ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

inf | Informatik
Computer Science

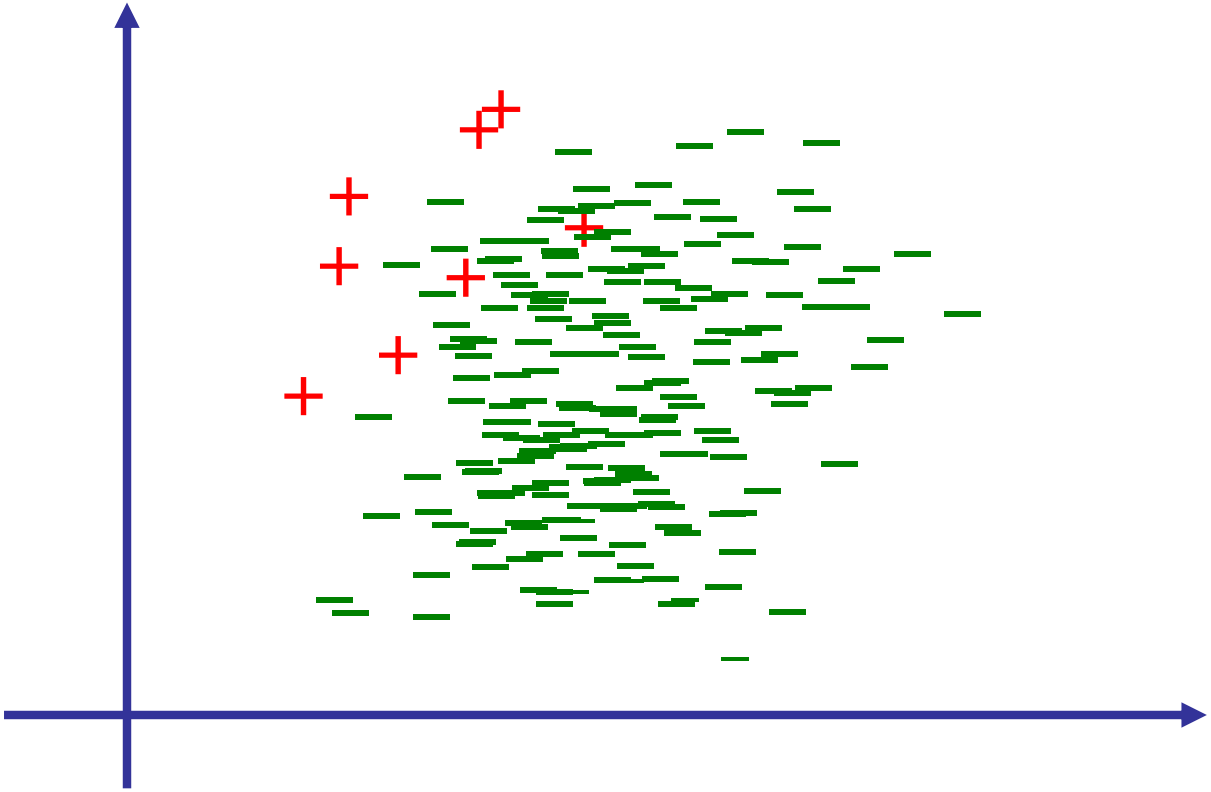
Introduction to Machine Learning

Class-imbalance

Prof. Andreas Krause
Learning and Adaptive Systems (las.ethz.ch)

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

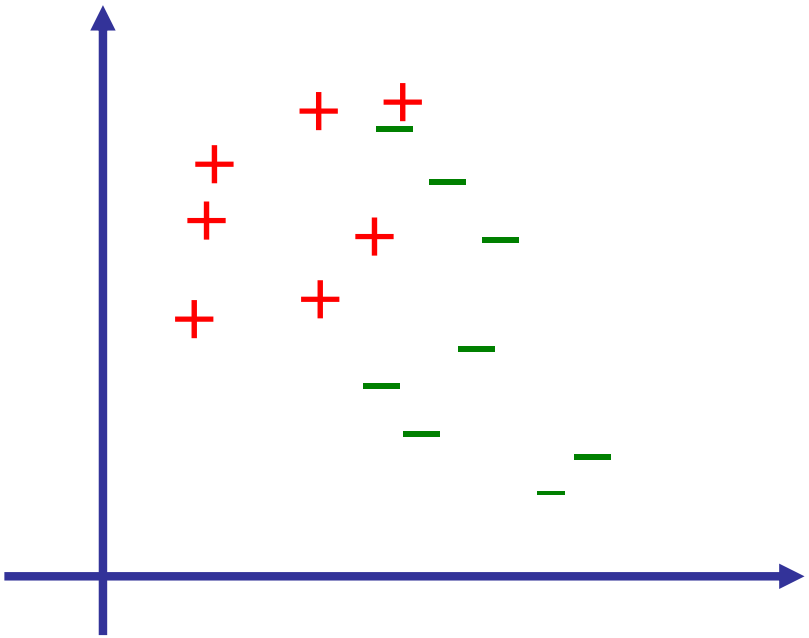
- Accuracy is not a good metric: May prefer certain mistakes over others
  (trade false positives and false negatives)

