Learning and Intelligent Systems

Practical issues:
Feature selection; **Imbalance**; Multi-class problems

Prof. Gunnar Rätsch
Biomedical Informatics ([bmi.inf.ethz.ch](http://bmi.inf.ethz.ch))
(Material: Andreas Krause)
Dealing with Imbalanced Data

What if the data set looks like this?
Sources of imbalanced data

- Fraud detection
- Spam Filtering
- Process monitoring
- Medical diagnosis
- Feedback in recommender systems
- ...
Cost Sensitive Perceptron

- Start at an arbitrary \( w_0 \in \mathbb{R}^d \)
- For \( t=1,2,... \) Do
  - Pick data point \((x', y') \in D\) from training set uniformly at random (with replacement), and set
  \[
  w_{t+1} = w_t - \eta_t \nabla \ell(w_t; x', y')
  \]

- Only difference: Use cost-sensitive loss function.
- For Perceptron:
  \[
  \ell(w; x, y) = c_y \max(0, -yw^T x)
  \]
  with parameters \( c_+, c_- > 0 \) controlling tradeoff
Cost-sensitive Perceptron loss

\[ \ell(w; x, y) = c_y \max(0, -yw^T x) \]

need to tune
Evaluating accuracy for imbalanced data

Suppose I claim to have a classifier with 90% accuracy on this data set. Is this good?
Evaluating accuracy for imbalanced data

- For imbalanced data, accuracy (i.e., fraction of correct classifications) is often not meaningful.
- It makes sense to distinguish (convention: + is rare class):

  ![Confusion Matrix](image)

- \( n = n_+ + n_- = p_+ + p_- \)
- \( \Sigma = n_+ \)
- \( \Sigma = n_- \)
Some metrics for imbalanced data

- **Accuracy:**
  \[
  \frac{TP + TN}{TP + TN + FP + FN} = \frac{\sum T \pm}{n_+ + n_-}
  \]

- **Precision:**
  \[
  \frac{TP}{TP + FP} = \frac{\sum T P}{n_+}
  \]
  \( \in [0, 1] \)

- **Recall:**
  \[
  \frac{TP}{TP + FN} = \frac{\sum T P}{n_+}
  \]
  \( \in [0, 1] \)

- **F1 score:**
  \[
  \frac{2TP}{2TP + FP + FN}
  \]
  Harmonic mean of prec. & recall

\[
\frac{2}{\frac{1}{\text{prec}} + \frac{1}{\text{rec}}} = \frac{2}{\frac{2TP}{n_+} + \frac{n_+}{TP + FN}}
\]
Trading false positives and false negatives

<table>
<thead>
<tr>
<th>Prec.</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>high (1)</td>
</tr>
<tr>
<td>high</td>
<td>high</td>
</tr>
<tr>
<td>high (1)</td>
<td>low</td>
</tr>
</tbody>
</table>

Highest F1 score
How to obtain tradeoff?

**Option 1:**
- Use **cost-sensitive classifier** (e.g., cost-sensitive Perceptron), and vary tradeoff parameter

**Option 2:**
- Find a **single classifier**, and vary classification threshold \( \tau \)

\[
y = \text{sign}(\mathbf{w}^T \mathbf{x} - \tau)
\]
Precision Recall Curve

[Davis & Goadrich, ICML'06]
More metrics for imbalanced data

- True positive rate (TPR) = recall!

\[
\frac{TP}{TP + FN}
\]

- False positive rate (FPR)

\[
\frac{FP}{TN + FP} = \frac{FP}{n_-}
\]

Several other metrics used!

Consider random classifier predicting + w.p. \( p \) indep of \( x \)

\[
TPR = \frac{p \cdot \frac{n_+}{n}}{p \cdot \frac{n_+}{n} + (1-p) \frac{n_-}{n}} = p
\]

\[
FPR = \frac{p \cdot \frac{n_-}{n}}{p \cdot \frac{n_-}{n} + (1-p) \frac{n_+}{n}} = p
\]
Receiver Operator Characteristic (ROC) Curve

[Davis & Goadrich, ICML’06]
Theorem [Davis & Goadrich ‘06]: Alg 1 dominates Alg 2 in terms of ROC Curve $\Leftrightarrow$ Alg 1 dominates Alg 2 in terms of Precision Recall curves
Area under the Curve

- Often want to compare the ability of classifiers to provide imbalanced classification.
- Can compute Area under the ROC or Precision Recall curves.
What you need to know

- Basic techniques for handling unbalanced data
  - Upsampling, downsampling
- Cost-sensitive loss functions
  - Cost sensitive Perceptron
- Evaluating classifiers on imbalanced data sets
  - Metrics (precision, recall etc.)
  - ROC / Precision Recall curves
Learning and Intelligent Systems

Practical issues:
Feature selection; Imbalance; Multi-class problems

Prof. Andreas Krause
Learning and Adaptive Systems (las.ethz.ch)
Dealing with multiple classes

Given: \( D = \{(x_1, y_1), \ldots, (x_n, y_n)\} \)

Want: \( f : \mathcal{X} \rightarrow \mathcal{Y} \)

So far, discussed binary methods. Do we have to invent something new for multiclass?
One-vs-all

- Solve $c$ binary classifiers, one for each class
  - Positive examples: all points from class $i$
  - Negative examples: all other points
- Classify using the classifier with largest confidence

\[
f_{i, c} : X \to \mathbb{R}
\]

\[
y = \arg\max_{i \in \{1 \ldots c\}} f_i(x)
\]

E.g., $f_i(x) = \frac{w_i^T x}{\|w_i\|_2}$
Confidence in classification

"Confidence": \( w^T x \)

depends on scale of \( w \)

Soc: replace \( w \) by

\( \tilde{w} = 10^6 \cdot w \)

One possible solution:

Normalize!

