CSE 190 – Lecture 4

Data Mining and Predictive Analytics

Nearest Neighbour & Classifier Evaluation

Last lecture…

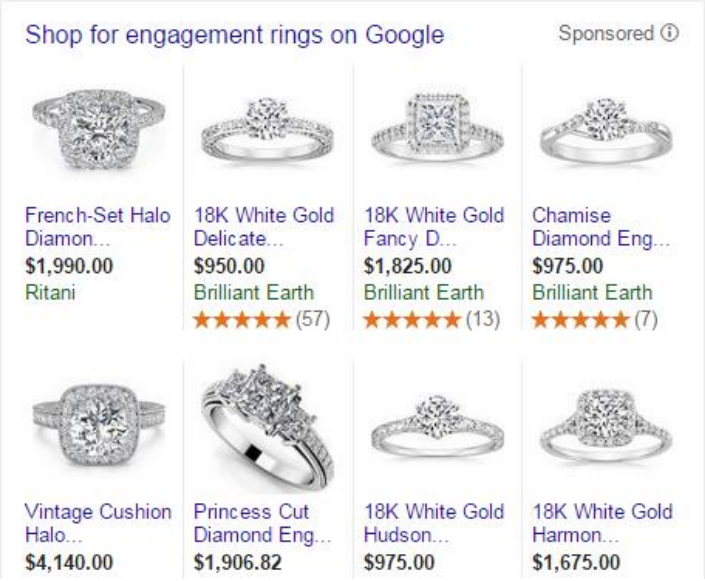
How can we predict **binary** or **categorical** variables?

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

{0,1}, {True, False}

{1, … , N}

Last lecture…



Will I **purchase** this product?

(yes)

Will I **click on** this ad?

(no)

Last lecture…

What animal appears in this image?

(mandarin duck)

Last lecture…

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.*

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

Last lecture…

The classifiers we saw last week all attempt to make decisions by associating weights (theta) with features (x) and classifying according to

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

The difference between the three is then simply how they're trained

1) Naïve Bayes

posterior prior likelihood

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

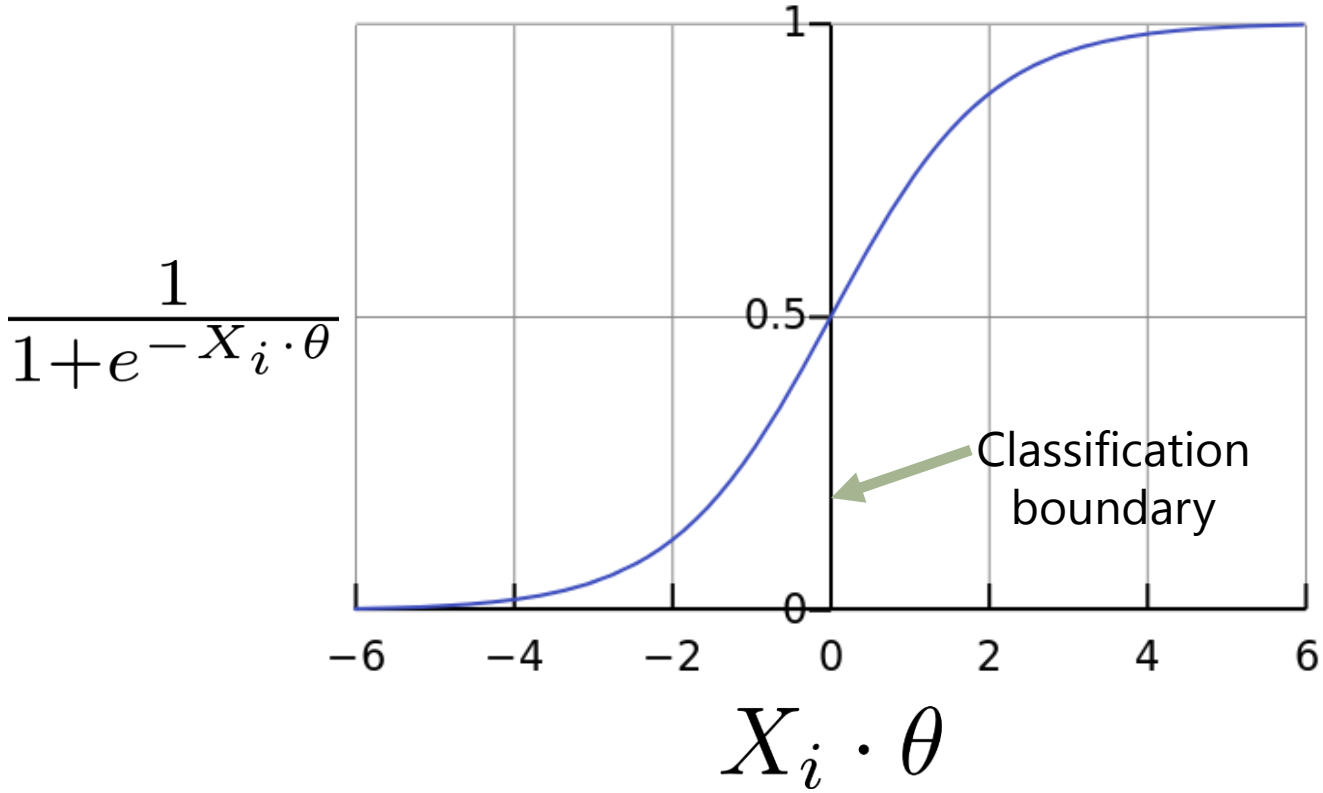
evidence

due to our conditional independence assumption:

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

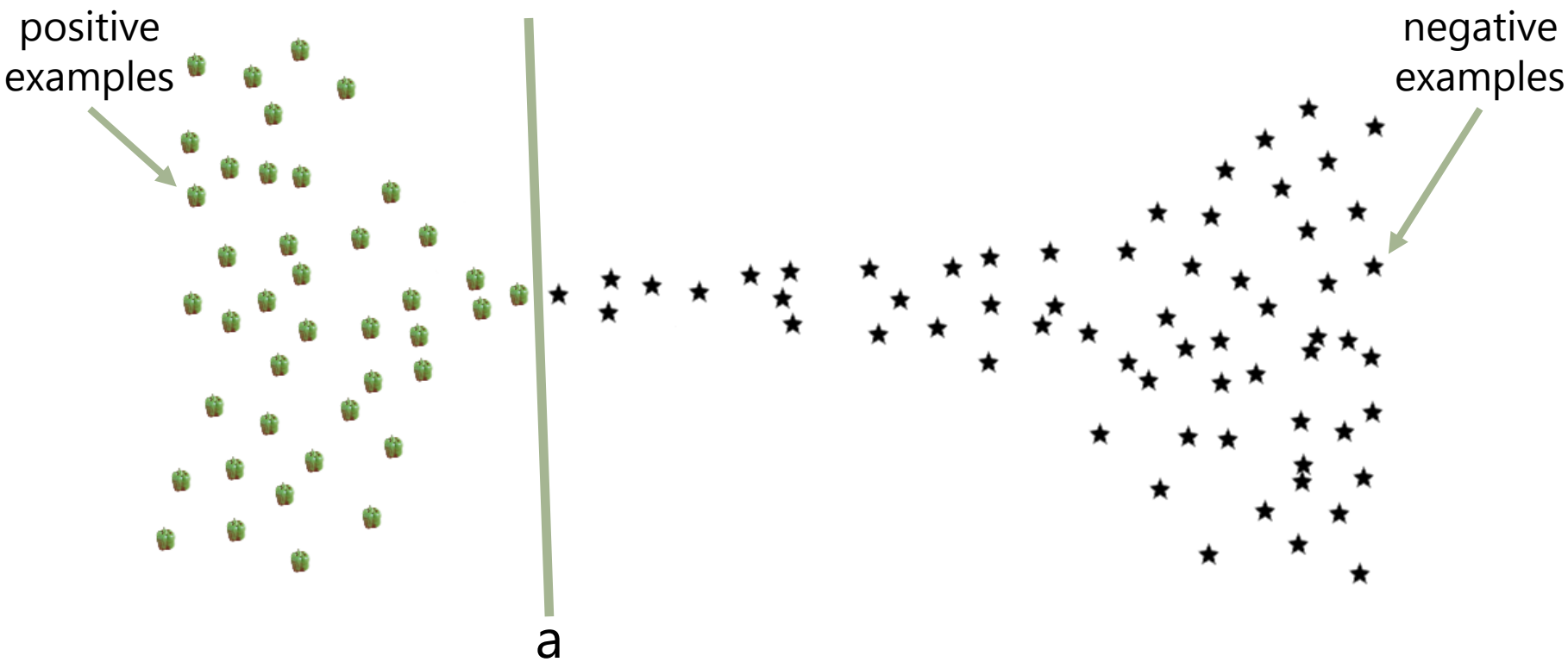
2) logistic regression

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



3) Support Vector Machines

Try to optimize the **misclassification error** rather than maximize 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

Today

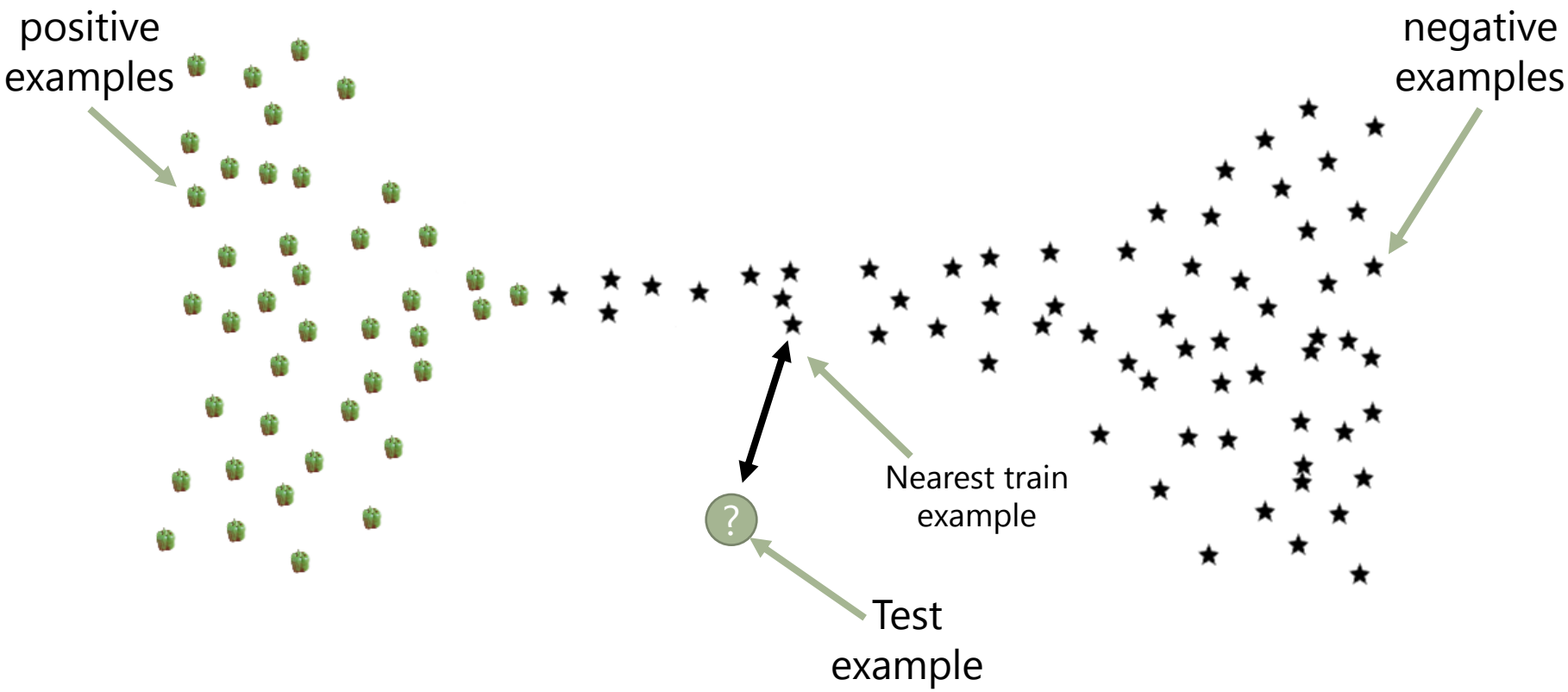
1. A different type of classifier, based on measuring the *distance* between labeled/unlabeled instances
2. How to evaluate whether our classifiers are good or not
3. Another case study – building a classifier to predict which products will be copurchased

Nearest-neighbour classification

One more (even simpler) classifier

Nearest-neighbour classification

Given a new point, just find the nearest labeled point, and assign it the same label



Nearest-neighbour classification

Given a new point, just find the nearest labeled point, and assign it the same label

$$\text{label}(X) = \underbrace{\text{label}(\underbrace{\arg\min_{Z \in \text{train}} d(X, Z)}_{\text{Nearest training point}})}_{\text{Label of nearest training point}}$$

