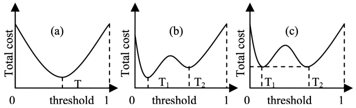
Applied Machine Learning

Imbalanced Data: Cost-Sensitive Learning



Learning goals

- Understand cost-sensitive learning principles
- Apply cost matrices to imbalanced classification problems
- Implement threshold tuning

Cost-Sensitive Learning

COST-SENSITIVE LEARNING: IN A NUTSHELL

- Cost-sensitive learning:
  - Classical learning: data sets are balanced, and all errors have equal costs
  - We now assume given, unequal cost
  - And try to minimize them in expectation
- Applications:
  - Medicine — Misdiagnosing as healthy vs. having a disease
  - (Extreme) Weather prediction — Incorrectly predicting that no hurricane occurs
  - Credit granting — Lending to a risky client vs. not lending to a trustworthy client.

		Truth	
		Default	Pays Back
Pred.	Default	0	10
	Pays Back	1000	0

- In these examples, **the costs of a false negative is much higher than the costs of a false positive**.
- In some applications, the costs are **unknown** $\rightsquigarrow$ need to be specified by experts, or be learnt.

COST MATRIX

- Input: cost matrix **C**

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

- $C(j,k)$ is the cost of classifying class k as j,
- 0-1-loss would simply be: $C(j,k) = \mathbb{1}_{[j \neq k]}$
- **C** designed by experts with domain knowledge
  1. Too low costs: not enough change in model, still costly errors
  2. Too high costs: might never predict costly classes

COST MATRIX FOR IMBALANCED LEARNING

- Common heuristic for imbalanced data sets:
  - $C(j,k) = \frac{n_j}{n_k}$ with $n_k \ll n_j$,
    misclassifying a minority class k as a majority class j
  - $C(j,k) = 1$ with $n_j \ll n_k$,
    misclassifying a majority class k as a minority class j
  - 0 for a correct classification

- Imbalanced binary classification:

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

- So: much higher costs for FNs

MINIMUM EXPECTED COST PRINCIPLE

- Suppose we have:
  - a cost matrix **C**
  - knowledge of the true posterior $p(\cdot \mid \mathbf{x})$
- Predict class j with smallest expected costs when marginalizing over true classes:

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

- If we trust a probabilistic classifier, we can convert its scores to labels:

$$h(\mathbf{x}) := \arg\min_{j=1,\dots,g} \sum_{k=1}^{g} \pi_k(\mathbf{x}) C(j, k).$$

- Can be better to take a less probable class ▸ "Elkan et al." 2001

OPTIMAL THRESHOLD FOR BINARY CASE 1/2

- Optimal decisions do not change if
  - **C** is multiplied by positive constant
  - **C** is added with constant shift
- Scale and shift **C** to get simpler **C**$'$:

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

where

- $C'(-1,1) = \frac{C(-1,1)-C(-1,-1)}{C(1,-1)-C(-1,-1)}$
- $C'(1,1) = \frac{C(1,1)-C(-1,-1)}{C(1,-1)-C(-1,-1)}$

OPTIMAL THRESHOLD FOR BINARY CASE 2/2

- We predict $\mathbf{x}$ as class 1 if

$$\mathbb{E}_{K \sim p(\cdot \mid \mathbf{x})}(C'(1, K)) \leq \mathbb{E}_{K \sim p(\cdot \mid \mathbf{x})}(C'(-1, K))$$

- Let's unroll the expected value and use $\mathbf{C}'$:

$$\begin{aligned}
&p(-1 \mid \mathbf{x})C'(1,-1) + p(1 \mid \mathbf{x})C'(1,1) \leq p(-1 \mid \mathbf{x})C'(-1,-1) + p(1 \mid \mathbf{x})C'(-1,1) \\
&\Rightarrow [1 - p(1 \mid \mathbf{x})] \cdot 1 + p(1 \mid \mathbf{x})C'(1,1) \leq p(1 \mid \mathbf{x})C'(-1,1) \\
&\Rightarrow p(1 \mid \mathbf{x}) \geq \frac{1}{C'(-1,1) - C'(1,1) + 1} \\
&\Rightarrow p(1 \mid \mathbf{x}) \geq \frac{C(1,-1) - C(-1,-1)}{C(-1,1) - C(1,1) + C(1,-1) - C(-1,-1)} = c^*
\end{aligned}$$

- If even $C(1,1) = C(-1,-1) = 0$, we get:

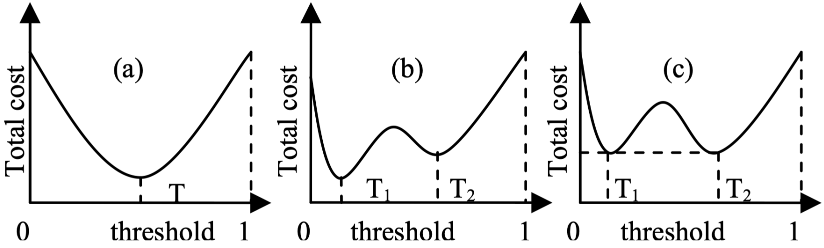
$$p(1 \mid \mathbf{x}) \geq \frac{C(1,-1)}{C(-1,1) + C(1,-1)} = c^*$$

- Optimal threshold c^* for probabilistic classifier

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

EMPIRICAL THRESHOLDING: BINARY CASE

- Theoretical threshold from MECP not always best, due to e.g. wrong model class, finite data, etc.
- Simply measure costs on data with different thresholds
- Then pick best threshold ▸ "Sheng et al." 2006:



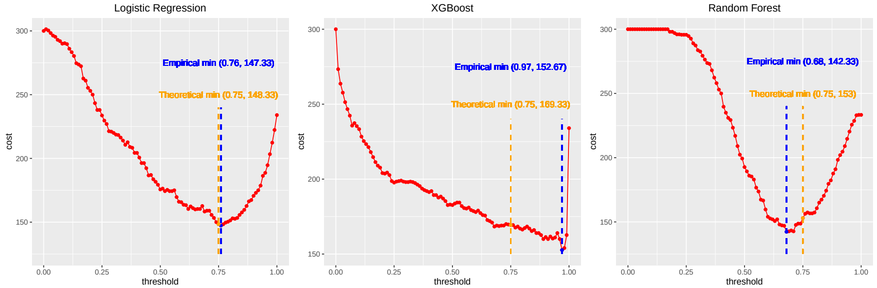
- What if two equal local minima? We prefer the one with wider span
- Do this on validation data / over cross-val to avoid overfitting!

EMPIRICAL THRESHOLDING: BINARY CASE

- Example: German Credit task

		True class	
		$y = \text{good}$	$y = \text{bad}$
Pred.	$\hat{y}$ = good	0	3
class	$\hat{y}$ = bad	1	0

- Theoretical: $C(good, bad)/(C(bad, good) + C(good, bad)) = 3/4 = c^*$
- Empirical version with 3-CV: For XGBoost, empirical minimum deviates substantially from theoretical version



EMPIRICAL THRESHOLDING: MULTICLASS

- In the standard setting, we predict class $h(\mathbf{x}) = \arg\max_k \pi_k(\mathbf{x})$.
- Let's use g thresholds c_k now
- Re-scale scores $\mathbf{s} = (\frac{\pi(\mathbf{x})_1}{c_1}, \dots, \frac{\pi(\mathbf{x})_g}{c_g})^\top$,
- Predict class $\arg\max_k \pi_k(\mathbf{x})$.
- Compute empirical costs over cross-validation
- Optimize over g (actually: $g - 1$) dimensional threshold vector $(c_1, \dots, c_g)^T$ to produce minimal costs

BINARY INSTANCE-SPECIFIC COST LEARNING

- Assumes instance-specific costs for every observation: $\mathcal{D}^{(n)} = \{(\mathbf{x}^{(i)}, \mathbf{c}^{(i)})\}_{i=1}^{n}$, where $(\mathbf{x}^{(i)}, \mathbf{c}^{(i)}) \in \mathbb{R}^p \times \mathbb{R}^2$.
- Define "true class" as cost minimal class
- Define observation weights: $|\mathbf{c}^{(i)}[1] - \mathbf{c}^{(i)}[0]|$

	$\mathbf{c}^{(i)}[0]$	$\mathbf{c}^{(i)}[1]$	$y^{(i)}$	$w^{(i)}$
$\mathbf{x}^{(1)}$	1	1	0	0
$\mathbf{x}^{(2)}$	1	2	0	1
$\mathbf{x}^{(3)}$	7	3	1	4

- Now solve weighted ERM:

$$\mathcal{R}_{emp}(\boldsymbol{\theta}) = \sum_{i=1}^{n} w^{(i)} L\left(y^{(i)}, f\left(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}\right)\right)$$

- NB: Instances with equal costs are effectively ignored.

