Introduction to Machine Learning

Tutorial VI

Joanna Ficek

Disclaimer

The slides may vary slightly from what has been discussed during the tutorial to incorporate some of the student's questions as well as some aspects discussed on the black-board only. The new slides have the title background colored in grey.

Outline

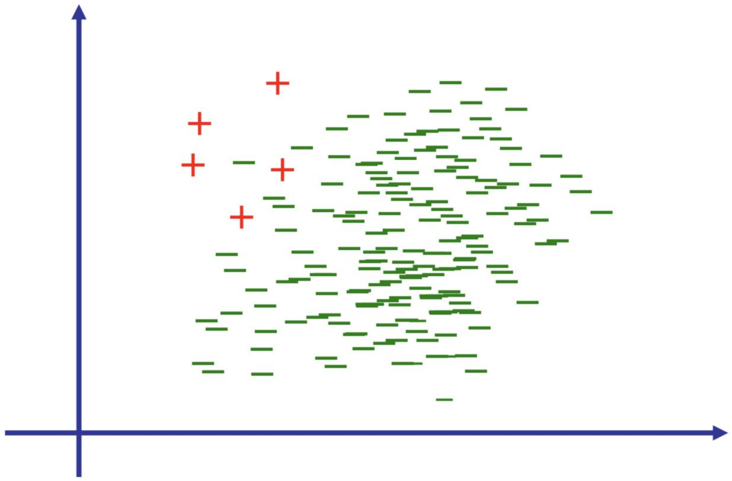
1. Class imbalance
   a. Problem setting
   b. Simple solutions
   c. Cost-sensitive algorithms
2. Evaluation
   a. Evaluation metrics in standard case
   b. Evaluation metrics in case of imbalanced classes
3. Multi-class SVM
   a. Naive approaches
   b. Multi-class Hinge loss
   c. Evaluation
4. Outlook: cross-entropy loss

Outline

1. Class imbalance
   a. Problem setting
   b. Simple solutions
   c. Cost-sensitive algorithms
2. Evaluation
   a. Evaluation metrics in standard case
   b. Evaluation metrics in case of imbalanced classes
3. Multi-class SVM
   a. Naive approaches
   b. Multi-class Hinge loss
   c. Evaluation
4. Outlook: cross-entropy loss

Class imbalance: problem setting

- Binary classification: one class substantially outnumbers the other class (wlog. positive class is rare)
- Examples
  - Diagnosis of rare diseases
  - Spam detection
  - Image detection: identification of rare fish population
- Why problematic?
  - Minority class contributes little to the empirical risk
  - Algorithms tend to predict negative class only
  - Evaluation of algorithm’s performance is deceiving



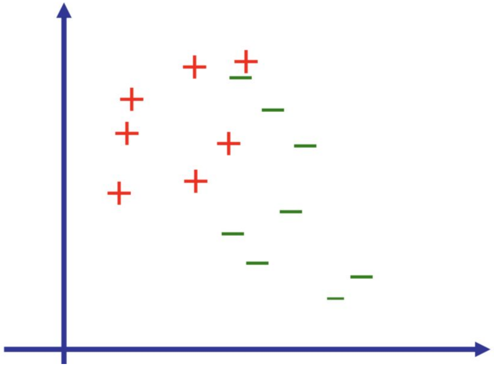
Class imbalance: simple approaches

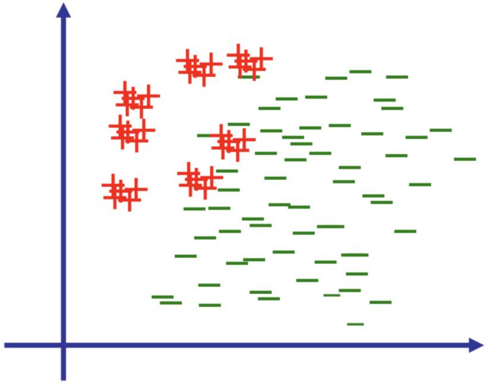
- Undersampling of the majority class
  - faster (+)
  - waste of available data (-)
  - loss of information about majority class (-)

- Upsampling of the minority class
  - uses all data (+)
  - risk of overfitting the minority class (-)
  - arbitrarily chosen data perturbation (-)

