Introduction to Machine Learning

Class Imbalance

Nils M. Kriege

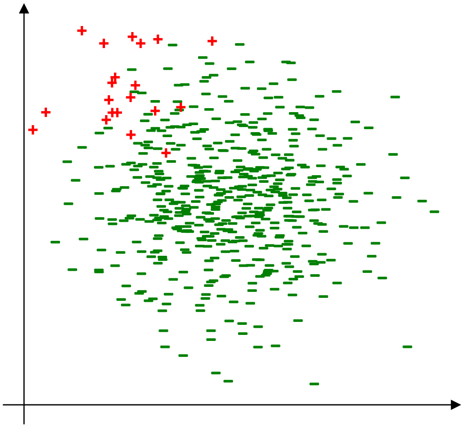
WS 2023

Data Mining and Machine Learning
Faculty of Computer Science
University of Vienna

Credit: Slides based on the IML Lectures by Sebastian Tschiatschek and 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
- ...

Issues with imbalanced data

- ⚠ Accuracy is not a good metric:
  May prefer certain mistakes over others (trade false positives and false negatives)

Issues with imbalanced data

- ⚠ Accuracy is not a good metric:
  May prefer certain mistakes over others (trade false positives and false negatives)
- ⚠ Minority class instances contribute little to the empirical risk
  $\Rightarrow$ may be ignored during optimization!

$$\sum_{i=1}^{n} l_p(\mathbf{x}_i, y_i, \mathbf{w}) = \sum_{\substack{i \in \{1,\dots,n\} \\ y_i = +1}} l_p(\mathbf{x}_i, y_i, \mathbf{w}) + \sum_{\substack{i \in \{1,\dots,n\} \\ y_i = -1}} l_p(\mathbf{x}_i, y_i, \mathbf{w})$$

Issues with imbalanced data

- ⚠ Accuracy is not a good metric:
  May prefer certain mistakes over others (trade false positives and false negatives)
- ⚠ Minority class instances contribute little to the empirical risk
  $\Rightarrow$ may be ignored during optimization!

$$\sum_{i=1}^{n} l_p(\mathbf{x}_i, y_i, \mathbf{w}) = \sum_{\substack{i \in \{1,\ldots,n\} \\ y_i = +1}} l_p(\mathbf{x}_i, y_i, \mathbf{w}) + \sum_{\substack{i \in \{1,\ldots,n\} \\ y_i = -1}} l_p(\mathbf{x}_i, y_i, \mathbf{w})$$

? How can we solve the problem?

Solutions

💡 Subsampling

- Remove training examples from the majority class (e.g., uniformly at random) such that the resulting data set is balanced

Solutions

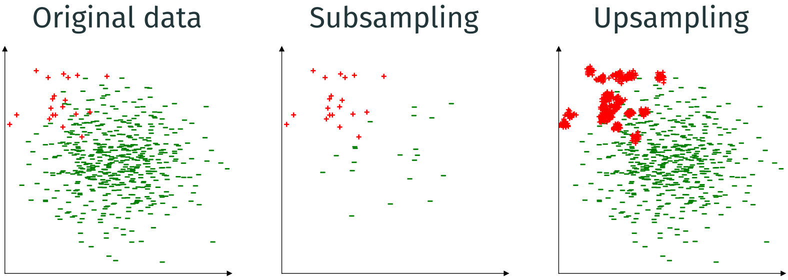
💡 Subsampling

- Remove training examples from the majority class (e.g., uniformly at random) such that the resulting data set is balanced

💡 Upsampling

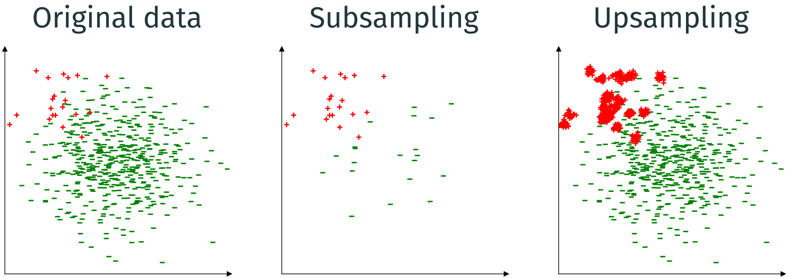
- Repeat data points from minority class (possibly with small random perturbation) to obtain balanced data set

Upsampling / subsampling



[class-imbalance/up-down-sampling.py]

Upsampling / subsampling



What are the pros and cons of these methods?