MULTICLASS COSTS

- Consider $g > 2$. Vanilla CSL is special case of instance specific, use $\mathbf{c}^{(i)}$ same for all $\mathbf{x}^{(i)}$ of the same class
- For two $\mathbf{x}^{(i)}$ with $y = 2$ and $y = 3$:

	$\mathbf{c}^{(i)}[1]$	$\mathbf{c}^{(i)}[2]$	$\mathbf{c}^{(i)}[3]$	$y^{(i)}$
$\mathbf{x}^{(1)}$	1	0	1	2
$\mathbf{x}^{(2)}$	3	1	0	3
$\mathbf{x}^{(3)}$	1	0	1	2

- Set $\mathbf{c}^{(i)}[y^{(i)}] = 0$, i.e. zero-cost for correct prediction.

CSOVO

- Let $\mathcal{D}^{(n)} = \{(\mathbf{x}^{(i)}, \mathbf{c}^{(i)})\}_{i=1}^{n}$, $(\mathbf{x}^{(i)}, \mathbf{c}^{(i)}) \in \mathbb{R}^p \times \mathbb{R}^g$.
- Example:

	$\mathbf{c}^{(i)}[1]$	$\mathbf{c}^{(i)}[2]$	$\mathbf{c}^{(i)}[3]$
$\mathbf{x}^{(1)}$	0	2	3
$\mathbf{x}^{(2)}$	1	0	1
$\mathbf{x}^{(3)}$	2	0	3

- Idea: Reduction principle to binary case (weighted fit) by one-versus-one (OVO).
- For class j vs. k:
  - How to deal with the label $y^{(i)}$? $y^{(i)}$ can be neither j nor k.
  - How to deal with the costs $\mathbf{c}^{(i)}[j]$ and $\mathbf{c}^{(i)}[k]$?

CSOVO

- When training a binary classifier $f^{(j,k)}$ for class j vs. k,
  - Choose cost min class from pair $\arg\min_{l\in\{j,k\}} \mathbf{c}^{(i)}[l]$ as ground truth
  - Sample weight is simply diff between the 2 costs $|\mathbf{c}^{(i)}[j] - \mathbf{c}^{(i)}[k]|$
- Example continued:

	$\mathbf{c}^{(i)}[1]$	$\mathbf{c}^{(i)}[2]$	$\mathbf{c}^{(i)}[3]$	$\mathbf{c}^{(i)}[1 \text{ vs } 2]$	$\tilde{y}^{(i)}[1 \text{ vs } 2]$	$w^{(i)}[1 \text{ vs } 2]$
$\mathbf{x}^{(1)}$	0	2	3	0/2	1	2
$\mathbf{x}^{(2)}$	1	0	1	1/0	2	1
$\mathbf{x}^{(3)}$	2	0	3	2/0	2	2

	$\mathbf{c}^{(i)}[1]$	$\mathbf{c}^{(i)}[2]$	$\mathbf{c}^{(i)}[3]$	$\mathbf{c}^{(i)}[2 \text{ vs } 3]$	$\tilde{y}^{(i)}[2 \text{ vs } 3]$	$w^{(i)}[2 \text{ vs } 3]$
$\mathbf{x}^{(1)}$	0	2	3	2/3	2	1
$\mathbf{x}^{(2)}$	1	0	1	0/1	2	1
$\mathbf{x}^{(3)}$	2	0	3	0/3	2	3

CSOVO

- Example continued

	$\mathbf{c}^{(i)}[1]$	$\mathbf{c}^{(i)}[2]$	$\mathbf{c}^{(i)}[3]$	$\mathbf{c}^{(i)}[1 \text{ vs } 3]$	$\tilde{y}^{(i)}[1 \text{ vs } 3]$	$w^{(i)}[1 \text{ vs } 3]$
$\mathbf{x}^{(1)}$	0	2	3	0/3	1	3
$\mathbf{x}^{(2)}$	1	0	1	-/-	-	0
$\mathbf{x}^{(3)}$	2	0	3	2/3	1	1

- Wrap everything up:
  1. For class j vs. k, transform all $(\mathbf{x}^{(i)}, \mathbf{c}^{(i)})$ to $(\mathbf{x}^{(i)}, \arg\min_{l \in \{j,k\}} \mathbf{c}^{(i)}[l])$ with sample-wise weight $|\mathbf{c}^{(i)}[j] - \mathbf{c}^{(i)}[k]|$.
  2. Train a weighted binary classifier $f^{(j,k)}$ using the above
  3. Repeat step 1 and 2 for different (j, k).
  4. Predict using the votes from all $f^{(j,k)}$.
- Theoretical guarantee:
  test costs of final classifier $\leq 2 \sum_{j<k}$ test cost of $f^{(j,k)}$.