Advanced Machine Learning

Imbalanced Learning via Cost-Sensitive Learning

Confusion matrix

		True class	
		$y = 1$	$y = -1$
Pred.	$\hat{y}$ = 1	TP	FP
class	$\hat{y}$ = -1	FN	TN

Cost matrix

		True class	
		$y = 1$	$y = -1$
Pred.	$\hat{y}$ = 1	$C(1, 1)$	$C(1, -1)$
class	$\hat{y}$ = -1	$C(-1, 1)$	$C(-1, -1)$

Learning goals

- Learn the modelling approach via the cost matrix in cost-sensitive learning settings
- Understand the connection between cost-sensitive and imbalanced learning
- Get to know MetaCost as a general approach to make classifiers cost-sensitve

COST-SENSITIVE LEARNING: IN A NUTSHELL

- Cost-sensitive learning is a learning paradigm, where different (mis-)classification costs are taken into consideration and the learner seeks to minimize the total costs in expectation.
- Thus, the major difference to the “classical” cost-insensitive learning setting is that cost-sensitive learning deals differently with misclassifications. Most of the algorithms of the latter kind assume that the data sets are balanced, and all errors have the same cost.
- This is motivated by a plethora of real-world applications, where different costs between misclassifications are present:
  - Medicine — Misdiagnosing a cancer patient as healthy vs. misdiagnosing a healthy patient as having cancer (and then check again).
  - Tracking criminals — Classify an innocent person as a terrorist vs. overlooking a terrorist.
  - (Extreme) Weather prediction — Incorrectly predicting that no hurricane occurs vs. predicting a strong wind as a hurricane.
  - Credit granting — Lending to a risky client vs. not lending to a trustworthy client.
  - . . .
- In all these examples the costs of a false negative is much higher than the costs of a false positive.

COST MATRIX

- In cost-sensitive learning we are provided with a cost matrix **C** of the form

		True Class y			
		1	2	...	g
Classification	1	$C(1,1)$ (True 1's)	$C(1,2)$ (False 1's for 2's)	...	$C(1,g)$ (False 1's for g's)
$\hat{y}$	2	$C(2,1)$ (False 2's for 1's)	$C(2,2)$ (True 2's)	...	$C(2,g)$ (False 2's for g's)
	$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
	g	$C(g,1)$ (False g's for 1's)	$C(g,2)$ (False g's for 2's)	...	$C(g,g)$ (True g's)

- Here, $C(i,j)$ is the cost of classifying j as i, which in the cost-insensitive learning case is simply $C(i,j) = \mathbb{1}_{[i \neq j]}$, i.e., each misclassification has the same cost of 1.
- Sometimes the cost matrix is provided by the application at hand, e.g. in the credit granting example, or provided by a domain expert. In many cases, however, the cost matrix needs to be estimated. This is usually done by using a heuristic or by learning a proper cost matrix from the training data.
- The cost matrix is essential for the learning process, as
  1. too low costs might not change the decision boundaries significantly leading to (still) costly predictions,
  2. too high costs might harm the generalization capability of the classifier on costly classes.

COST MATRIX FOR IMBALANCED LEARNING

- For imbalanced data sets biases towards majority classes can be enlarged or even too strong biases towards the minority can be created if the cost matrix is set inappropriately.
- A common heuristic for imbalanced data sets is to use
  - the imbalance ratio between majority and minority classes for misclassifying a minority class j as a majority class i, i.e., $C(i,j) = \frac{n_i}{n_j}$ for classes i and j such that $n_j \ll n_i$,
  - and costs of 1 for misclassifying a majority class j as a minority class i, i.e., $C(i,j) = 1$ for classes i and j such that $n_i \ll n_j$.
  - Usually, the costs of a correct classification is set to 0, which is justified also from a theoretical point of view as we will see later.
- In an imbalanced binary classification problem we obtain the following cost matrix using this heuristic:

	True class	
	$y = 1$	$y = -1$
Pred. $\hat{y}$ = 1	0	1
class $\hat{y}$ = -1	$\frac{n_-}{n_+}$	0

- Thus, this heuristic is consistent with the general real case that the cost of false negatives is much higher than the cost of false positives.