[class-imbalance/up-down-sampling.py]

Problems with Up-/Subsampling

Method	*Subsampling*	*Upsampling*
Advantages	Smaller data set $\Rightarrow$ faster	Makes use of all data
Disadvantages	Available data is wasted. May lose information about the majority class	Slower (data set up to twice as large) Adding perturbations requires arbitrary choices

Solutions

- Subsampling
- Upsampling
- 💡 Cost-sensitive classification methods

Cost Sensitive Classification

- 💡 Modify Perceptron / SVM to take class balance into account

Cost Sensitive Classification

- 💡 Modify Perceptron / SVM to take class balance into account
- Only difference: Use cost-sensitive loss function

Cost Sensitive Classification

- 💡 Modify Perceptron / SVM to take class balance into account
- Only difference: Use cost-sensitive loss function
- Replace loss by $\ell_{\text{CS}}(\mathbf{w}; \mathbf{x}, y) = c_y \ell(\mathbf{w}; \mathbf{x}, y)$:
  - Perceptron:

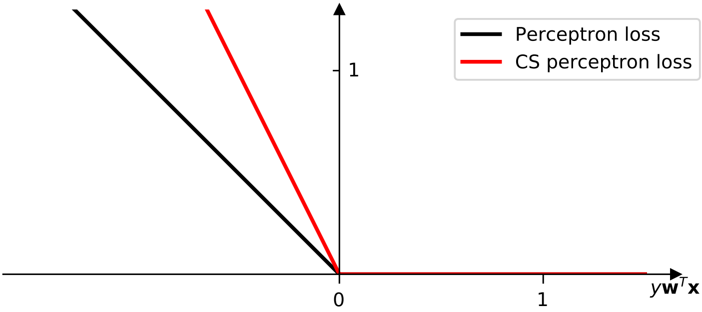
$$\ell_{\text{CS-P}}(\mathbf{w}; \mathbf{x}, y) = c_y \max(0, -y\mathbf{w}^T\mathbf{x})$$

  - SVM:

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

  - with parameters $c_+, c_- > 0$ controlling tradeoff

Cost-sensitive Perceptron loss



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

Example: Cost Sensitive Perceptron

Cost-sensitive Perceptron

- Start at an arbitrary $\mathbf{w}_0 \in \mathbb{R}^d$
- For $t = 1, 2, \ldots$ do
  - Pick data point $(\mathbf{x}', y') \in \mathcal{D}$ from training set uniformly at random (with replacement), and set

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla \ell_{\text{CS-P}}(\mathbf{w}_t; \mathbf{x}', y')$$

Example: Cost Sensitive Perceptron

Cost-sensitive Perceptron

- Start at an arbitrary $\mathbf{w}_0 \in \mathbb{R}^d$
- For $t = 1, 2, \ldots$ do
  - Pick data point $(\mathbf{x}', y') \in \mathcal{D}$ from training set uniformly at random (with replacement), and set

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla \ell_{\text{CS-P}}(\mathbf{w}_t; \mathbf{x}', y')$$

- Only difference: Use cost-sensitive loss function
- For Perceptron:

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

with parameters $c_+, c_- > 0$ controlling tradeoff

Avoiding redundancy

$$\hat{R}(\mathbf{w}; c_+, c_-) = \frac{1}{n} \sum_{\substack{i \in \{1,\dots,n\} \\ y_i = +1}} c_+ l(\mathbf{x}_i, y_i, \mathbf{w}) + \sum_{\substack{i \in \{1,\dots,n\} \\ y_i = -1}} c_- l(\mathbf{x}_i, y_i, \mathbf{w})$$

$$\forall \alpha > 0 \colon \hat{R}(\mathbf{w}; \alpha c_+, \alpha c_-) = \alpha \hat{R}(\mathbf{w}; c_+, c_-)$$

$$\Rightarrow \operatorname*{argmin}_{\mathbf{w}} \hat{R}(\mathbf{w}; \alpha c_+, \alpha c_-) = \operatorname*{argmin}_{\mathbf{w}} \alpha \hat{R}(\mathbf{w}; c_+, c_-) \quad \alpha = 1/c_-$$

$$= \operatorname*{argmin}_{\mathbf{w}} \hat{R}(\mathbf{w}; c_+/c_-, 1)$$

Avoiding redundancy

$$\hat{R}(\mathbf{w}; c_+, c_-) = \frac{1}{n} \sum_{\substack{i \in \{1,\dots,n\} \\ y_i = +1}} c_+ l(\mathbf{x}_i, y_i, \mathbf{w}) + \sum_{\substack{i \in \{1,\dots,n\} \\ y_i = -1}} c_- l(\mathbf{x}_i, y_i, \mathbf{w})$$

