Introduction to Machine Learning

Multi-class problems

Nils M. Kriege

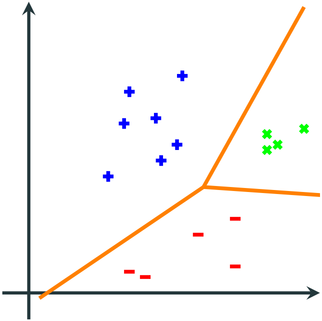
WS 2023

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

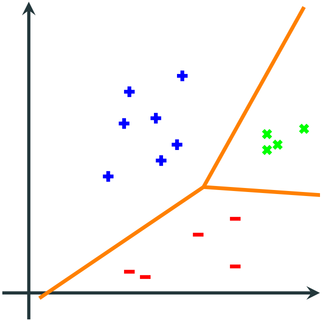
Dealing with multiple classes

- **Given**:
  $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}, y_i \in \mathcal{Y} = \{1, \ldots, c\}, \mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$
- **Want**: $f : \mathcal{X} \to \mathcal{Y}$



Dealing with multiple classes

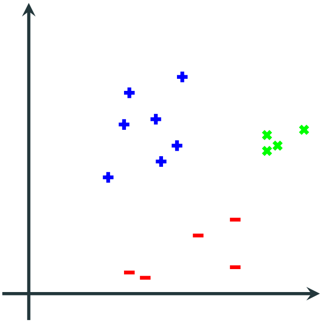
- **Given**:
  $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}, y_i \in \mathcal{Y} = \{1, \ldots, c\}, \mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$
- **Want**: $f : \mathcal{X} \to \mathcal{Y}$



? 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*:

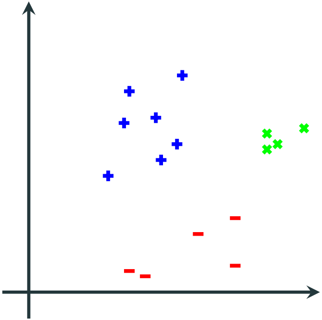


- For each class: $f^{(i)} : \mathbf{X} \to \mathbb{R}$ with $f^{(i)}(\mathbf{x}) = \mathbf{x}^T \mathbf{w}^{(i)}$
- Prediction:

$$\hat{y} = \underset{i}{\operatorname{argmax}} f^{(i)}(\mathbf{x}) = \underset{i}{\operatorname{argmax}} \mathbf{x}^T \mathbf{w}^{(i)}$$

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*:

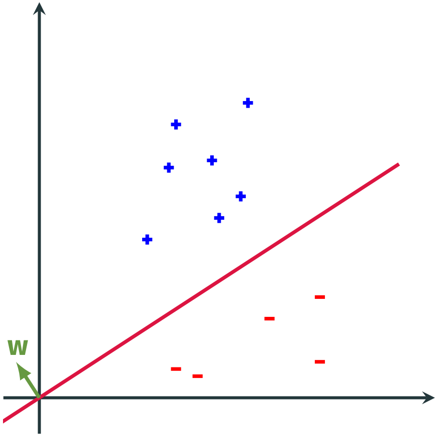


  - For each class: $f^{(i)} : \mathbf{X} \to \mathbb{R}$ with $f^{(i)}(\mathbf{x}) = \mathbf{x}^T \mathbf{w}^{(i)}$
  - Prediction:

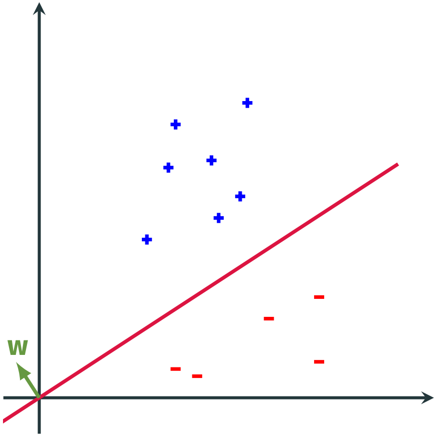
$$\hat{y} = \underset{i}{\operatorname{argmax}} f^{(i)}(\mathbf{x}) = \underset{i}{\operatorname{argmax}} \mathbf{x}^T \mathbf{w}^{(i)}$$

? Is $\mathbf{w}^T \mathbf{x}$ a good measure of confidence?