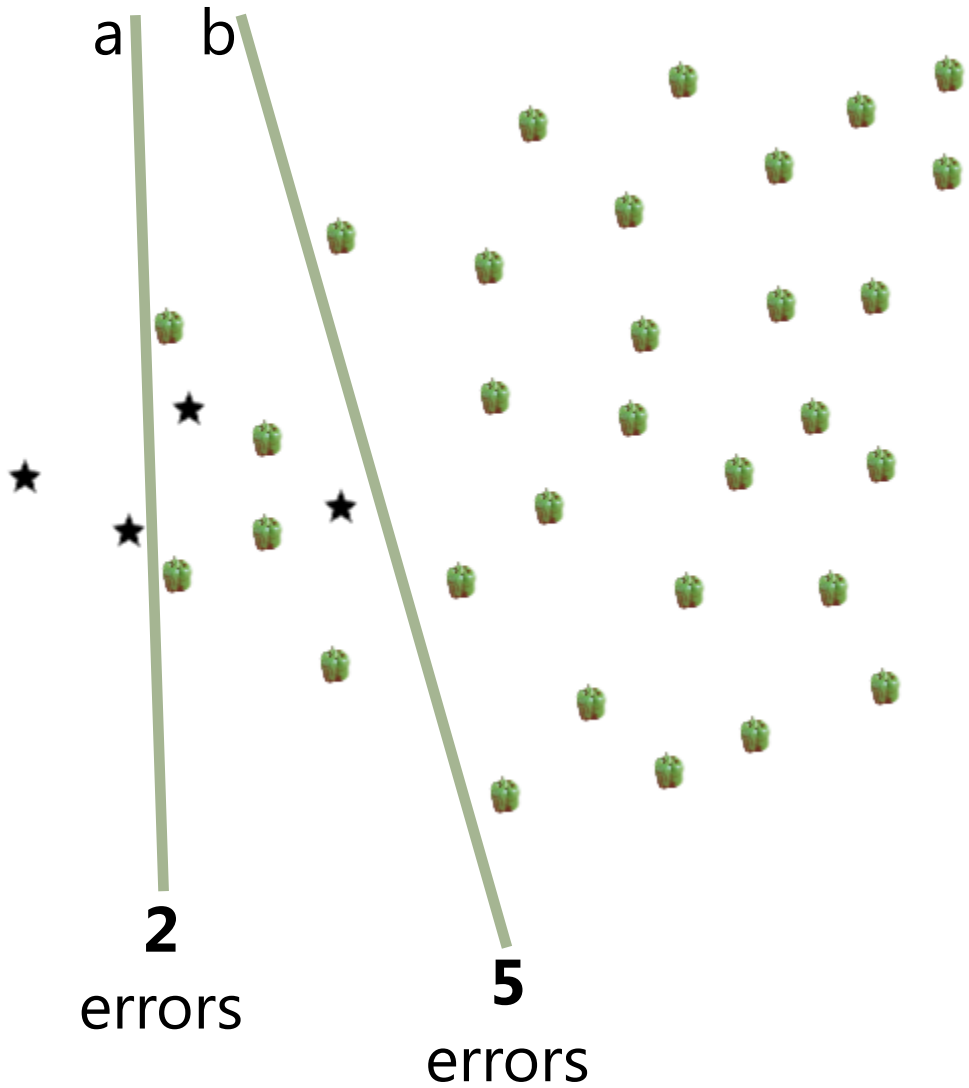
We'll look more at this type of classifier in the case-study later tonight

CSE 190 – Lecture 4

Data Mining and Predictive Analytics

Evaluating classifiers

Which of these classifiers is best?



Which of these classifiers is best?

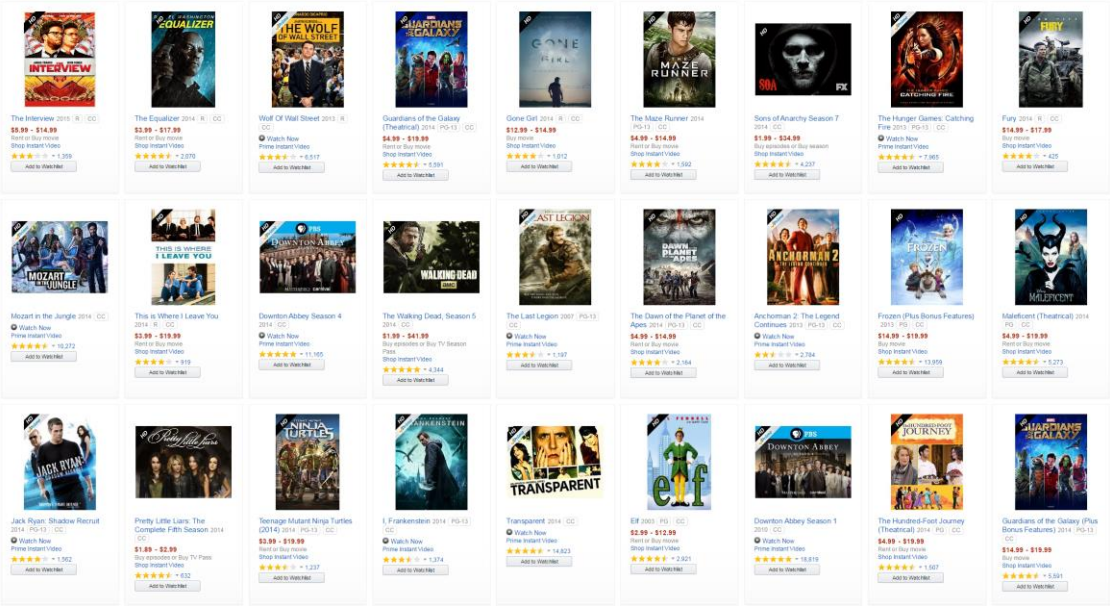
The solution which minimizes the #errors may not be the best one

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)

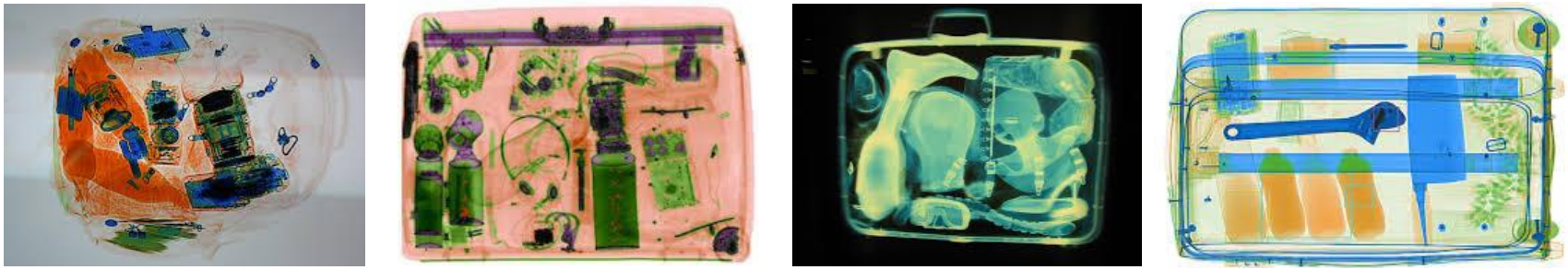
e.g. will I purchase a product? If I purchase 0.00001% of products, then a classifier which just predicts “no” everywhere is 99.99999% accurate, but not very useful