- => Naive solutions, but still used in practice

- Many extensions including generation of synthetic data exist, see Elrahman and Abraham (2013) A Review of Class Imbalance Problem, *JNIC* 1: 332-340





Class imbalance: cost-sensitive algorithms

- Idea: modify the loss function to account for class imbalance

- E.g. cost-sensitive Hinge-loss

$$\ell_{CS-H}(\mathbf{w};\mathbf{x},y) = c_y \max(0, 1 - y\mathbf{w}^T\mathbf{x})$$

- How does introduction of cost-sensitivity influence the loss function?
  - It changes the slope of the loss function

- How does introduction of cost-sensitivity influence the gradient?
  - It scales the gradient

Class imbalance: cost-sensitive algorithms (2)

Clarification:

- Wlog. the positive class in the minority class
- To avoid redundancy, instead of using two cost-parameters, it is sufficient to combine them in a ratio $c = \frac{c_+}{c_i}$ (see lecture slides p.12). Then the resulting cost-sensitive loss function can be decomposed as

$$l_c(\mathbf{w}; \mathbf{x}_i, y_i, c) = \begin{cases} 0 & \text{if } \mathbf{w}^T\mathbf{x}_i y_i \geq 0 \text{ (correctly classified)} \\ -c y_i \mathbf{w}^T\mathbf{x}_i & \text{if } \mathbf{w}^T\mathbf{x}_i y_i < 0 \wedge y_i = 1 \quad \text{(false negatives)} \\ -y_i \mathbf{w}^T\mathbf{x}_i & \text{if } \mathbf{w}^T\mathbf{x}_i y_i < 0 \wedge y_i = -1 \quad \text{(false positives)} \end{cases}$$

- Now, the slope of the loss changes only for the minority class after reparametrization (see lecture slides p.10, green corresponds to the minority-class loss and black to the majority-class loss)

Outline

1. Class imbalance
   a. Problem setting
   b. Simple solutions
   c. Cost-sensitive algorithms
2. Evaluation
   a. Evaluation metrics in standard case
   b. Evaluation metrics in case of imbalanced classes
3. Multi-class SVM
   a. Naive approaches
   b. Multi-class Hinge loss
   c. Evaluation
4. Outlook: cross-entropy loss

Evaluation: standard case

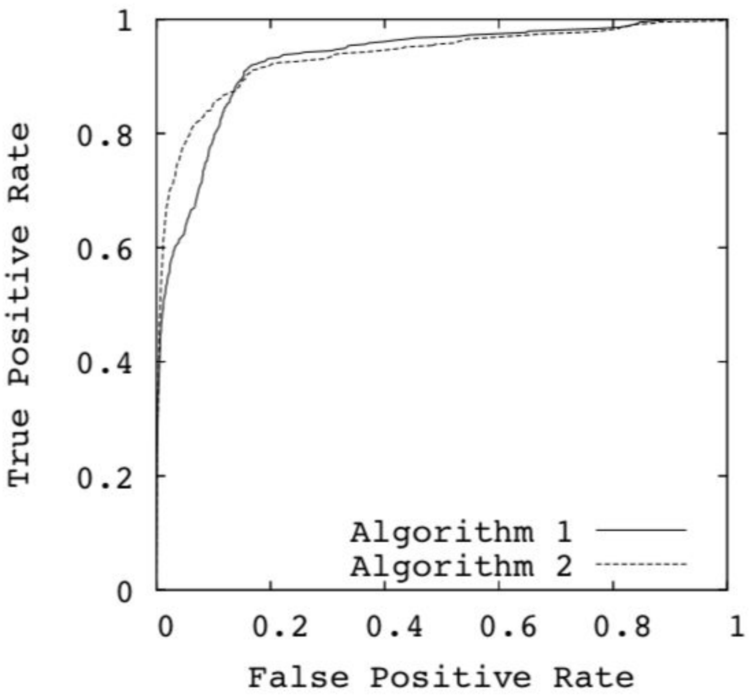
True labels

Predicted labels

	Positive	Negative
Positive	TP	FP
Negative	FN	TN

- Accuracy: (TP + TN)/n
- TPR: TP/(TP+FN)
- FPR: FP/(FP+TN)

