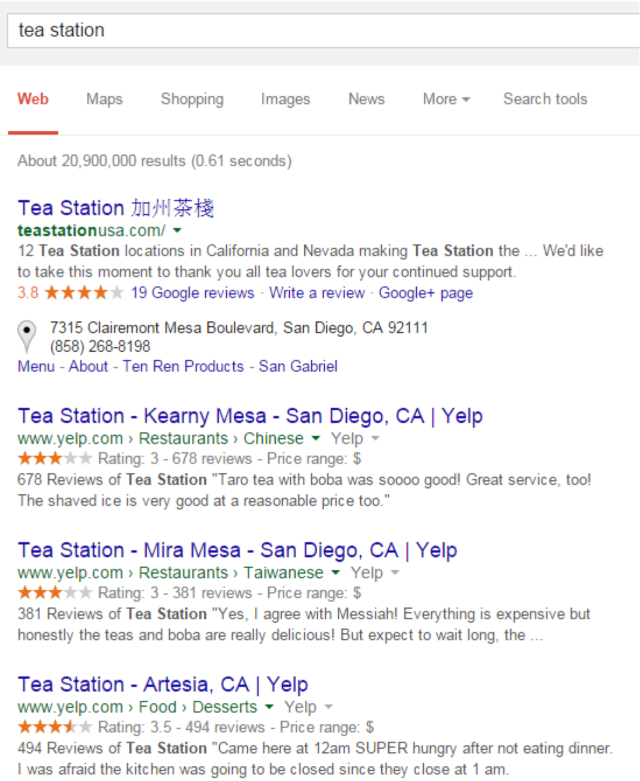


e.g. which of these bags contains a weapon?

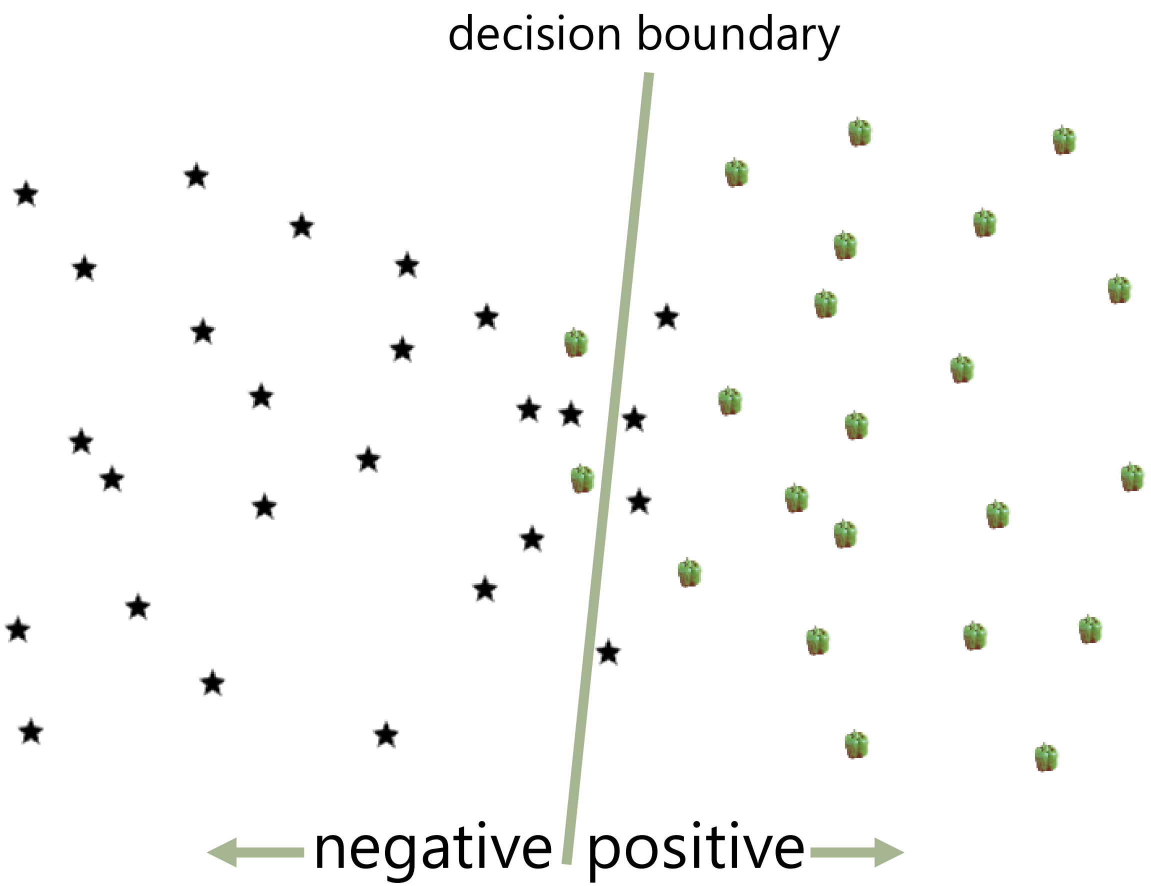
Which of these classifiers is best?