MINIMUM EXPECTED COST PRINCIPLE

- Suppose we are provided with a cost matrix **C** and also we have knowledge of the true posterior distribution $p(\cdot \mid \mathbf{x})$. The most natural way to classify a given feature $\mathbf{x}$ is then by following the *minimum expected cost principle*.
- Minimum expected cost principle: Use the class for prediction with the smallest expected costs, where the expected costs of a class $i \in \{1, \dots, g\}$ is

$$\mathbb{E}_{J \sim p(\cdot \mid \mathbf{x})}(C(i, J)) = \sum_{j=1}^{g} p(j \mid \mathbf{x}) C(i, j).$$

- Thus, if we have a classifier f which uses a probabilistic score function $\pi : \mathcal{X} \to [0,1]^g$ with $\pi(\mathbf{x}) = (\pi(\mathbf{x})_1, \dots, \pi(\mathbf{x})_g)^\top$ and $\sum_{j=1}^{g} \pi(\mathbf{x})_j = 1$ for the classification, then one can easily modify f to take the expected costs into account:

$$\tilde{f}(\mathbf{x}) = \arg\min_{i=1,\dots,g} \sum_{j=1}^{g} \pi(\mathbf{x})_j C(i, j).$$

MINIMUM EXPECTED COST PRINCIPLE: BINARY CASE

- In the binary classification setting (i.e., $\mathcal{Y} = \{-1, 1\}$) the minimum expected costs principle translates to predict the positive class if

$$\begin{aligned}
&\mathbb{E}_{J\sim p(\cdot \mid \mathbf{x})}(C(1, J)) \leq \mathbb{E}_{J\sim p(\cdot \mid \mathbf{x})}(C(-1, J)) \\
&\Leftrightarrow p(-1 \mid \mathbf{x})C(1, -1) + p(1 \mid \mathbf{x})C(1, 1) \\
&\qquad \leq p(-1 \mid \mathbf{x})C(-1, -1) + p(1 \mid \mathbf{x})C(-1, 1) \\
&\Leftrightarrow p(-1 \mid \mathbf{x})\left(C(1, -1) - C(-1, -1)\right) \leq p(1 \mid \mathbf{x})\left(C(-1, 1) - C(1, 1)\right)
\end{aligned}$$

- Note that the decision for predicting the positive class does not change if we use instead of $\mathbf{C}$ the simpler cost matrix $\mathbf{C}_{simple}$, where
  - $C_{simple}(-1, -1) = C_{simple}(1, 1) = 0$
  - $C_{simple}(1, -1) = C(1, -1) - C(-1, -1)$
  - $C_{simple}(-1, 1) = C(-1, 1) - C(1, 1)$.

MINIMUM EXPECTED COST PRINCIPLE: BINARY CASE

- Thus, one can assume without loss of generality that the cost matrix **C**

		True class	
		$y = 1$	$y = -1$
Pred.	$\hat{y}$ = 1	$C(1, 1)$	$C(1, -1)$
class	$\hat{y}$ = -1	$C(-1, 1)$	$C(-1, -1)$

is of a simpler form $\mathbf{C}_{simple}$:

		True class	
		$y = 1$	$y = -1$
Pred.	$\hat{y}$ = 1	0	$C(1, -1) - C(-1, -1)$
class	$\hat{y}$ = -1	$C(-1, 1) - C(1, 1)$	0

- An analogous result can be shown for the multiclass setting.