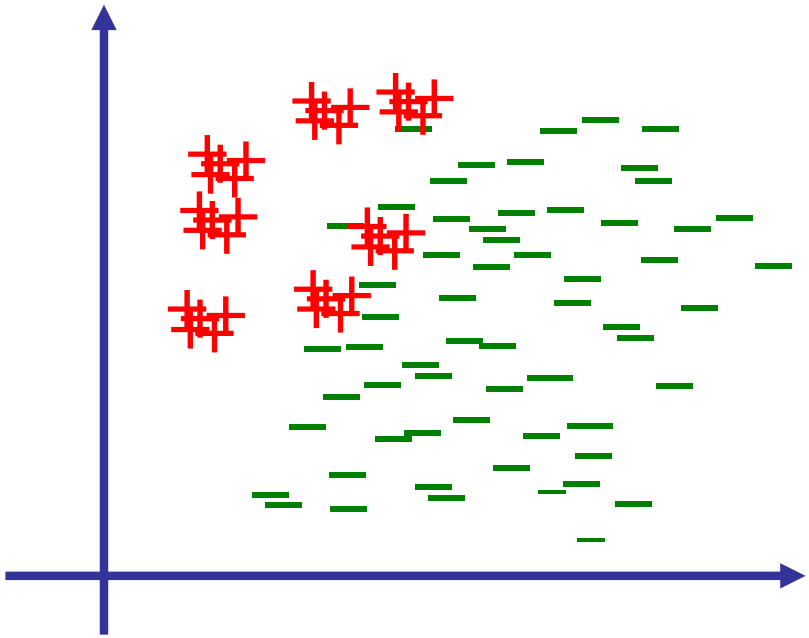
- Minority class instances contribute little to the empirical risk → may be ignored during optimization!

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 / downsampling





Problems with Up-/Downsampling

Method	*Downsampling*	*Upsampling*
Advantages	Smaller data set ➔ 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