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

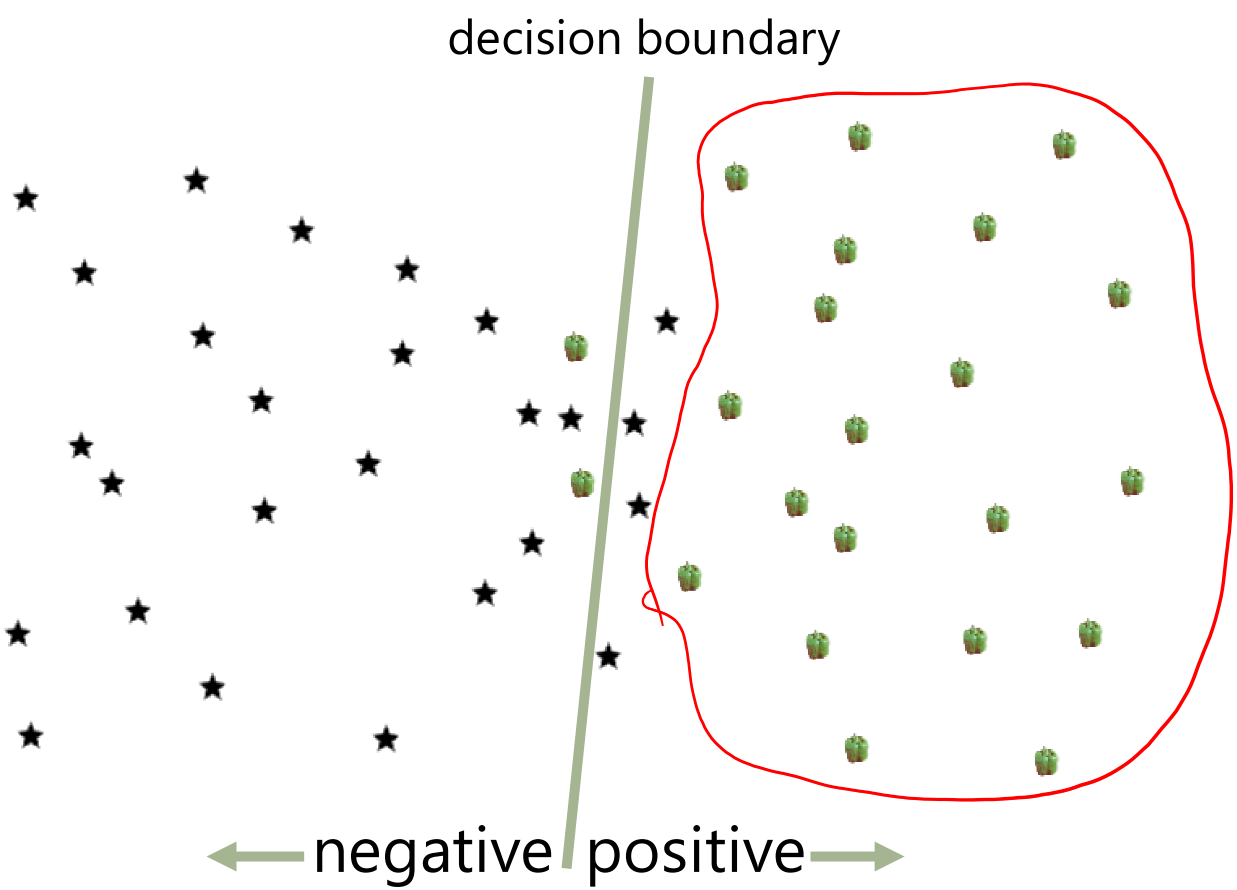
e.g. does a relevant result appear among the first page of results?