Confidence in classification

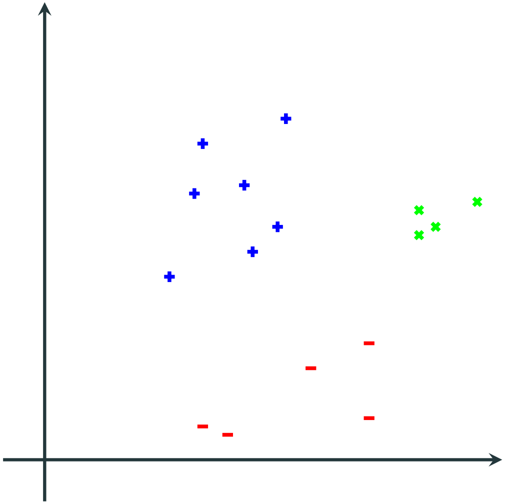


Confidence in classification



- For a fixed $\mathbf{w}$, $\mathbf{x}^T\mathbf{w}$ is a valid measure
- But for $\alpha > 0$: $\text{sign}(\mathbf{w}^T\mathbf{x}) = \text{sign}\left((\alpha\mathbf{w})^T\mathbf{x}\right)$
- Scale of $\mathbf{w}$ influences confidence, but not decision boundary
- Solutions:
    1. Normalize $\mathbf{w}$
    2. Use regularization

Confidence in classification



Confidence in classification

- Only works well if classifiers produce confidence scores on the "same scale"

Confidence in classification

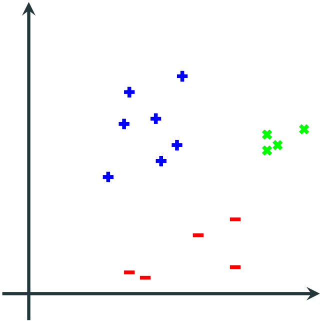
- Only works well if classifiers produce confidence scores on the "same scale"
- Individual binary classifiers see imbalanced data, even if the whole data set is balanced

Confidence in classification

- 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

Method	*One-vs-all*	*One-vs-one*
Advantages	Only c classifiers needed (faster!)	No confidence needed
Disadvantages	Requires confidence in prediction / leads to class imbalance	Slower (need to train $c(c-1)/2$ models)

Alternative methods

- Other encodings
  - E.g., error correcting output codes
- Explicit multi-class models
  - E.g., multi-class Perceptron / SVM etc.
  - Some models are naturally multi-class (e.g., nearest neighbor, generative probabilistic models, see later)
- Often one-vs-all / one-vs-one works very well

How many binary classifiers do we need?

- One-vs-all: c
- One-vs-one: $c(c-1)/2$

- Can we get away with less?

How many binary classifiers do we need?

- One-vs-all: c
- One-vs-one: $c(c-1)/2$

- Can we get away with less?
- **Yes!** Using binary encoding we need $\lceil \log_2 c \rceil$ classifiers only.

Multi-class vs. coding

- 💡 **Key idea**: Can in principle view multi-classification as "decoding" the class label
  - Each classifier predicts one bit
- Might be able to get away using $O(\log_2 c)$ classifiers!
- Can use ideas from coding theory to do multi-class classification

Example: Error correcting output codes

Idea: Use few additional bits to detect (and correct) errors!

Class label	Binary encoding			
1	-1	-1	-1	-1
2	+1	+1	-1	-1
3	-1	-1	+1	+1
4	+1	-1	+1	-1
5	+1	+1	+11	+1

Each encoding contain at least two times -1 or $+1$. Single "bit flips" can be detected.

Example: Error correcting output codes

Idea: Use few additional bits to detect (and correct) errors!