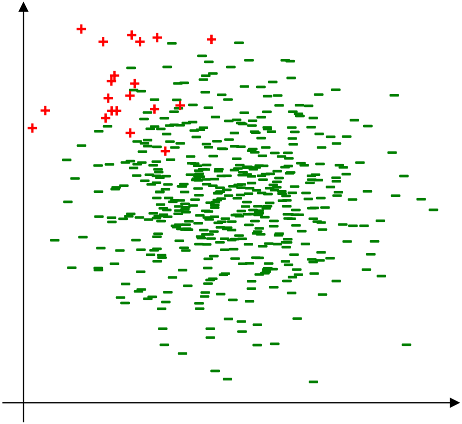
$$\forall \alpha > 0\colon\ \hat{R}(\mathbf{w}; \alpha c_+, \alpha c_-) = \alpha \hat{R}(\mathbf{w}; c_+, c_-)$$

$$\Rightarrow \operatorname*{argmin}_{\mathbf{w}} \hat{R}(\mathbf{w}; \alpha c_+, \alpha c_-) = \operatorname*{argmin}_{\mathbf{w}} \alpha \hat{R}(\mathbf{w}; c_+, c_-) \quad \alpha = 1/c_-$$

$$= \operatorname*{argmin}_{\mathbf{w}} \hat{R}(\mathbf{w}; c_+/c_-, 1)$$

$\Rightarrow$ A single coefficient is sufficient

Evaluating accuracy for imbalanced data



Suppose I claim to have a classifier with 97% accuracy on this data set. Is this good?

Evaluating accuracy for imbalanced data

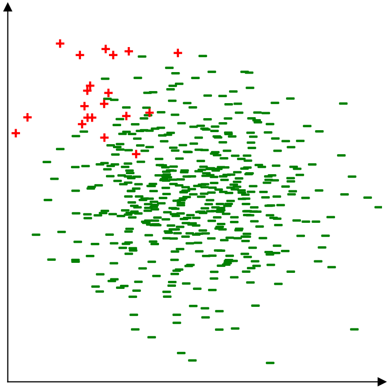
- For imbalanced data, accuracy (i.e., fraction of correct classifications) is often not meaningful
- It makes sense to distinguish (convention: + is rare class):

True label

Predicted label		Positive	Negative	
	Positive	TP	FP	$\sum = p_+$
	Negative	FN	TN	$\sum = p_-$
		$\sum = n_+$	$\sum = n_-$	

$$p_+ + p_- = n_+ + n_- = n$$

Trading false positives and false negatives



Some metrics for imbalanced data

- Accuracy:

$$\frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{n} \in [0, 1]$$

Some metrics for imbalanced data

- Accuracy:

$$\frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{n} \in [0, 1]$$

- Precision:

$$\frac{TP}{TP + FP} = \frac{TP}{p_+} \in [0, 1]$$

Some metrics for imbalanced data

- Accuracy:

$$\frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{n} \in [0, 1]$$

- Precision:

$$\frac{TP}{TP + FP} = \frac{TP}{p_+} \in [0, 1]$$

- Recall:

$$\frac{TP}{TP + FN} = \frac{TP}{n_+} \in [0, 1]$$

Some metrics for imbalanced data

- Accuracy:

$$\frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{n} \in [0, 1]$$

- Precision:

$$\frac{TP}{TP + FP} = \frac{TP}{p_+} \in [0, 1]$$

- Recall:

$$\frac{TP}{TP + FN} = \frac{TP}{n_+} \in [0, 1]$$

- F1 score:

$$\frac{2TP}{2TP + FP + FN} \in [0, 1]$$

How to obtain tradeoff?

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

How to obtain tradeoff?