Evaluating classifiers

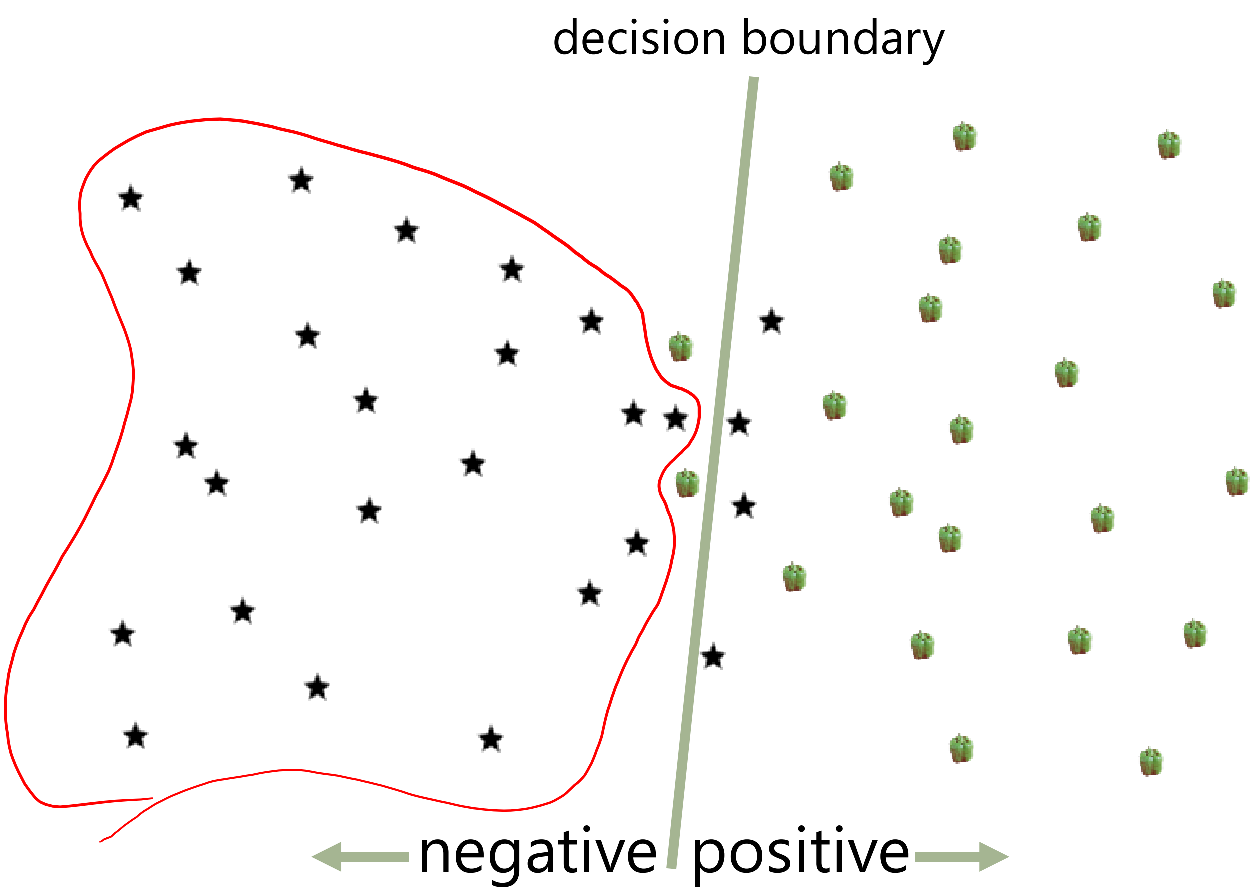


Evaluating classifiers