- Use cost-sensitive classification methods

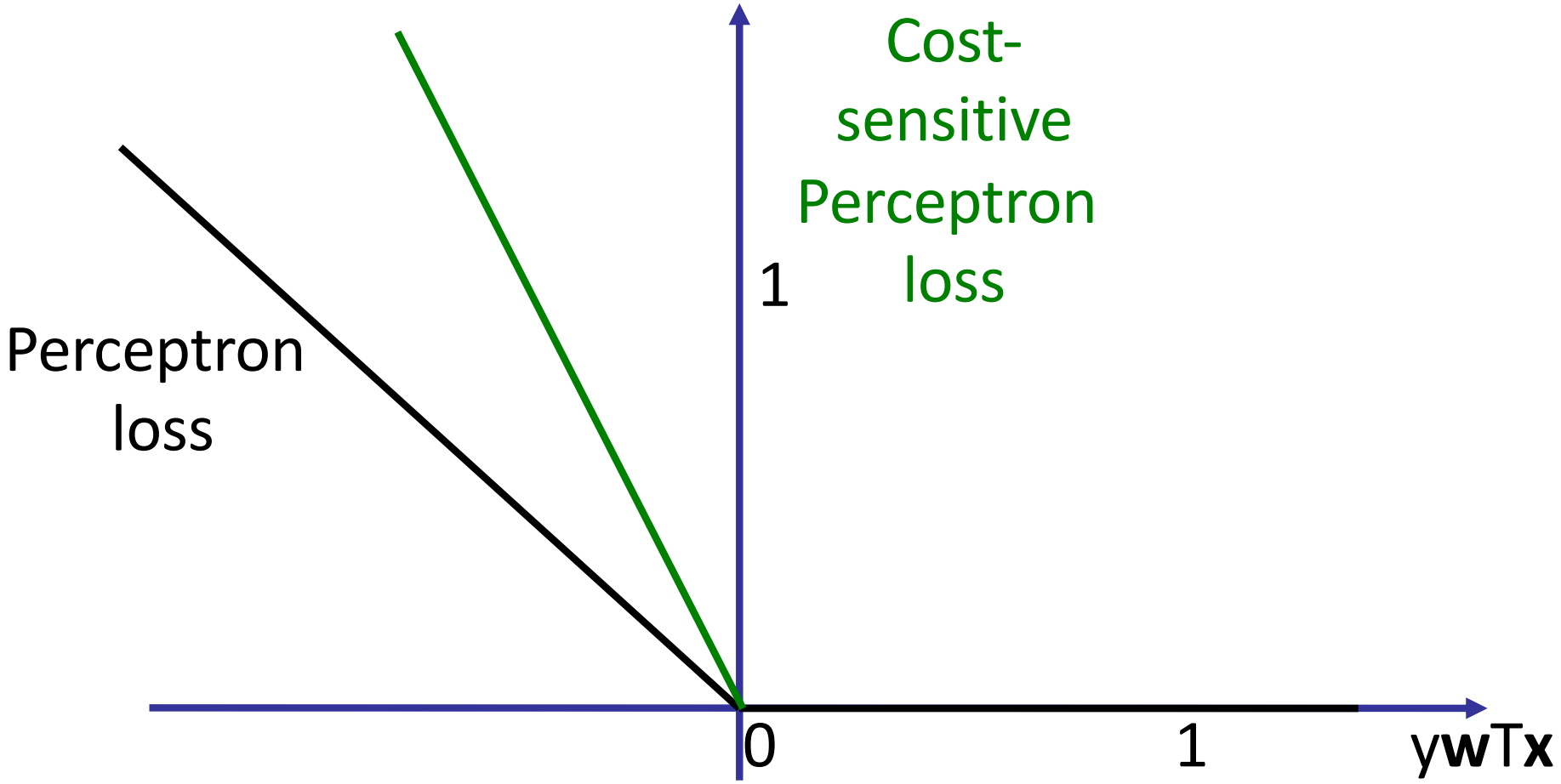
Cost Sensitive Classification

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

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

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

Cost-sensitive Perceptron loss



$$\ell(\mathbf{w}; \mathbf{x}, y) = c_y \max(0, -y\mathbf{w}^T\mathbf{x})$$

Example: Cost Sensitive Perceptron

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

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

- Only difference: Use cost-sensitive loss function.
- For Perceptron: $\ell(\mathbf{w}; \mathbf{x}, y) = c_y \max(0, -y\mathbf{w}^T\mathbf{x})$

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

Avoiding redundancy

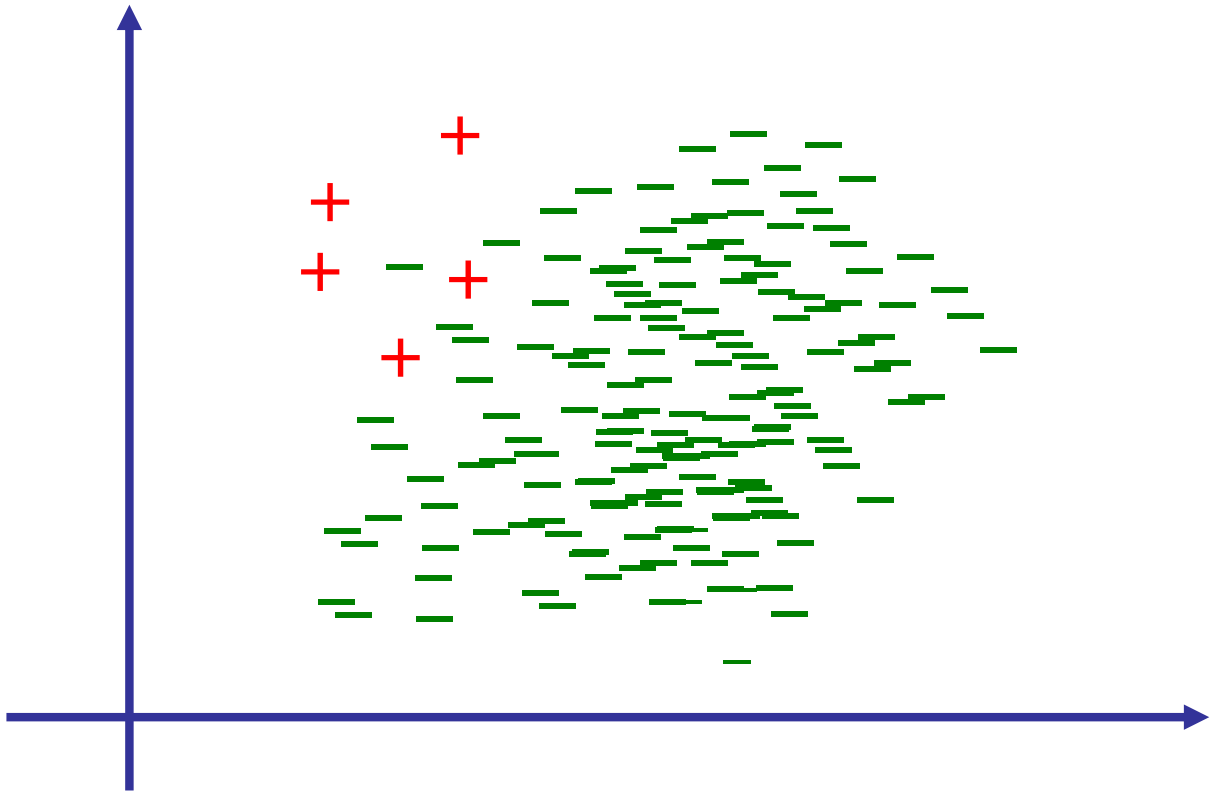
$$\hat{R}(w; c_+, c_-) = \frac{1}{n} \sum_{i: y_i = +1} c_+ \, \ell(w; x_i, y_i) + \frac{1}{n} \sum_{i: y_i = -1} c_- \, \ell(w; x_i, y_i)$$