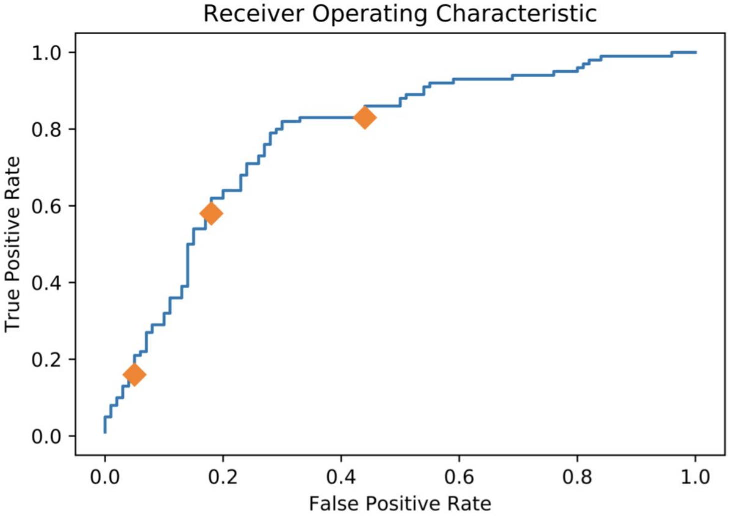
Evaluation: ROC curve



Davis and Goadrich
(ICML 2006)

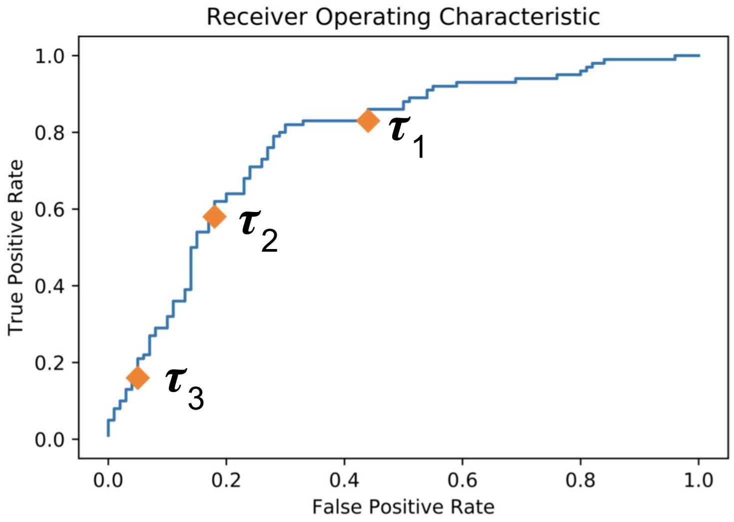
Evaluation: ROC curve (2)

- Imagine that your classifier's output is h(x) and that we use a decision threshold τ (i.e. $\hat{y}_i = +$ if $h(x_i) \geq \tau$)
- Now, place the decision thresholds $\tau_1 < \tau_2 < \tau_3$ on the diamonds: (exam2017 question)



Evaluation: ROC curve (3)

- Correct solution



- Both the TPR and FPR have a constant denominator (#true positive labels and #true negative labels, respectively); if τ decreases sufficiently, more instances will be classified as + and hence, the TP as well as FP will raise

Evaluation: class imbalance case

		True labels	
		Positive	Negative
Predicted labels	Positive	TP	FP
	Negative	FN	TN

- ~~Accuracy: (TP + TN)/n~~
- Recall: TP/(TP+FN)
- Precision: TP/(TP+FP)