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)

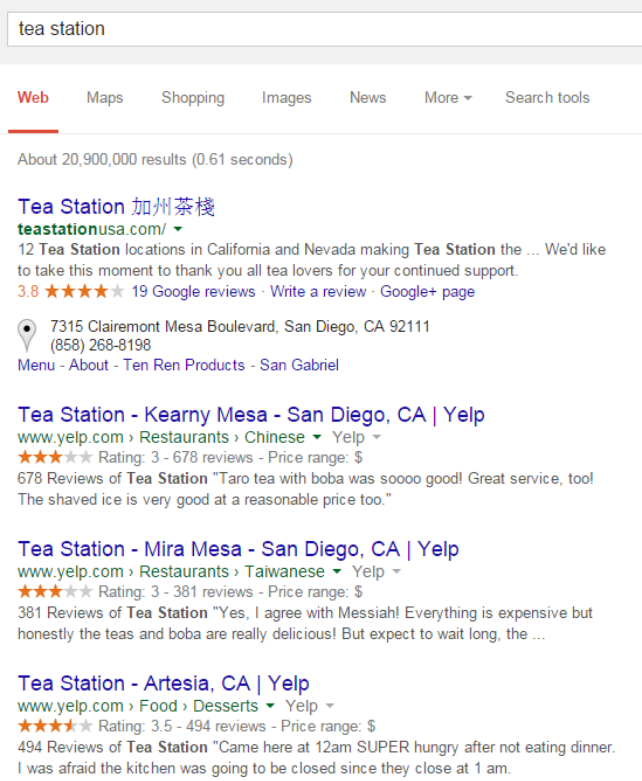


e.g. which of these bags contains a weapon?

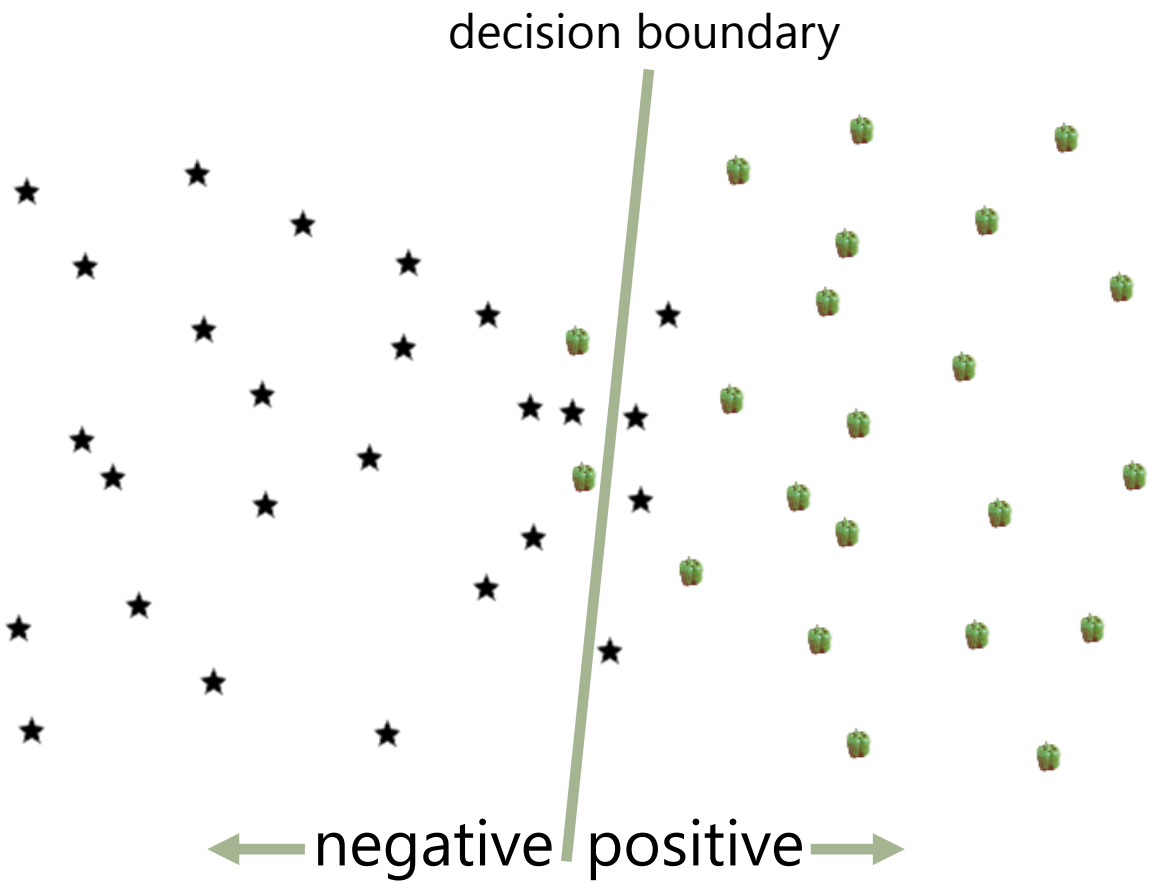
Which of these classifiers is best?