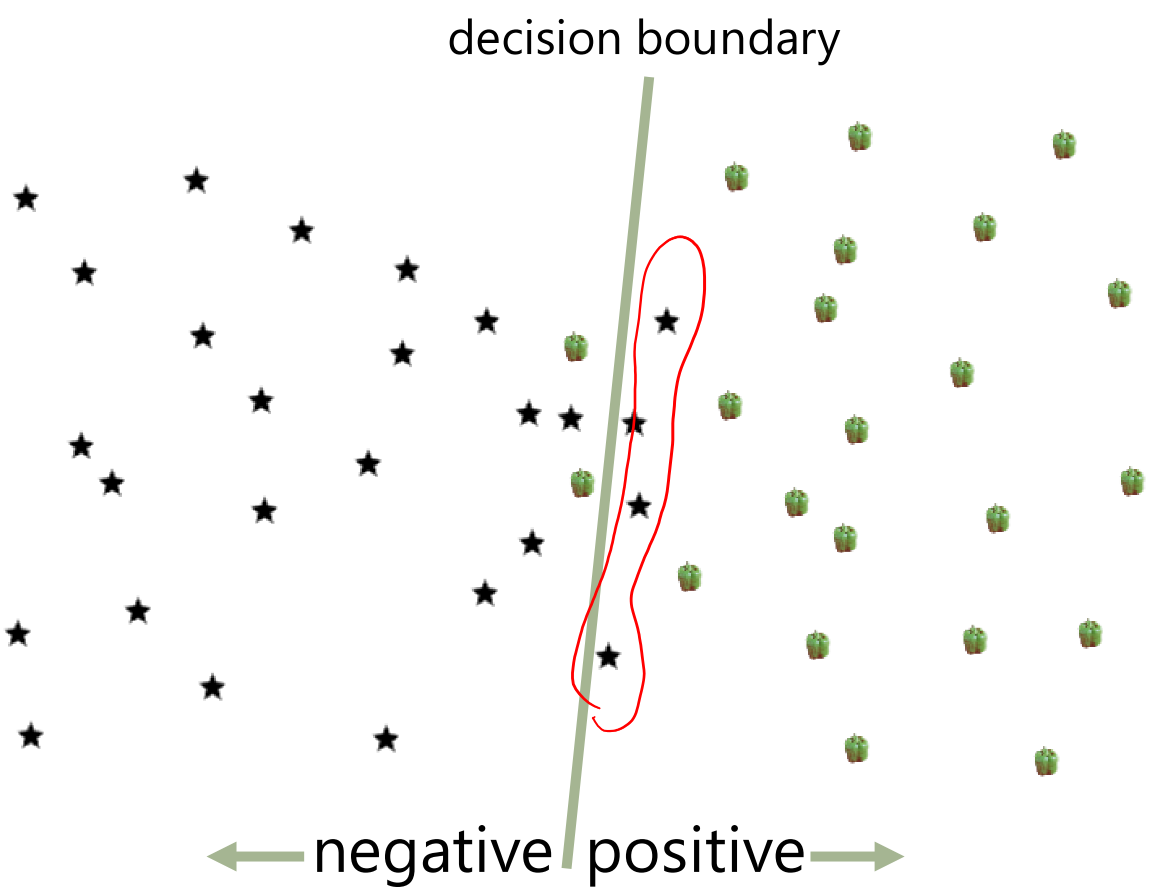
TP (true positive): Labeled as T, predicted as T