MINIMUM EXPECTED COST PRINCIPLE: BINARY CASE

- With this simpler cost matrix (relabeling $\mathbf{C}$ by $\mathbf{C}_{simple}$), the decision to predict the positive class boils down to

$$
\begin{aligned}
& p(-1 \mid \mathbf{x})C(1,-1) \leq p(1 \mid \mathbf{x})C(-1,1) \\
& \Leftrightarrow (1 - p(1 \mid \mathbf{x}))C(1,-1) \leq p(1 \mid \mathbf{x})C(-1,1) \\
& \Leftrightarrow \underbrace{\frac{C(1,-1)}{C(1,-1) + C(-1,1)}}_{=:c^*} \leq p(1 \mid \mathbf{x})
\end{aligned}
$$

- This yields the optimal threshold value c^* for probabilistic score classifiers, so that any probabilistic classifier f using a probabilistic score $\pi : \mathcal{X} \rightarrow [0,1]$ can be modified to

$$\tilde{f}(\mathbf{x}) = 2 \cdot \mathbb{1}_{[\pi(\mathbf{x}) \geq c^*]} - 1.$$

METACOST: OVERVIEW

- Substantial work has gone into making individual algorithms cost-sensitive. A better solution would be to have a procedure that can convert a broad variety of cost-insensitive classifiers into cost-sensitive ones.
- MetaCost is a wrapper method which can be used for *any* type of classifier to obtain a cost-sensitive classifier. It treats the underlying classifier as a black-box in the sense that neither knowledge about its mechanism is required nor changes to its mechanism is needed.
- MetaCost needs only a cost-matrix **C** for the underlying learning setting, which is used to adapt the decision boundaries towards predictions with low expected costs.

  (Some tuning parameters are also needed.)
- Roughly speaking, the procedure of MetaCost is: relabel the training examples with their "optimal" classes, i.e., the ones with low expected costs, and apply the classifier on the relabeled data set.

METACOST: ALGORITHM

The procedure of MetaCost is divided into three phases:

1. Bagging — The underlying classifier is used (trained) L times on different bootstrapped samples of the training data, respectively.
2. Relabeling — These L trained classifiers are used to relabel the original training data set by taking the cost-matrix into account.
3. Cost-sensitivity — The classifier is trained on the relabeled data set resulting in a cost-sensitive classifier.

Predictions of the classifier f are converted (if necessary) into probabilistic prediction:

$$\text{ProbPrediction}(j, f, \mathbf{x}) = \begin{cases} (f(\mathbf{x}))_j & f \text{ is a prob. classifier,} \\ \text{One-hot}(f(\mathbf{x})) & \text{else,} \end{cases}$$

where One-hot$(f(\mathbf{x}))$ uses a one-hot-encoding of the prediction to make it a probability, i.e., one for the predicted class and 0 else.

MetaCost
Input: $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^n$ training data,
$L \in \mathbb{N}$ number of bagging iterations,
$B \in \mathbb{N}$ bootstrap size,
f (black-box) classifier, $\mathbf{C}$ cost matrix,
\# 1st phase:
for $l = 1, \ldots, L$ **do**
  $\mathcal{D}_l \leftarrow$ BootstrapSample ($\mathcal{D}$, B)
  $f_l \leftarrow$ train f on $\mathcal{D}_l$
end for
\# 2nd phase:
for $i = 1, \ldots, n$ **do**
  if $\mathbf{x}^{(i)} \in \mathcal{D}_l$ for all $l = 1, \ldots, L$ **then**
    $\tilde{L} \leftarrow \{1, \ldots, L\}$
  else
    $\tilde{L} \leftarrow \bigcup_{l:\mathbf{x}^{(i)} \notin \mathcal{D}_l} \{l\}$
  end if
  for $j = 1, \ldots, g$ **do** (relabel for binary case)
    $p_j(\mathbf{x}^{(i)}) \leftarrow \frac{1}{|\tilde{L}|} \sum_{l \in \tilde{L}} p_j(\mathbf{x}^{(i)} \mid f_l)$
    $p_j(\mathbf{x}^{(i)} \mid f_l) = \text{ProbPrediction}(j, f_l, \mathbf{x}^{(i)})$
    $\tilde{y}^{(i)} \leftarrow \arg\min_{i*} \sum_{j=1}^{g} p_j(\mathbf{x}^{(i)}) C(i^*, j)$
  end for
  $\tilde{D} \leftarrow \tilde{D} \cup \{(\mathbf{x}^{(i)}, \tilde{y}^{(i)})\}$
end for
\# 3rd phase:
$f_{meta} \leftarrow$ train f on $\tilde{D}$