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

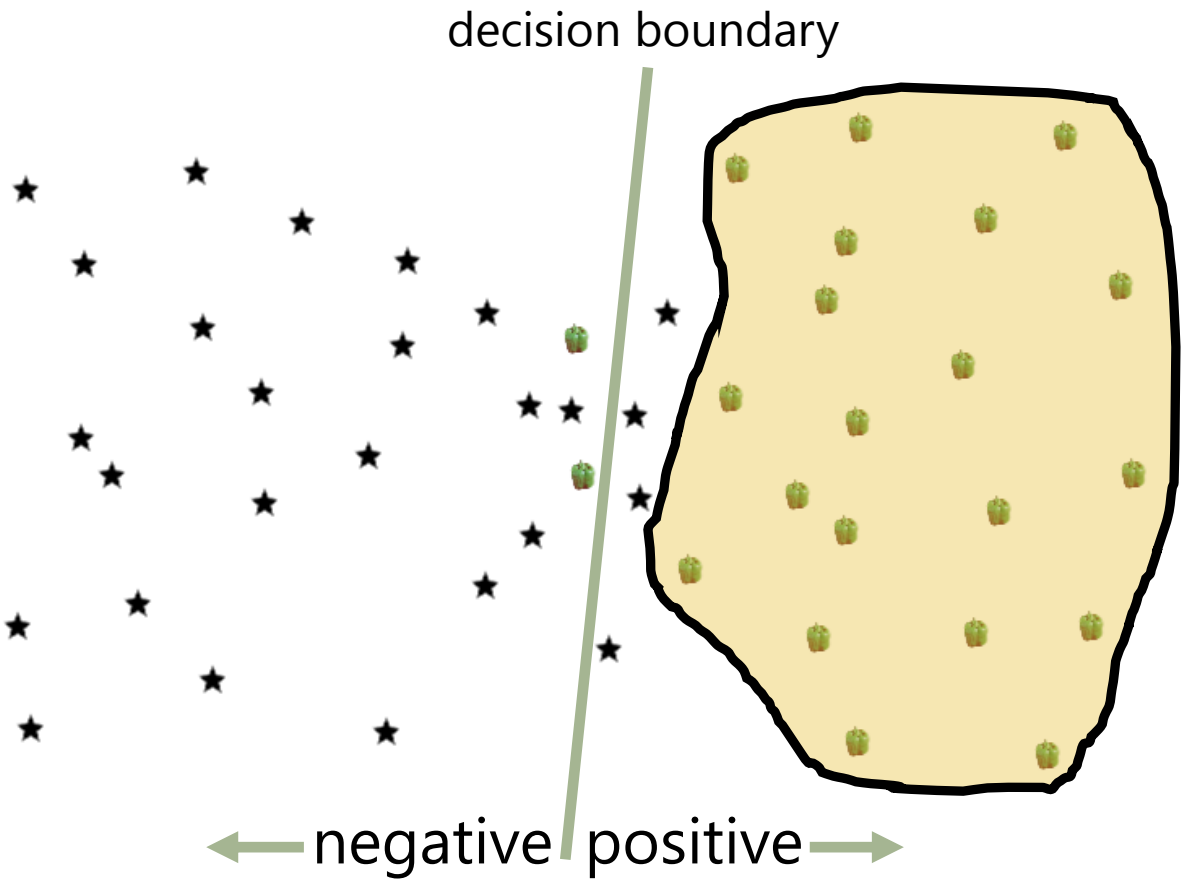
e.g. does a relevant result appear among the first page of results?