Evaluating classifiers



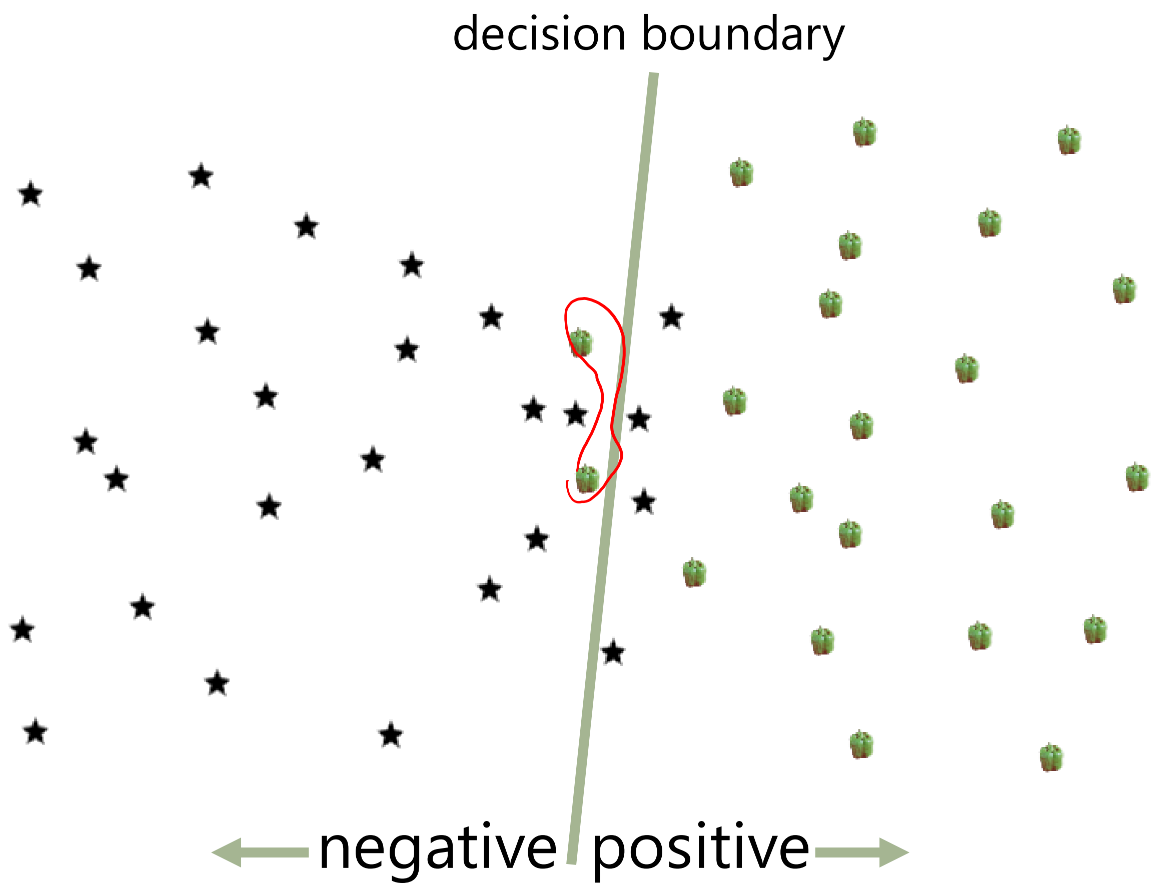
TN (true negative): Labeled as F, predicted as F

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
Prediction	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]

$X_i \cdot \theta$

tp tn fn fn tp fp tp tp tn tp

$TP = 5 \quad TN = 2$

$FP = 1 \quad FN = 2$

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

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

Evaluating classifiers

How to optimize a balanced error measure:

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

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

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

$$\left|\{y \in Y \mid y_i = 0\}\right|$$

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

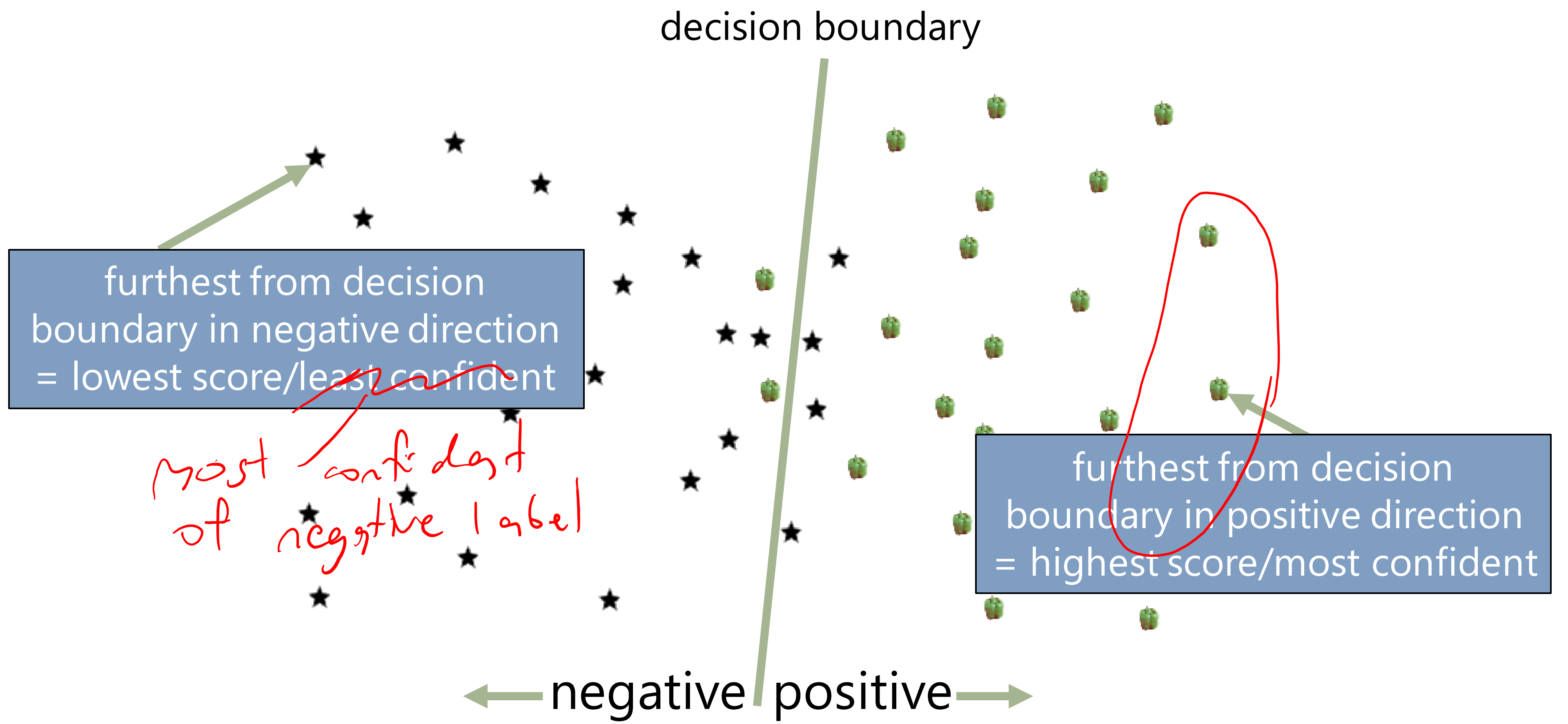
```
@relation '5year-weka.filters.unsupervised.instance.SubsetByExpression-Enot  ismissing(ATT20)'

@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: http://jmcauley.ucsd.edu/code/week2.py

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 $-X_i \cdot \theta$
- 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:

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

$\tilde{y}$ = [1 1 1 -1 1 -1 1 -1 1 -1]

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 = $5/6$

Evaluating classifiers – ranking

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

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

(harmonic mean of precision and recall)