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

$$\Rightarrow \quad \operatorname*{argmin}_w \; - \; = \operatorname*{argmin}_w \; - \; = \operatorname*{argmin}_w \hat{R}\left(w; \frac{c_+}{c_-}, 1\right)$$

$$\Rightarrow \quad \text{w.l.o.g.:} \quad \alpha := \frac{1}{c_-}$$

Evaluating accuracy for imbalanced data



Suppose I claim to have a classifier with 99% 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	
		Positive	Negative
Predicted label	Positive	TP	FP
	Negative	FN	TN

$\Sigma = p_+$

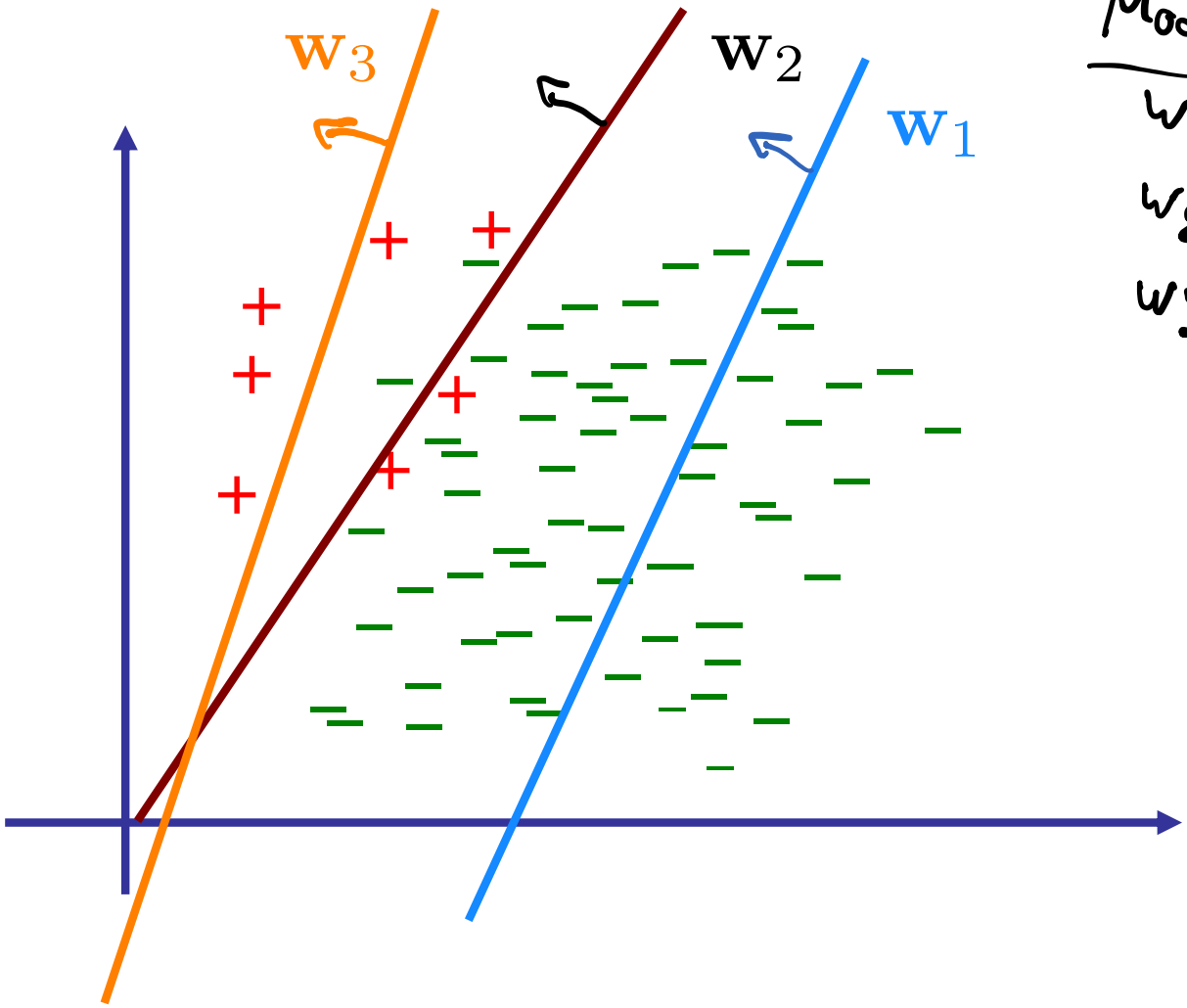
$\Sigma = p_-$

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