Evaluating classifiers

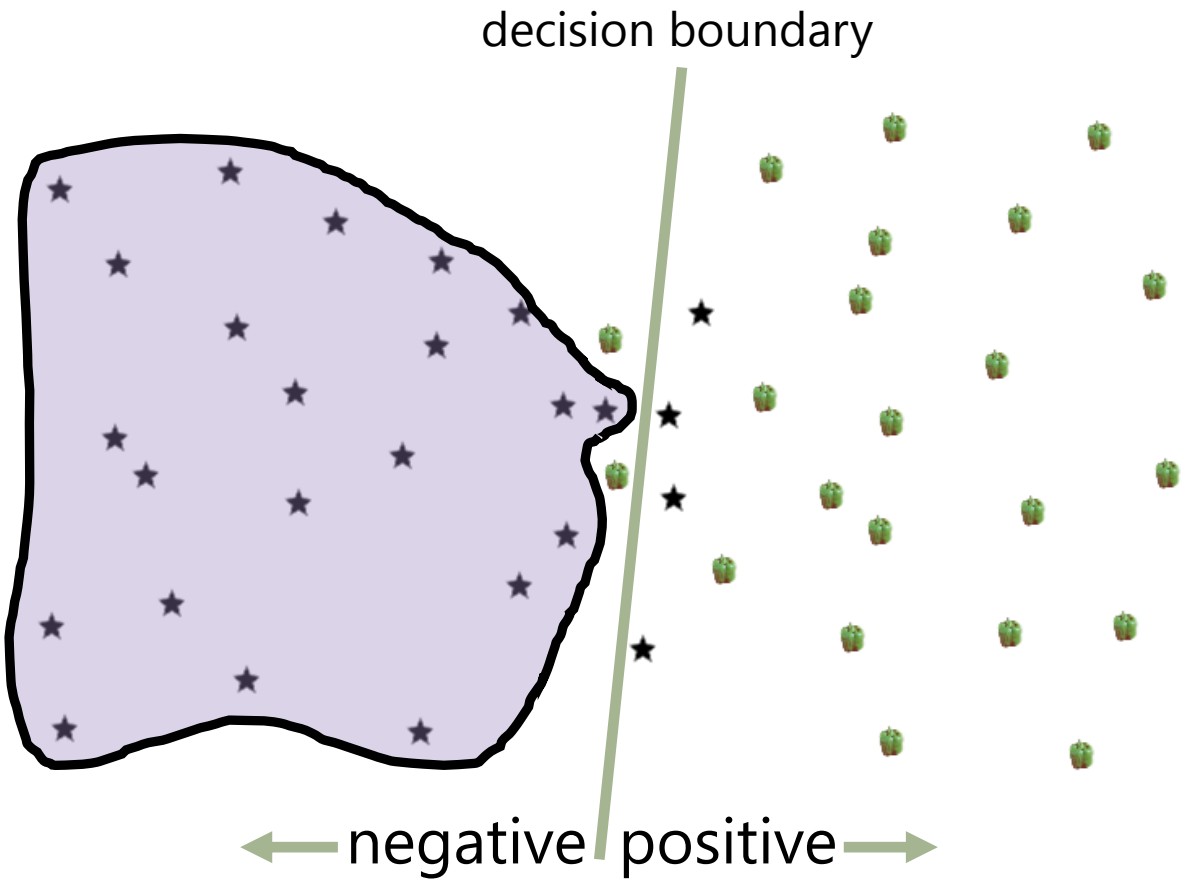


Evaluating classifiers



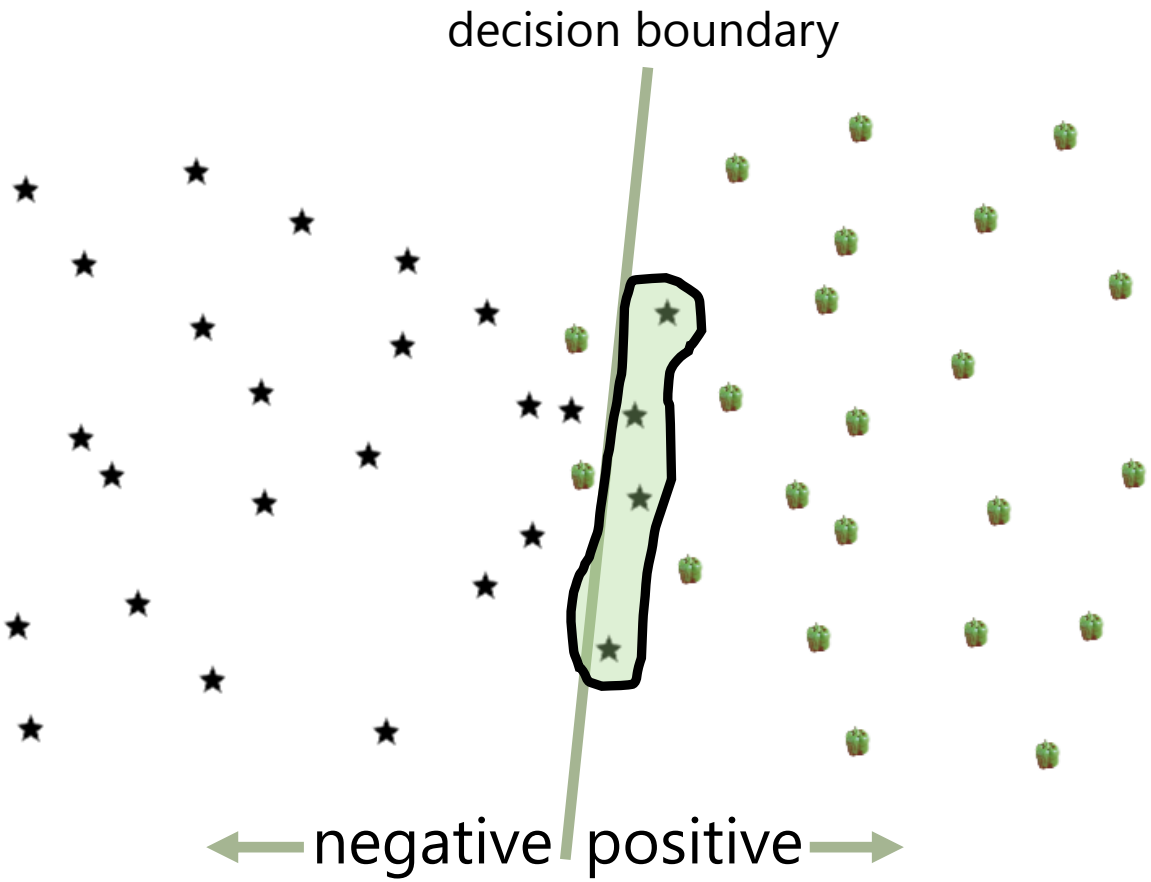
TP (true positive): Labeled as positive, predicted as positive

Evaluating classifiers



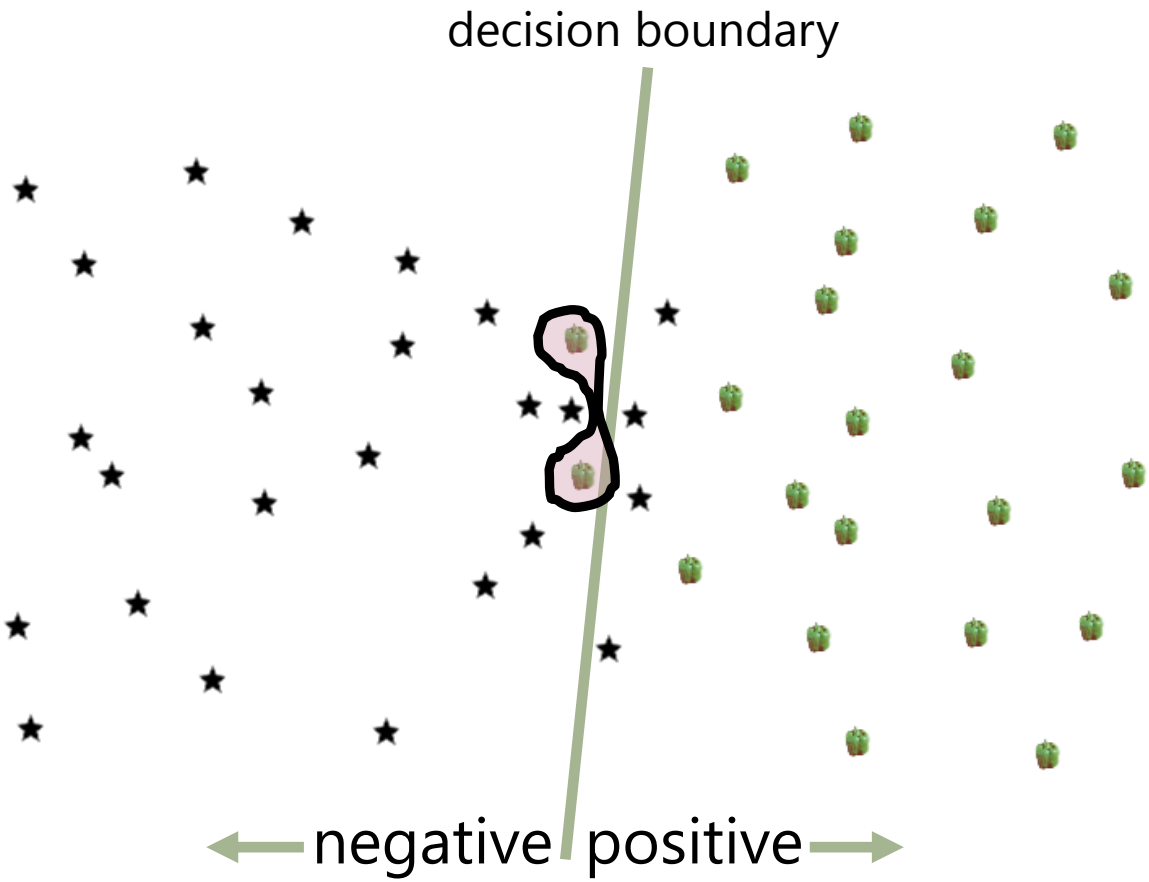
TN (true negative): Labeled as negative, predicted as negative

Evaluating classifiers



FP (false positive): Labeled as negative, predicted as positive

Evaluating classifiers



FN (false negative): Labeled as positive, predicted as negative

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

"sensitivity"

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

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

"specificity"