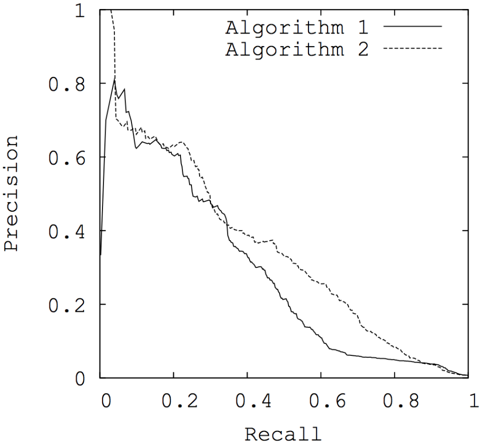
- Option 1:
  - Use cost-sensitive classifier (e.g., cost-sensitive Perceptron), and vary tradeoff parameter
- Option 2:
  - Find a single classifier, and vary classification threshold τ:

$$y = \mathrm{sign}(\mathbf{w}^T\mathbf{x} - \tau)$$

  - Parameter τ shifts decision boundary orthogonal to $\mathbf{w}$.

Precision Recall Curve

[Davis & Goadrich, ICML'06]



More metrics for imbalanced data

- True positive rate (TPR):

$$\frac{TP}{TP + FN}$$

  (=recall)

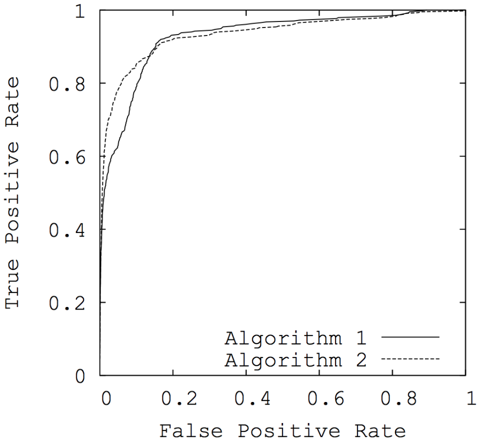
- False positive rate (FPR):

$$\frac{FP}{TN + FP}$$

- Several other metrics used!

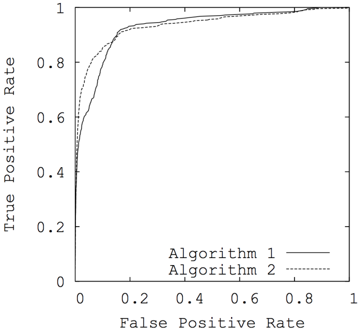
Receiver Operator Characteristic (ROC) Curve

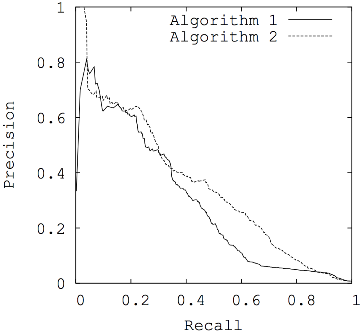
[Davis & Goadrich, ICML'06]



Comparison of the curves

[Davis & Goadrich, ICML'06]

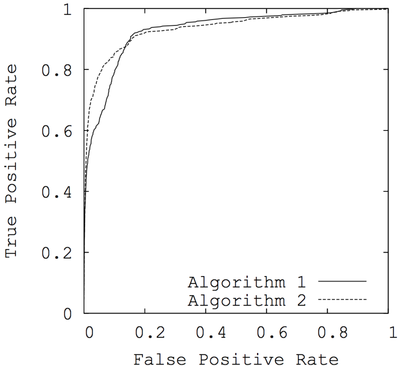




Theorem [Davis & Goadrich '06]: Alg 1 dominates Alg 2 in terms of ROC Curve ⇔ Alg 1 dominates Alg 2 in terms of Precision Recall curves

Area under the Curve

- Often want to compare the ability of classifiers to provide imbalanced classification
- 💡 Compare 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 / SVM
- Evaluating classifiers on imbalanced data sets
  - Metrics (precision, recall, F1 etc.)
  - ROC / Precision Recall curves, AUC

Supervised learning summary so far

Representation/ features	Linear hypotheses, non-linear hypotheses through feature transformations, kernels
Model/ objective	Loss-function (squared loss, ℓ_p loss, 0/1 loss, Perceptron loss, Hinge loss, cost-sensitive loss) + Regularization (ℓ_2 norm, ℓ_1 norm, ℓ_0 penalty)
Method	Exact solution, Gradient Descent, (mini-batch) SGD, Greedy selection
Evaluation metric	Empirical risk = (mean) squared error, Accuracy, F1 score, AUC
Model selection	k-fold cross-validation, Monte Carlo cross-validation

Further Reading / References

- Aly, Mohamed. ”Survey on Multiclass Classification Methods.” (2005)