$\Sigma = n_+$

$\Sigma = n_-$

Trading false positives and false negatives



Model	FP	FN
w_1	high	0
w_2	low	low
w_3	0	high

Some metrics for imbalanced data

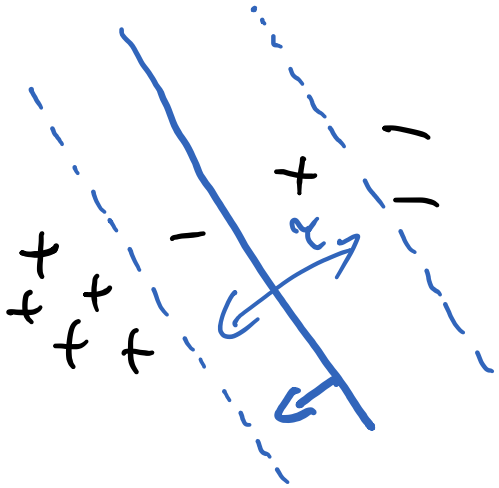
- Accuracy: $\frac{TP+TN}{TP+TN+FP+FN} = \frac{TP+TN}{n}$
- Precision: $\frac{TP}{TP+FP} = \frac{TP}{p_+} \in [0,1]$
- Recall: $\frac{TP}{TP+FN} = \frac{TP}{n_+} \in [0,1]$
- F1 score: $\frac{2TP}{2TP+FP+FN} = \frac{2}{\frac{1}{prec}+\frac{1}{rec}} \in [0,1]$

How to obtain tradeoff?

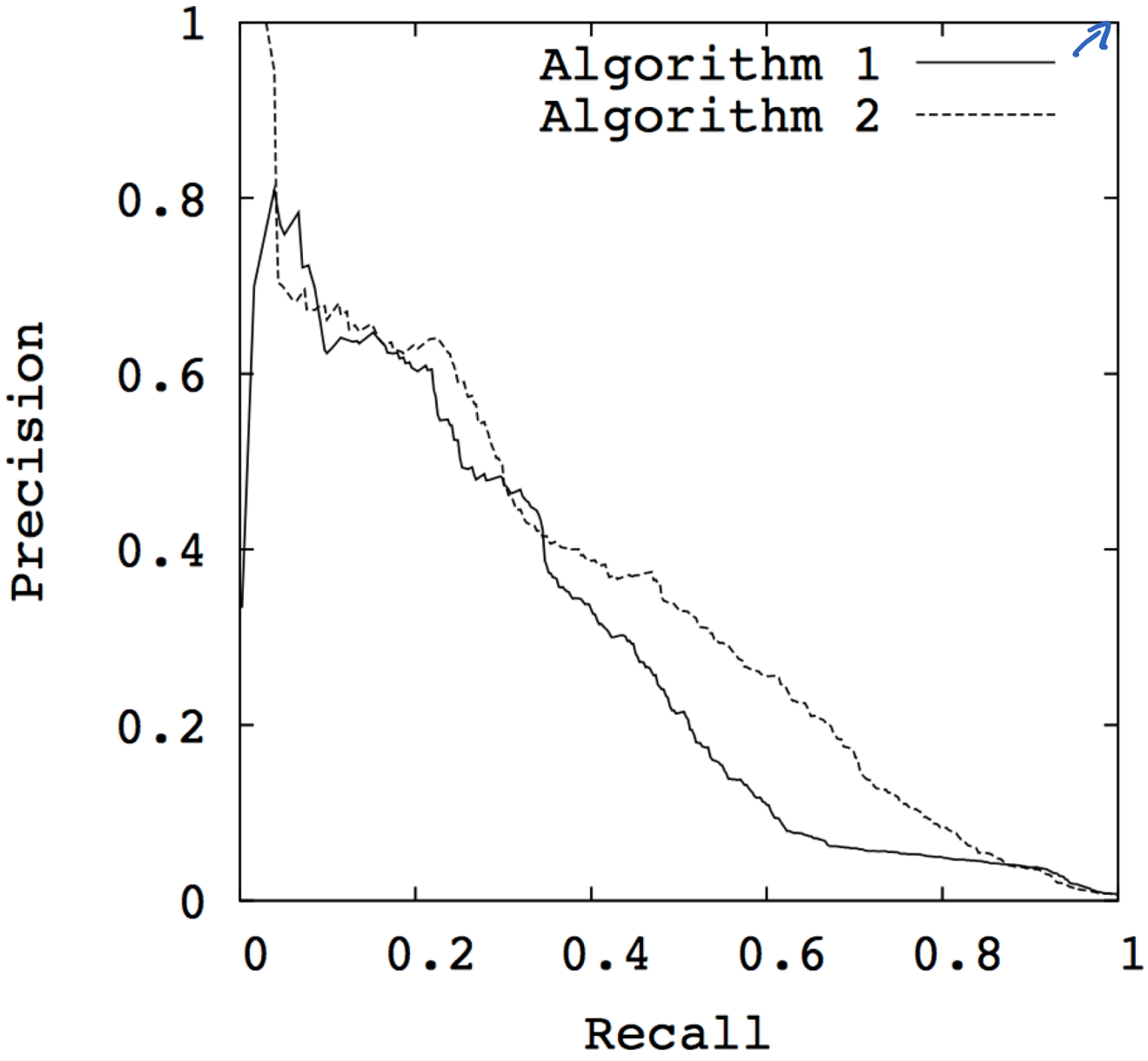
- Option 1:
  - Use cost-sensitive classifier (e.g., cost-sensitive Perceptron), and vary tradeoff parameter