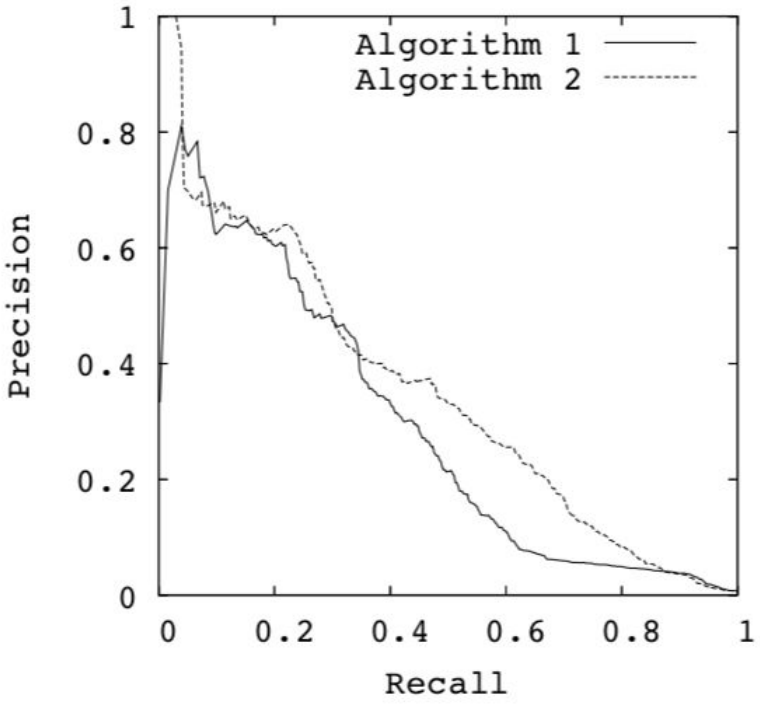
Evaluation: precision-recall trade-off

- Hypothetical situation: we use an algorithm that outputs probabilities (e.g. logistic regression later in the course), with decision threshold τ

- What happens to precision and recall if we raise the decision threshold?
  - Precision will probably, but not necessarily, increase (higher τ leads to lower FP)
  - Recall will decrease or stay the same

- Precision-recall trade-off depends on the situation
  - FN worse, e.g. in disease diagnostics
  - FP worse, e.g. in spam detection (we miss an important email)

Evaluation: precision-recall curve



Davis and Goadrich
(ICML 2006)

Evaluation: precision-recall trade-off (2)

- Hypothetical situation: alg1 has higher precision, alg2 higher recall
- Would averaging precision and recall be a good solution to determine which algorithm works better?
  - No! => Use F-score
- F-score: 2TP/(2TP + FP + FN) $= \frac{2}{precision^{-1} + recall^{-1}} = 2\frac{precision * recall}{precision + recall}$
  - Harmonic mean of precision and recall
  - Equal weights on precision and recall
  - Comment: Hand and Christen (2017). A note on using the F-measure for evaluating record linkage algorithms. *Statistics and Computing*. 10.1007/s11222-017-9746-6.
  - More emphasis on recall (β-times):

F_{β} -score = $(1+\beta^2)\frac{precision * recall}{\beta^2 precision + recall}$

Outline

1. Class imbalance
   a. Problem setting
   b. Simple solutions
   c. Cost-sensitive algorithms
2. Evaluation
   a. Evaluation metrics in standard case
   b. Evaluation metrics in case of imbalanced classes
3. Multi-class SVM
   a. Simple approaches
   b. Multi-class Hinge loss
   c. Evaluation
4. Outlook: cross-entropy loss

Multi-class SVM: simple approaches

- Idea: use binary classification in the multi-class (c) case => reduction
- One-versus-all (OVA)/ all-versus-rest (AVR)
  - c binary classifiers: one class vs. all other classes at a time
  - Predict

$$\hat{y} = argmax_i \; f^{(i)}(\mathbf{x}) = argmax_i \; \mathbf{w}^{(i)T}\mathbf{x}$$

  - (+) simple, fast
  - (-) requires comparable scaling, class imbalance issue, problematic if one class not linearly separable from all other
- One-versus-one (OVO)/ all-versus-all (AVA)
  - c(c-1)/2 binary classifiers
  - (+) simple, doesn’t use confidence
  - (-) slow, ties
- Both used in practice, see e.g. Rifkin and Klautau (2004).In Defense of One-Vs-All Classification.*JMLR* 5: 101-141

Multi-class SVM: Hinge-loss

- Idea: use C weight vectors, one per class and optimize jointly
- Given each data point (**x**, y) we want to achieve that

$$\mathbf{w}^{(y)T}\mathbf{x} \geq max_{i \neq y}\, \mathbf{w}^{(i)T}\mathbf{x} + 1 \qquad (*)$$

- Multi-class Hinge-loss

$$l_{MC-H}(\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(c)}; \mathbf{x}, y) = max(0, 1 + max_{j \in \{1,\ldots,y-1,y+1,\ldots,c\}} \mathbf{w}^{(j)T}\mathbf{x} - \mathbf{w}^{(y)T}\mathbf{x})$$

- $\nabla_{\mathbf{w}^{(i)}}\, l_{MC-H}(\mathbf{w}^{(1:c)}; \mathbf{x}, y)$?