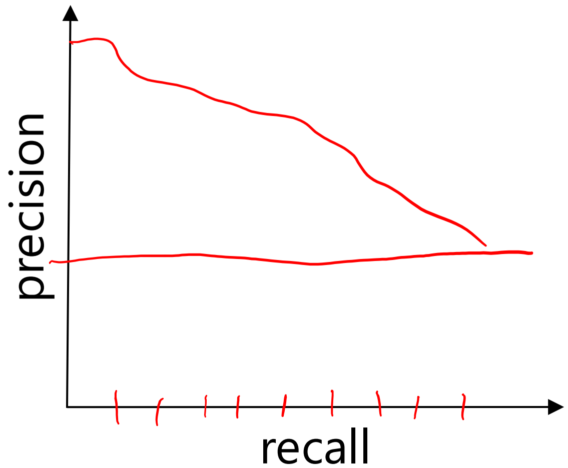
$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \text{precision} + \text{recall}}$$

(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



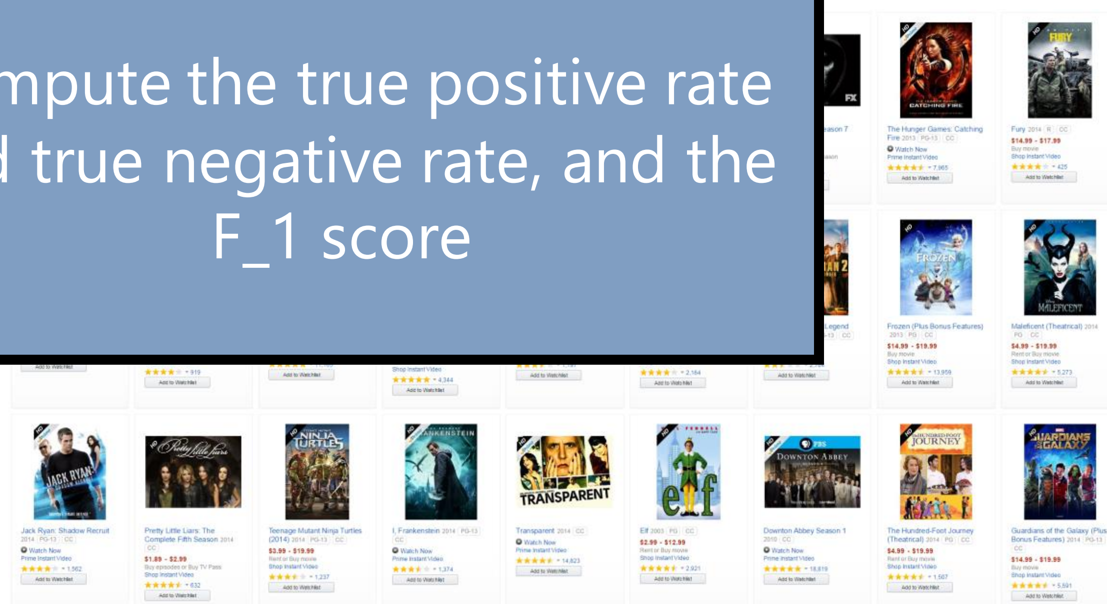
Summary

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
classifier which jus
predicts "no"
everywhere is
99.99999% accurate,
but not very useful



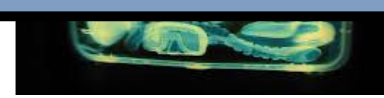
Summary

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





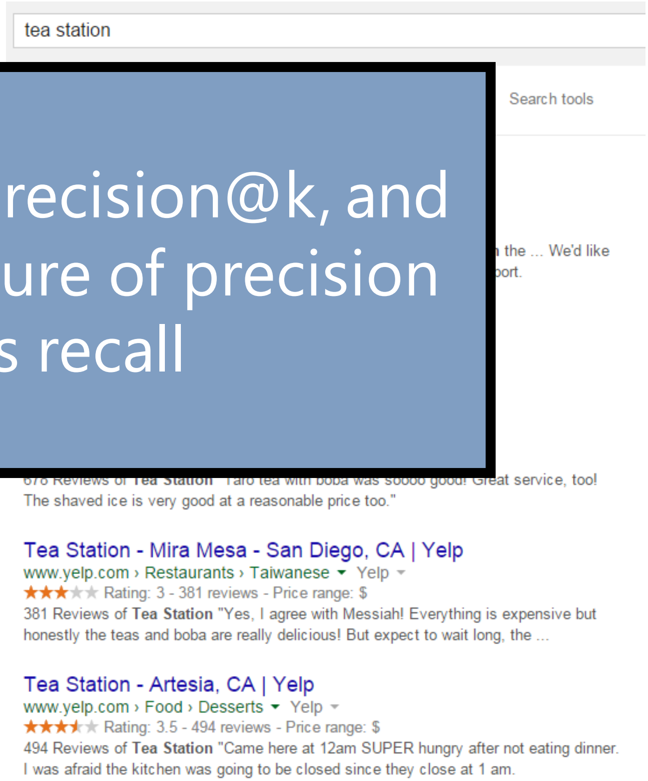

e.g. which of these bags contains a weapon?

Summary

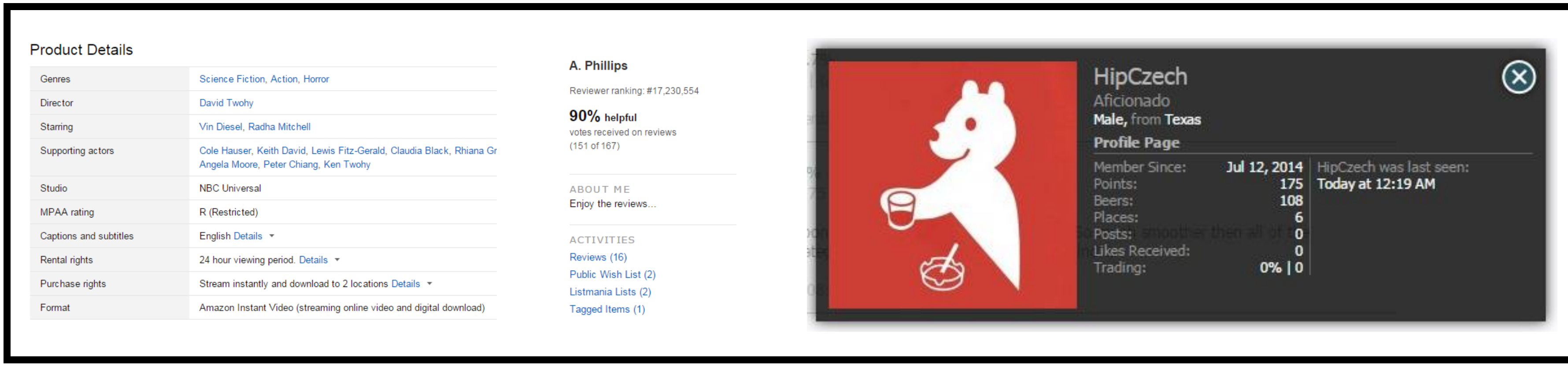
3. When we only care about the "most confident" predictions

e.g. does
result
among
page of results?

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

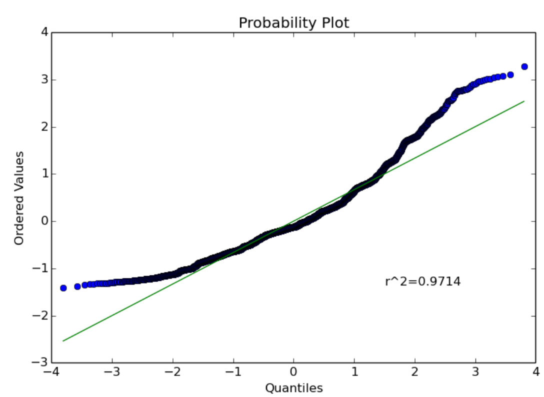


So far: Regression



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?



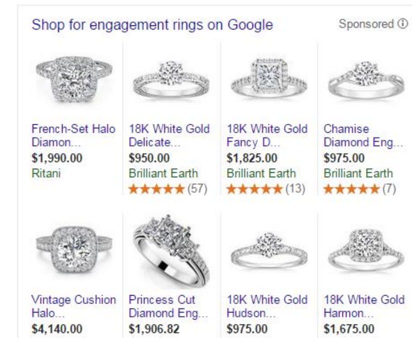