- Option 2:
  - Find a single classifier, and vary classification threshold τ

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



Precision Recall Curve



[Davis & Goadrich, ICML'06]

More metrics for imbalanced data

- True positive rate (TPR)
  = recall!

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

- False positive rate (FPR)

$$\frac{FP}{TN+FP} = \frac{FP}{n_-}$$

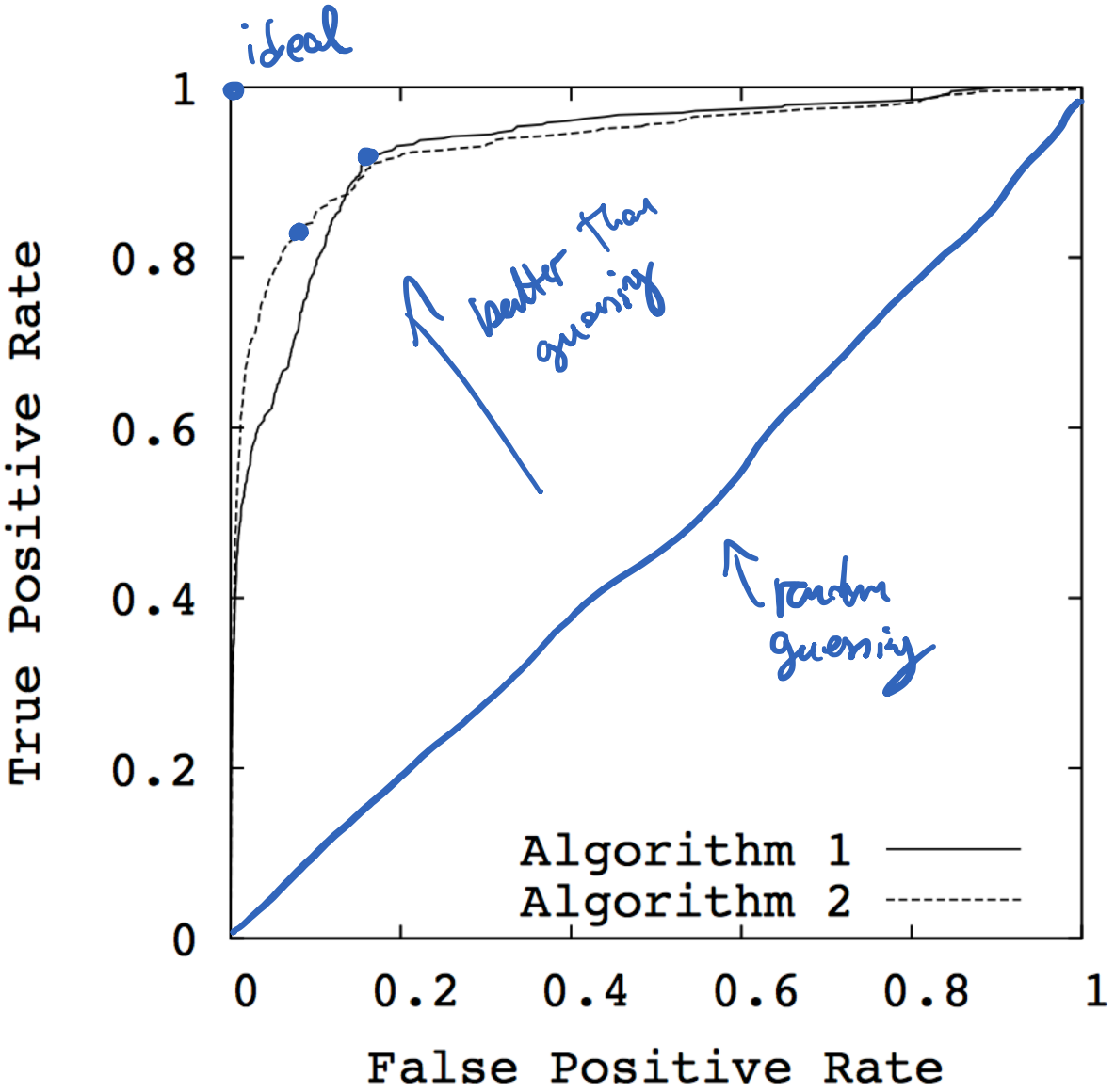
- Several other metrics used!