Evaluating classifiers

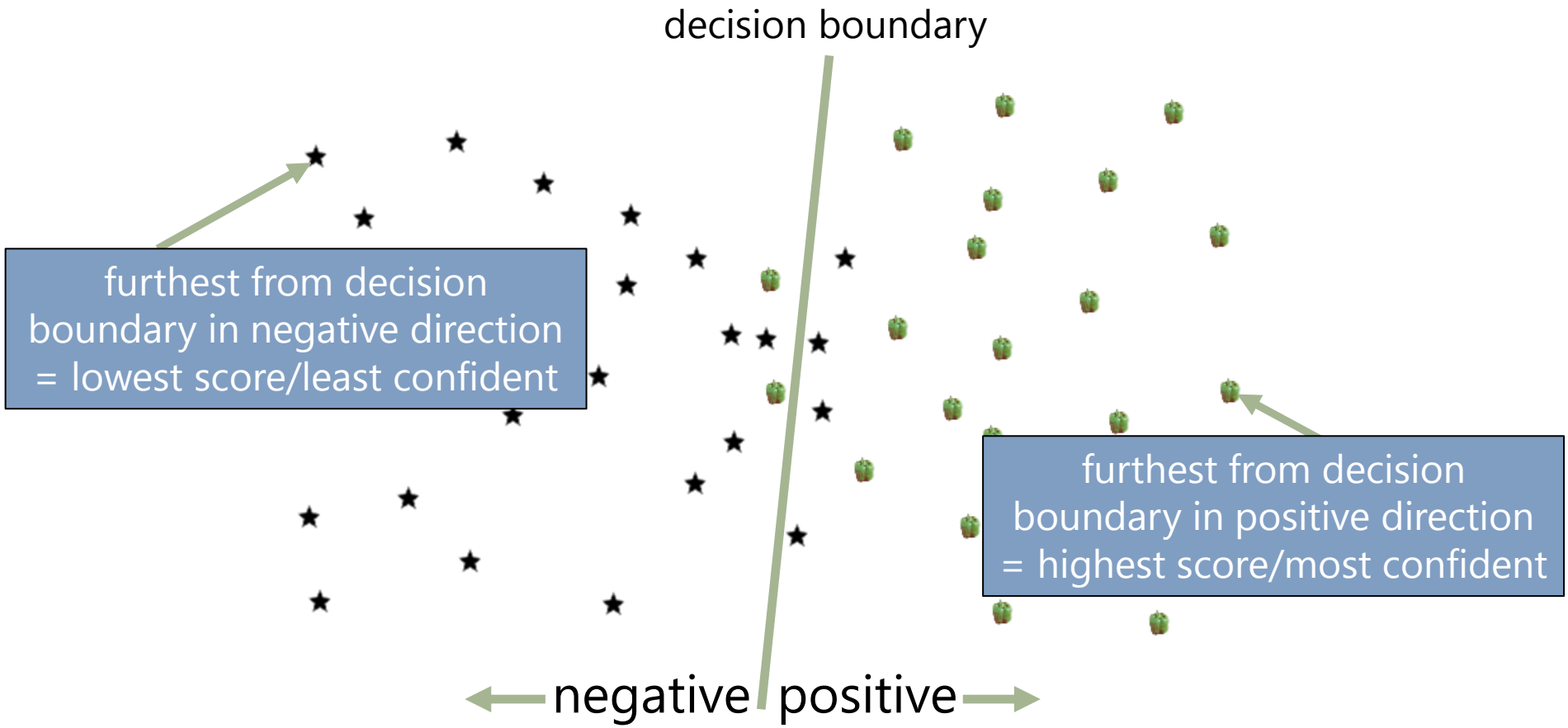
		Label	
		true	false
Prediction	true	true positive	false positive
	false	false negative	true negative

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

= ½ for a random/naïve classifier, 1 for a perfect classifier

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 (the same quantity we compute in the homework question)
- 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:

[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)

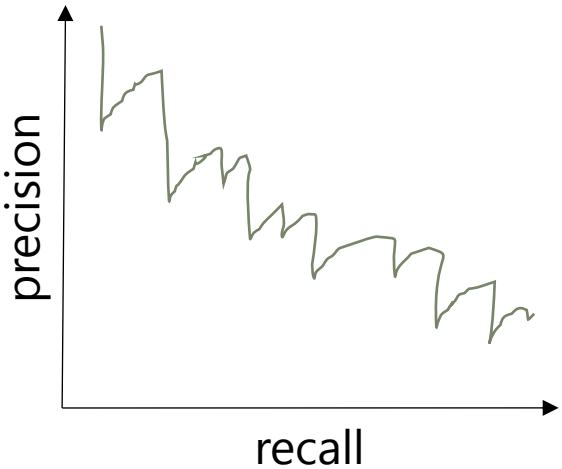
$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \text{precision} + \text{recall}}$$

(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



Todo: less crappy plot

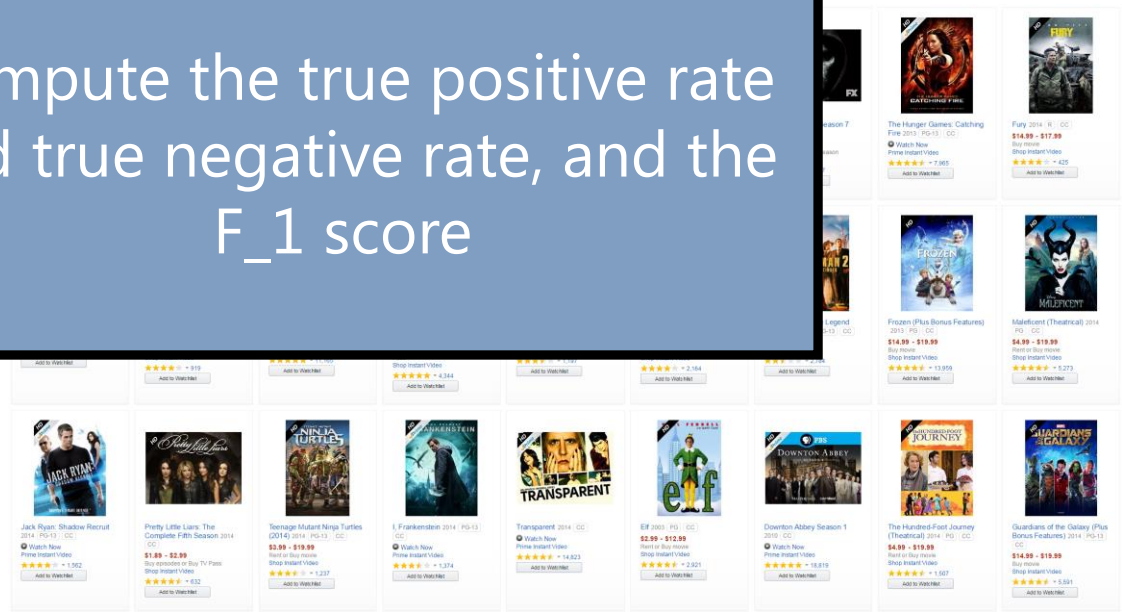
Summary

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)

e.g. will I purchase product? If I purchase 0.000019 of products, then a classifier which just predicts "no" everywhere is 99.99999% accurate, but not very useful