Class label	Binary encoding					
1	−1	−1	−1	−1	−1	−1
2	+1	+1	+1	−1	−1	−1
3	−1	−1	−1	+1	+1	+1
4	+1	−1	+1	−1	+1	−1
5	+1	+1	+1	+1	+1	+1

Predict class label that is closest to the predicted encoding in terms of the Hamming distance, e.g.,

$$(-1, -1, -1, -1, -1, +1) \mapsto (-1, -1, -1, -1, -1, -1)$$

Multi-class SVMs

- 💡 **Key idea**: Maintain c weight vectors, one for each class

$$\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(c)}$$

Predict: $\hat{y} = \text{argmax}_{i \in \{1,\ldots,c\}} \mathbf{w}^{(i)T}\mathbf{x}$

- Given each data point (x, y), want to achieve that:

$$\underbrace{\mathbf{w}^{(y)T}\mathbf{x}}_{\text{conf. correct class}} \geq \underbrace{\mathbf{w}^{(i)T}\mathbf{x}}_{\text{conf. other class}} + 1 \quad \forall i \in \{1, \ldots, c\} \setminus \{y\}$$

$$\iff \quad \mathbf{w}^{(y)T}\mathbf{x} \geq \max_{i \in \{1,\ldots,c\} \setminus \{y\}} \mathbf{w}^{(i)T}\mathbf{x} + 1 \qquad (*)$$

Multi-class Hinge Loss

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

The multi-class Hinge loss $\ell_{\text{MC-H}}(\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(c)})$ is zero iff (*) is satisfied, i.e., the difference in confidence between the correct classifier and all other classifiers is at least 1.

Multi-class Hinge Loss

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

The multi-class Hinge loss $\ell_{\text{MC-H}}(\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(c)})$ is zero iff (*) is satisfied, i.e., the difference in confidence between the correct classifier and all other classifiers is at least 1.

Gradient:

$$\nabla_{\mathbf{w}^{(i)}} \ell_{\text{MC-H}}(\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(c)}; \mathbf{x}, y) = \begin{cases} 0 & \text{if (*) is staisfied or} \\ & i \neq y \wedge i \notin \text{argmax}_j \, \mathbf{w}^{(j)T}\mathbf{x}, \\ -\mathbf{x} & \text{if not (*) and } i = y, \\ \mathbf{x} & \text{otherwise.} \end{cases}$$

Note: Confusion matrices

- When evaluating multi-class classifiers, one often considers confusion matrices:

True label

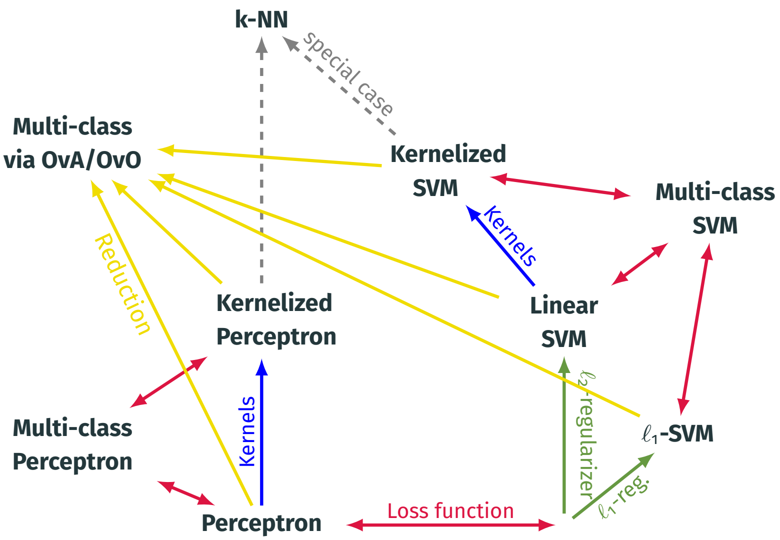
Predicted label

	Cat	Dog	Elephant
Cat	5	2	0
Dog	3	7	0
Elephant	1	0	6

What you need to know

- Using binary classification for multi-class problems (One-vs-all, one-vs-one)
- Multi-class SVM
- Benefits of the respective methods

Multi-classification big picture



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, multi-class hinge loss) + Regularization (ℓ_2 norm, ℓ_1 norm, ℓ_0 penalty)
Method	Exact solution, Gradient Descent, (mini-batch) SGD, Greedy selection, reductions
Evaluation metric	Empirical risk = (mean) squared error, Accuracy, F1 score, AUC, confusion matrices
Model selection	k-fold cross-validation, Monte Carlo cross-validation

Further Reading / References

- S. Shalev-Schwartz & S. Ben-David, "Understanding Machine Learning: From Theory to Algorithms", Chapter 17 (in particular 17.1 and 17.2)