Web Mining and Recommender Systems

Classification (& Regression Recap)
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**

\[ f(\text{data}) \rightarrow \text{labels} \]
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...

\[ f(\text{user features}, \text{movie features}) \rightarrow \text{star rating} \]
Four important ideas:

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

\[
\text{MSE} \sim \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i \cdot \theta - N(0, \sigma))^2
\]

\[
d_i = (x_i \cdot \theta - y_i)
\]
Four important ideas:

2) Gradient descent for model optimization

1. Initialize $\theta$ 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!

\[ \theta = \arg \min_{\theta} \text{error}(\theta) + \text{complex}(\theta) \]

\[ \text{val}(\theta) = \text{error}(\theta, v) \]

\[ \theta = \arg \min_{\theta} \text{val}(\theta) \]
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
How can we predict **binary** or **categorical** variables?

\[ f(\text{data}) \rightarrow \text{labels} \]

\{0,1\}, \{True, False\}

\{1, \ldots, N\}
Will I **purchase** this product?  
(yes)

Will I **click on** this ad?  
(no)
What animal appears in this image?

(mandarin duck)
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.
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

---

**Tea Station** 加州茶栈

Teastationusa com/  
12 Tea Station locations in California and Nevada making Tea Station the ... We'd like to take this moment to thank you all tea lovers for your continued support.

[3.8 ★★★★★](#)  19 Google reviews - Write a review - Google+ page

7315 Clairemont Mesa Boulevard, San Diego, CA 92111  
(858) 268-8198

Menu - About - Ten Ron Products - San Gabriel

Tea Station - Kearny Mesa - San Diego, CA | Yelp  
www.yelp.com / Restaurants » Chinese » Yelp
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
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. $p(\text{movie is “action” | schwarzenegger in cast})$

Just count!

#films with Arnold = 45

#action films with Arnold = 32

$p(\text{movie is “action” | schwarzenegger in cast}) = \frac{32}{45}$
Naïve Bayes

What about:

\[
p(\text{movie is "action" |}
  \text{schwarzenegger in cast and}
  \text{release year = 2017 and}
  \text{mpaa rating = PG and}
  \text{budget < $1000000})
\]

\(#(\text{training}) \text{ films with Arnold, released in 2017, rated PG, with a}
\text{budget below $1M} = 0\)

\(#(\text{training}) \text{ action films with Arnold, released in 2017, rated PG, with a}
\text{budget below $1M} = 0\)
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 assumes that features are **conditionally independent** given the label

$$(\text{feature}_i \perp \text{feature}_j | \text{label})$$
Naïve Bayes

\[(\text{feature}_i \perp \text{feature}_j | \text{label})\]

\[a \perp b \Rightarrow p(a, b) = p(a)p(b)\]
\[a \perp b | c \Rightarrow p(a, b | c) = p(a | c)p(b | c)\]

\[a = \text{I'm wearing shorts} \]
\[b = \text{you're II} \]
\[c = \text{it's 30\textdegree} \]
Conditional independence?

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

(a is conditionally independent of b, given c)

“if you know c, then knowing a provides no additional information about b”

(I remembered my umbrella \(\perp\) the streets are wet | it’s raining)
(\text{feature}_i \perp \text{feature}_j | \text{label})

\implies

p(\text{feature}_i, \text{feature}_j | \text{label}) = p(\text{feature}_i | \text{label})p(\text{feature}_j | \text{label})
Naïve Bayes

\[
p(label|features) = \frac{p(y) \cdot p(x|y)}{p(x)}
\]
Naïve Bayes

\[
p(label|features) = \frac{p(label)p(features|label)}{p(features)}
\]

due to our conditional independence assumption:

\[
p(label|features) = \frac{p(label) \prod_i p(feature_i|label)}{p(features)}
\]
Naïve Bayes

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

\[ p(\neg label|features) = \frac{p(label) \prod_i p(feature_i|\neg label)}{p(features)} \]

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

\[ p(label|features) \ vs. \ p(\neg label|features) \]

both of which have the same denominator
Naïve Bayes

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

\[ p(label|features) \text{ vs. } p(\neg label|features) \]

both of which have the same denominator
Web Mining and Recommender Systems

Naïve Bayes – Worked Example
Amazon editorial descriptions:

Amazon.com Review
For most children, summer vacation is something to look forward to. But not for our 13-year-old 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 ceiling (and from officials at Hogwarts School of Witchcraft and Wizardry who strictly forbid students to go 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 rescued by a 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 Black is loose. Not only that, but he's after Harry Potter. But why? And why do the Dementors, the guards, unaffected? Once again, Rowling has created a mystery that will have children and adults cl

50k descriptions:
http://jmcauley.ucsd.edu/cse258/data/amazon/book_descriptions_50000.json
Example 1

\[
P(\text{book is a children’s book} \mid \text{“wizard” is mentioned in the description and “witch” is mentioned in the description})
\]

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
Q: What would happen if we trained two regressors, and attempted to “naively” combine their parameters?
Double-counting

\[ \text{length} = O_0 + O_1 \left[ \text{"wizards" } \cdot \text{ desc} \right] \]

\[ = O_0 + O_2 \left[ \text{"witches" } \cdot \text{ } \right] \]

\[ \text{length} = O_0 + O_1 + O_2 \]
A: Since both features encode essentially the same information, we’ll end up double-counting their effect.
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

$$y_i = \begin{cases} 1 & \text{if } X_i \cdot \theta > 0 \\ 0 & \text{otherwise} \end{cases}$$
Q: How to convert a real-valued expression \((X_i \cdot \theta \in \mathbb{R})\) into a probability \((p_\theta(y_i | X_i) \in [0, 1])\)
A: sigmoid function: \( \sigma(t) = \frac{1}{1+e^{-t}} \)
A: sigmoid function: $\sigma(t) = \frac{1}{1+e^{-t}}$

$\sigma(x; \theta) = \frac{1}{1+e^{-x \cdot \theta}}$
Logistic regression

**Training:**

\[ X_i \cdot \theta \] should be maximized when \( y_i \) is positive and minimized when \( y_i \) is negative.

\[
\arg \max_{\theta} \prod_{i} \left[ \delta(y_i=1) \sigma(x_i; \theta) + \delta(y_i=0) (1 - \sigma(x_i; \theta)) \right]
\]
Training:

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

$$\arg \max_{\theta} \prod_i \left[ \delta(y_i = 1)p_\theta(y_i | X_i) + \delta(y_i = 0)(1 - p_\theta(y_i | X_i)) \right]$$

$\delta(\text{arg}) = 1$ 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

\[ L_\theta(y|X) = \prod_{y_i=1} p_\theta(y_i|X_i) \prod_{y_i=0} (1 - p_\theta(y_i|X_i)) \]
Logistic regression

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

\[ \frac{\partial l}{\partial \theta_k} = \]
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:

\[ \frac{\partial l}{\partial \theta_k} = \sum_i X_{ik}(1 - \sigma(X_i \cdot \theta)) + \sum_{y_i=0} -X_{ik} - 2\lambda \theta_k \]
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
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
Q: Where would a logistic regressor place the decision boundary for these features?
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) $\theta x - \alpha = 0$
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

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

$$\arg \min_{\theta, \alpha, \xi > 0} \frac{1}{2} \| \theta \|_2^2 + C \sum_i \xi_i$$

such that

$$\forall_i y_i (\theta \cdot X_i - \alpha) \geq 1 - \xi_i$$
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)
Web Mining and Recommender Systems

Classification – Worked example
Judging a book by its cover

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

(code available on course webpage)
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
Previously...

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

\[
f(\text{data}) \rightarrow \text{labels} \rightarrow \{0,1\}, \{\text{True, False}\}, \{1, \ldots, N\}
\]
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

\[
p(label|features) = \frac{p(label)p(features|label)}{p(features)}
\]

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: \[ \sigma(t) = \frac{1}{1 + e^{-t}} \]
Q: Where would a logistic regressor place the decision boundary for these features?
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?