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

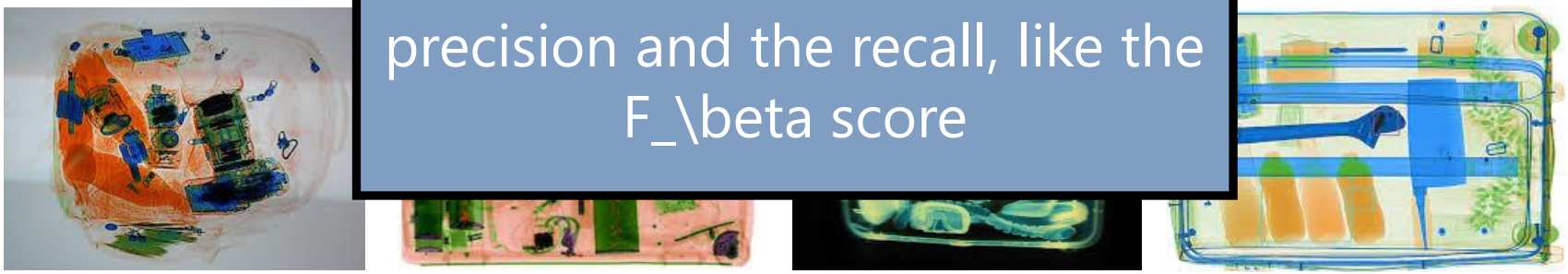


Summary

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



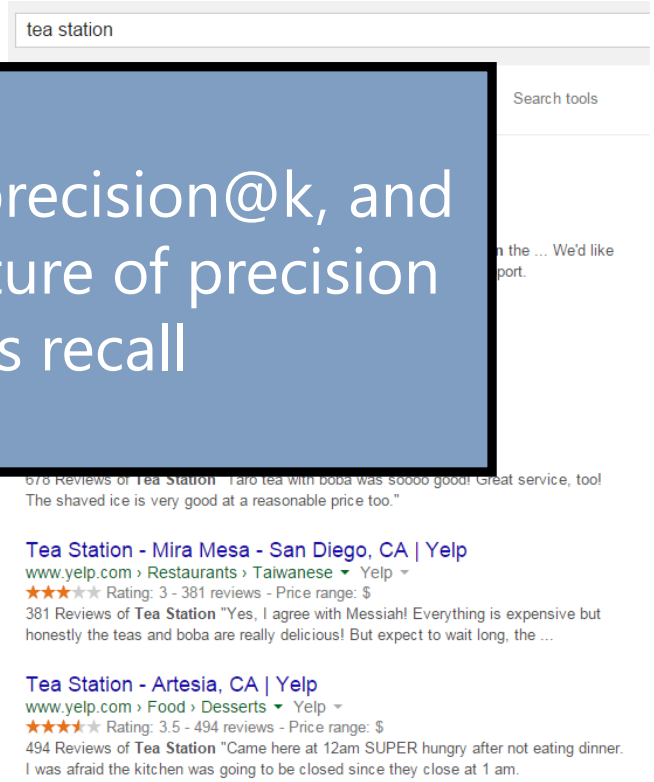
e.g. which of these bags contains a weapon?

Summary

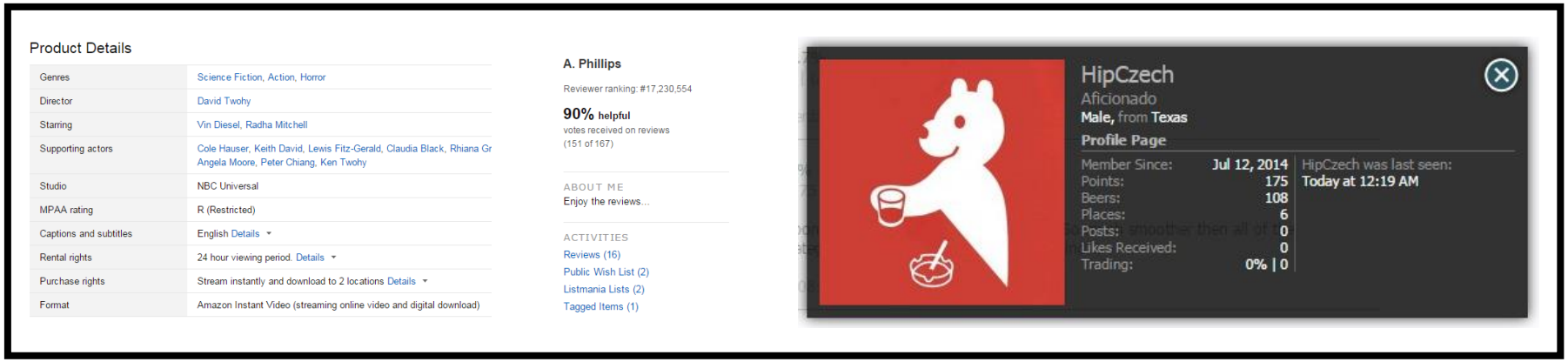
3. When we only care about the "most confident" predictions

e.g. does
result
among
page of results?

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

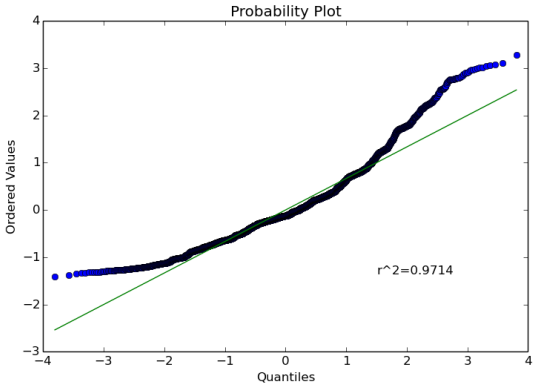


So far: Regression



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?



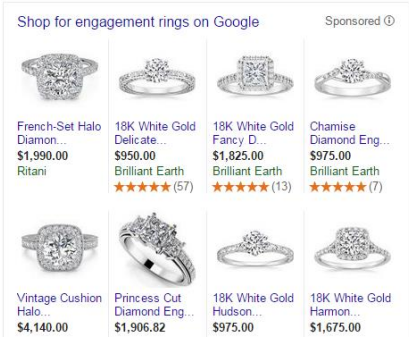
How can we assess our decision to optimize a particular error measure, like the MSE?

So far: Classification

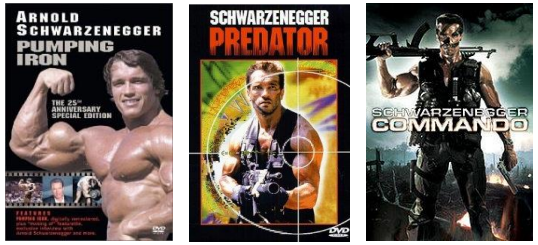
Next we adapted these ideas to **binary** or **multiclass** outputs

What animal is in this image?

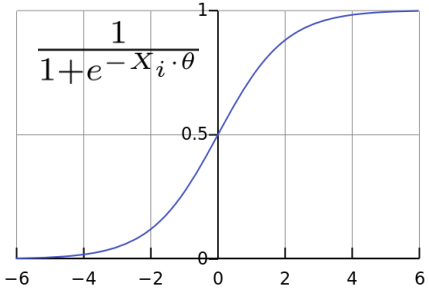
Will I **purchase** this product?



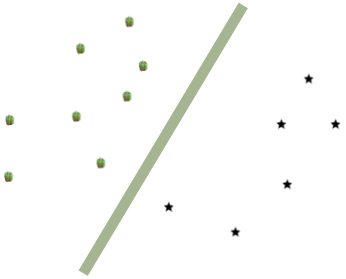
Will I **click on** this ad?



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

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

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