\( \tilde{w} = \frac{w}{\|w\|_2} \)
One-vs-all (OvA) decision boundary
Challenges with one-vs-all

- Only works well if classifiers produce confidence scores on the „same scale“
- Individual binary classifiers see imbalanced data, even if the whole data set is balanced
- One class might not be linearly separable from all other classes
One-vs-one

- Train $c(c-1)/2$ binary classifiers, one for each pair of classes (i,j)
  - Positive examples: all points from class $i$
  - Negative examples: all points from class $j$

- Apply voting scheme
  - Class with highest number of positive prediction wins
Comparison: one-vs-all and one-vs-one

<table>
<thead>
<tr>
<th>Method</th>
<th>One-vs-all</th>
<th>One-vs-one</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advantages</td>
<td>Only $c$ classifiers needed (faster!)</td>
<td>No confidence needed</td>
</tr>
<tr>
<td>Disadvantages</td>
<td>Requires confidence in prediction / leads to class imbalance</td>
<td>Slower (need to train $c^*(c-1)/2$ models)</td>
</tr>
</tbody>
</table>
Alternative methods

- Other encodings
  - E.g., error correcting output codes

- Explicit multi-class models
  - E.g., multi-class Perceptron / SVM etc. (not discussed here)
  - Some models are naturally multi-class (e.g., generative probabilistic models, see later)

- Often one-vs-all / one-vs-one is hard to beat
How many binary classifiers do we need?

- One-vs-all: \( c \)
- One-vs-one: \( \frac{c(c-1)}{2} \)

Can we get away with less?

<table>
<thead>
<tr>
<th>Class</th>
<th>Binary encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00 - 00</td>
</tr>
<tr>
<td>1</td>
<td>00 - 01</td>
</tr>
<tr>
<td>2</td>
<td>00 - 10</td>
</tr>
<tr>
<td>3</td>
<td>00 - 11</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>C-1</td>
<td>1 1 1 1</td>
</tr>
</tbody>
</table>

Job of i-th classifier: predict i-th bit!
Multi-class vs. coding

- Can in principle view multi-classification as „decoding“ the class label
  - Each classifier predicts one bit
- Might be able to get away using $O(\log c)$ classifiers!

- Can use ideas from coding theory to do multi-class classification
## Example: Error correcting output codes

<table>
<thead>
<tr>
<th>Class label</th>
<th>Binary encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1 -1 -1 -1</td>
</tr>
<tr>
<td>2</td>
<td>+1 +1 -1 -1</td>
</tr>
<tr>
<td>3</td>
<td>-1 -1 +1 +1</td>
</tr>
<tr>
<td>4</td>
<td>+1 -1 +1 -1</td>
</tr>
<tr>
<td>5</td>
<td>+1 +1 +1 +1</td>
</tr>
</tbody>
</table>
Example: Error correcting output codes

<table>
<thead>
<tr>
<th>Class label</th>
<th>Binary encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1 -1 -1 -1 -1 -1</td>
</tr>
<tr>
<td>2</td>
<td>+1 +1 +1 -1 -1 -1</td>
</tr>
<tr>
<td>3</td>
<td>-1 -1 -1 +1 +1 +1</td>
</tr>
<tr>
<td>4</td>
<td>+1 -1 +1 -1 +1 -1</td>
</tr>
<tr>
<td>5</td>
<td>+1 +1 +1 +1 +1 +1</td>
</tr>
</tbody>
</table>

Prediction + - - - - -
What you need to know

- Using binary classification for multi-class problems (One-vs-all, one-vs-one)
- Benefits of the respective methods
Supervised learning big picture so far

Kernelized Regression

L1-Regression (Lasso)

Ridge Regression

Least squares Regression

Loss funct.

k-NN

"Special case"

Linear SVM

Kernelized SVM

Kernelized Perceptron

Perceptron

Loss funct.

I1-SVM

I2-regularizer

I2-regular.
Multi-classification big picture

- Multi-classification via OvA/OvO
- Kernelized Perceptron
- Multiclass Perceptron
- Perceptron
- k-NN
- "Special case" (k-NN)
- Kernelized SVM
- Linear SVM
- Multiclass SVM
- I1-SVM
- I1-regular.
- Loss funct.
- I2-regularizer
- Loss funct.
Supervised learning summary so far

**Representation/features**: Linear hypotheses; nonlinear hypotheses using kernels

**Model/objective**: Loss-function + Regularization

- Squared loss, 0/1 loss, Perceptron loss, Hinge loss, cost-sensitive losses, structured hinge loss, Bayesian expected loss, ...
- \( L^2 \) norm, \( L^1 \) norm, mixed norms, total variation, ...

**Method**: Exact solution, Gradient Descent, SGD, Convex Programming, Sampling, Dynamic programming, ...

**Model selection**: Cross-Validation, Bayes factor, Minimum description length, ...