Multi-class classification: tutorial highlights

- During the tutorial we discussed the pros and cons of the reduction approaches as well as possible solutions to the shortcomings

- During the tutorial we worked through an example of image classification (3 labels, 3 training examples) and analyzed what happens with the multi-class Hinge loss

- During the tutorial we discussed in more details the gradient of the multi-class Hinge loss (see lecture slides p.39)

Multi-class SVM: evaluation

- Based on the extended confusion matrix
- Micro- and macro-averaged metrics
- E.g. Extensions of the F-score
  - Macro-averaged F-score

$$\frac{1}{c}\sum_{i=1}^{c} F(i)$$

  - Micro-averaged F-score: F-score on pooled counts from each class contingency table
  - More information: K.Murphy (2012). Machine Learning A Probabilistic Approach. p. 183
- Class imbalance case
  - Weight metric towards largest classes => use micro-averaging
  - Weight metric towards smallest classes => use macro-averaging

Outline

1. Class imbalance
   a. Problem setting
   b. Simple solutions
   c. Cost-sensitive algorithms
2. Evaluation
   a. Evaluation metrics in standard case
   b. Evaluation metrics in case of imbalanced classes
3. Multi-class SVM
   a. Naive approaches
   b. Multi-class Hinge loss
   c. Evaluation
4. Outlook: cross-entropy loss

Outlook: cross-entropy and KL-divergence

Definition 1 (KL-divergence). *Let p and q be a valid probability distributions over the same probability space Ω s.t. $q(x) > 0 \forall x \in \Omega$, then KL-divergence between two distributions is defined as*

$$D_{KL} = \mathbb{E}_p \left[\log \left(\frac{p(x)}{q(x)} \right) \right]. \tag{1}$$

Definition 2 (Cross-entropy). *Let p and q be a valid probability distributions over the same probability space Ω s.t. $q(x) > 0 \forall x \in \Omega$, then cross-entropy is defined as follows,*

$$\mathrm{CE}(p, q) = \mathbb{E}_p[-\log(q(x)))] \tag{2}$$

which is equivalent to $\mathrm{CE}(p, q) = H(p) + D_{KL}(p, q)$, where H is the entropy.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep learning*, vol.1 MIT press Cambridge 2016

Outlook: cross-entropy interpretation

- Information theory view:
  "Cross-entropy is the average number of bits needed to encode data coming from a source with distribution *p* when we use model *q* to define our codebook" [Murphy (2012). Machine Learning A Probabilistic Perspective. p.57-58]

- Probabilistic view:
  minimizing the cross-entropy corresponds to minimizing the negative log-likelihood of the correct class (see the upcoming lectures)

Outlook: cross-entropy loss

- Used for classification algorithms with probability ($\in [0,1]$) as the output
- E.g. a softmax classifier (generalization of logistic regression to multi-class case; more in the upcoming lectures)
- Cross-entropy loss

$$l_{MC-CE}(\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(c)}; \mathbf{x}, y) = -\,log\left(\frac{e^{\mathbf{w}^{(y)T}\mathbf{x}}}{\sum_{j=1}^{c} e^{\mathbf{w}^{(j)T}\mathbf{x}}}\right)$$

- The scores for each class are normalized by applying the softmax function

$$f_j(\mathbf{z}) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

=> the real-valued scores (in **z**) are normalized, i.e. $\in [0,1]$) and sum to 1

Outlook: cross-entropy loss (tutorial highlights)

- During the tutorial we discussed and saw on an example the differences between SVM and softmax classifier with emphasis on the scores
  - SVM outputs uncalibrated, unscaled scores that are possibly difficult to interpret
  - Softmax (due to applying the softmax function to the scores) allows for interpretation as normalized class probabilities
- During the tutorial we looked at the shape of the cross-entropy loss (for binary positive class) and saw that the penalization is not linear and that it changes (decreases) depending on how close to 1 the resulting probability is;
- The SVM “cares” only about the confidence in the true class to be higher by some margin than the confidences in all the other classes; Softmax classifier “cares” about the details, i.e. the true class probability could always be higher;

Questions

- Question regarding the project => Piazza
- Other questions to: joanna.ficek@inf.ethz.ch