Pos predict + with prob. p

$$\Rightarrow \quad \mathbb{E}[TPR] = \frac{\mathbb{E}[TP]}{n_+} = \frac{p \cdot n_+}{n_+} = p$$

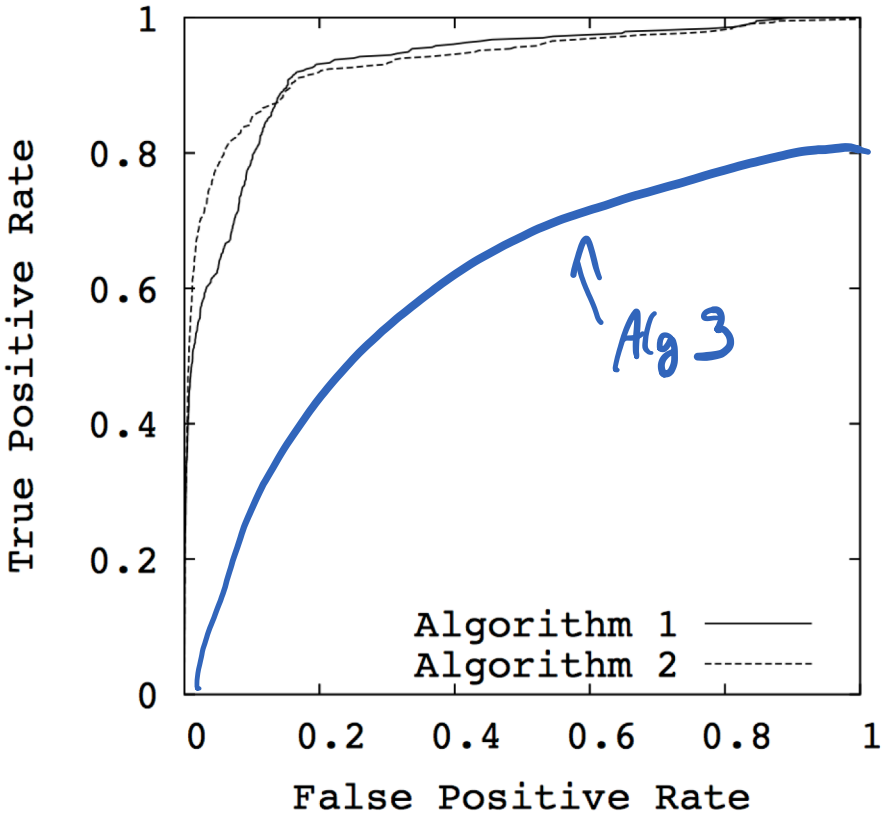
$$\mathbb{E}[FPR] = \frac{\mathbb{E}[FP]}{n_-} = \frac{p \cdot n_-}{n_-} = p$$