Want the margin to be as wide as possible

While penalizing points on the wrong side of it
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 $(feature_i \perp feature_j|label)$ 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
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
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

decision boundary

negative positive
Evaluating classifiers

TP (true positive): Labeled as positive, predicted as positive.
Evaluating classifiers

TN (true negative): Labeled as negative, predicted as negative
FP (false positive): Labeled as negative, predicted as positive
Evaluating classifiers

**FN (false negative):** Labeled as \( \text{positive} \), predicted as \( \text{negative} \)
Evaluating classifiers

<table>
<thead>
<tr>
<th>Prediction</th>
<th>Label</th>
<th>true</th>
<th>false</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true positive</td>
<td>false positive</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>false negative</td>
<td>true negative</td>
<td></td>
</tr>
</tbody>
</table>

Classification accuracy = correct predictions / #predictions

Error rate = incorrect predictions / #predictions
Evaluating classifiers

<table>
<thead>
<tr>
<th>Prediction</th>
<th>Label</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true positive</td>
</tr>
<tr>
<td></td>
<td>false</td>
<td>false positive</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false negative</td>
</tr>
<tr>
<td></td>
<td>false</td>
<td>true negative</td>
</tr>
</tbody>
</table>

True positive rate (TPR) = \( \frac{\text{true positives}}{\#\text{labeled positive}} \)

True negative rate (TNR) = \( \frac{\text{true negatives}}{\#\text{labeled negative}} \)
### Evaluating Classifiers

<table>
<thead>
<tr>
<th>Label</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true positive</td>
</tr>
<tr>
<td>false</td>
<td>false positive</td>
</tr>
</tbody>
</table>

**Balanced Error Rate (BER)**

\[
\text{BER} = \frac{1}{2} (\text{FPR} + \text{FNR})
\]

- \( \frac{1}{2} \) for a random/naïve classifier,
- 0 for a perfect classifier
Evaluating classifiers

\[
y = [1, -1, 1, 1, 1, -1, 1, 1, -1, 1]
\]

\[
\text{Confidence} = [1.3, -0.2, -0.1, -0.4, 1.4, 0.1, 0.8, 0.6, -0.8, 1.0]
\]
Evaluating classifiers

How to optimize a balanced error measure:

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

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

- Furthest from decision boundary in negative direction = lowest score/least confident
- Furthest from decision boundary in positive direction = highest score/most confident
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
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)
The classifiers we’ve seen can associate **scores** with each prediction

e.g.

\[ y = [ 1, -1, 1, 1, 1, -1, 1, 1, -1, 1, 1 ] \]

\[ \text{Confidence} = [1.3, -0.2, -0.1, -0.4, 1.4, 0.1, 0.8, 0.6, -0.8, 1.0] \]

Sort both according to confidence:
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 =
- Number of items we returned =
- Number of **relevant items** we returned =
The classifiers we’ve seen can associate **scores** with each prediction

\[
\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|} \\
\text{“fraction of retrieved documents that are relevant”}
\]

\[
\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|} \\
\text{“fraction of relevant documents that were retrieved”}
\]
Evaluating classifiers – ranking

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

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

e.g.
- Total number of relevant items = 7
- Number of items we returned = 6
- Number of relevant items we returned = 5

\[
\text{precision@6} = 
\]
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))
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
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.

Compute the true positive rate and true negative rate, and the F_1 score.
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?
3. When we only care about the “most confident” predictions

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

Compute the precision@k, and plot the signature of precision versus recall.
Web Mining and Recommender Systems

Classifier Evaluation: Worked Example
We'll look at a simple dataset from the UCI repository: [https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data](https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data)

```plaintext
@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,-66.52,0.34204,0.10949,0.57752,1.0381,0.32036,0.10949,0.1976,0.096885,0.10949,1475.2,0.24742,1.8027,0.10949,0.077287,50.199,1.1574,0.13523,0.062287,0.41949,0.32036,0.20912,1.0387,0.026093,6.1267,0.37788,0.077287,155.33,2.3498,0.24377,0.13523,1.4493,571.37,0.32101,0.095457,0.12879,0.11189,0.095457,127.3,377.096,0.45289,0.66883,54.621,0.10746,0.075859,1.0193,0.55407,0.42557,0.73717,0.73866,15182,0.080955,0.27543,0.91905,0.002024,7.2711,4.7343,142.76,2.5568,3.2597,0...
```

Did the company go bankrupt?

Code on course webpage
Web Mining and Recommender Systems

Supervised Learning: Summary so far
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

\[ f(\text{data}) \rightarrow \text{labels} \]
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:

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