How can we assess our decision to optimize a particular error measure, like the MSE?

So far: Classification

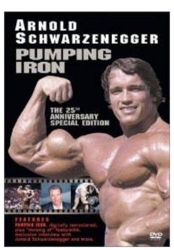
Next we adapted these ideas to **binary** or **multiclass** outputs

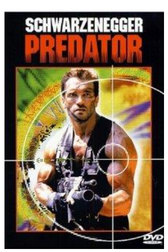
What animal is in this image?

Will I **purchase** this product?

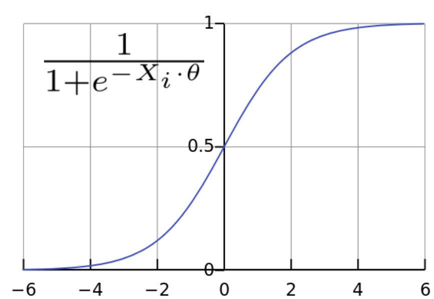


Will I **click on** this ad?

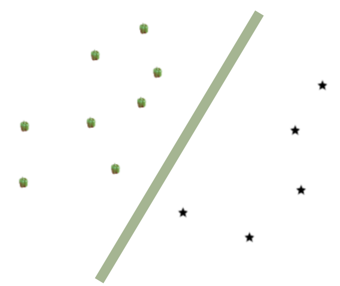




Combining features using naïve Bayes models



Logistic regression



Support vector machines

So far: supervised learning

Given **labeled training data** of the form

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

Infer the function

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

So far: supervised learning

We've looked at two types of prediction algorithms:

Regression

$$y_i = X_i \cdot \theta$$

Classification

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

Questions?

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