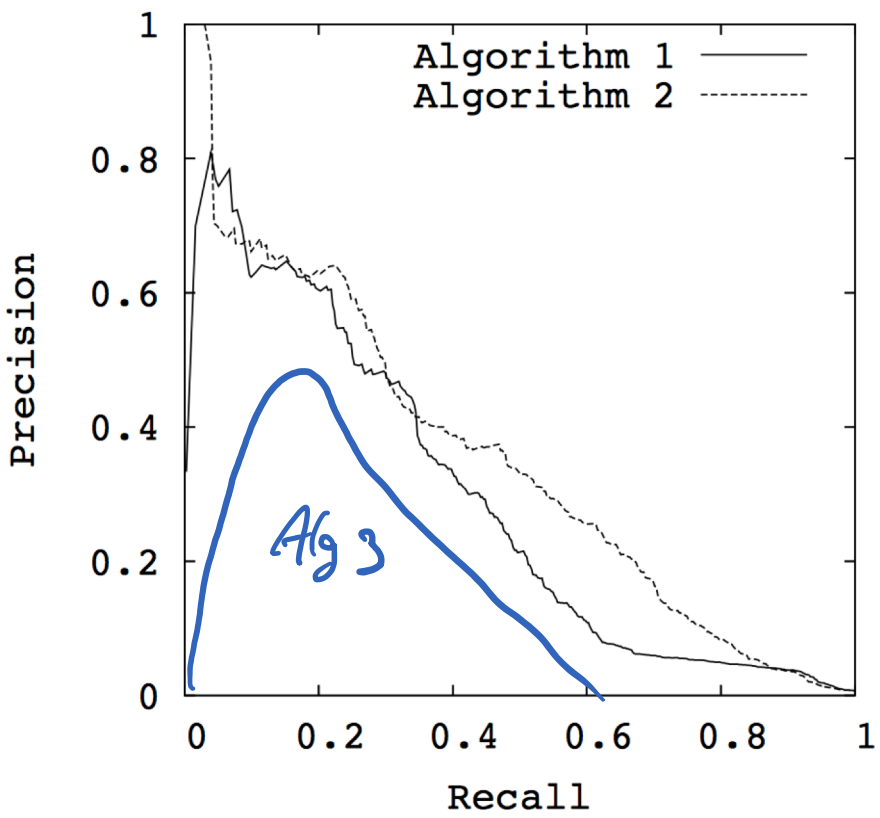
Receiver Operator Characteristic (ROC) Curve



[Davis & Goadrich, ICML'06]

Comparison of the curves



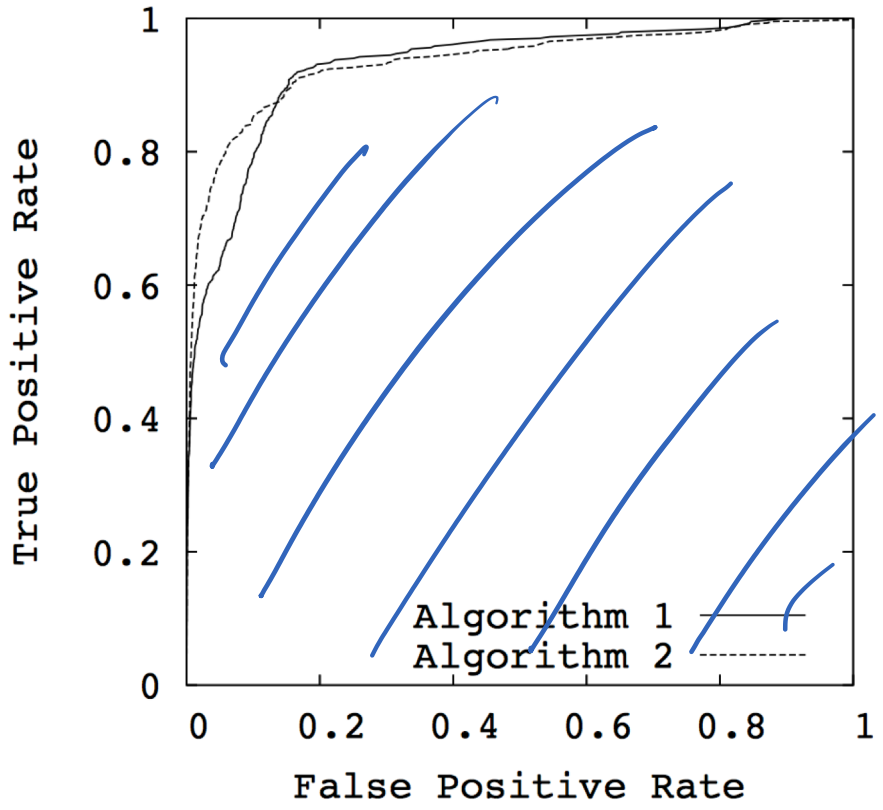


Theorem [Davis & Goadrich '06]: Alg 1 dominates Alg 2 in terms of ROC Curve ⇔ Alg 1 dominates Alg 2 in terms of Precision Recall curves

Area under the Curve

- Often want to compare the ability of classifiers to provide imbalanced classification
- Can compute Area under the ROC or Precision Recall curves

AUC



Random: AUC = $\frac{1}{2}$

Ideal: AUC = 1

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; nonlinear hypotheses with nonlinear feature transforms, kernels		
Model/ objective:	Loss-function Squared loss, 0/1 loss, Perceptron loss, Hinge loss, cost sensitive losses	+	Regularization L^2 norm, L^1 norm
Method:	Exact solution, Gradient Descent, (mini-batch) SGD, Convex Programming, …		
Evaluation metric:	Mean squared error, Accuracy, F1 score, AUC, …		
Model selection:	K-fold Cross-Validation, Monte Carlo CV		

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

inf | Informatik
Computer Science

Introduction to Machine Learning

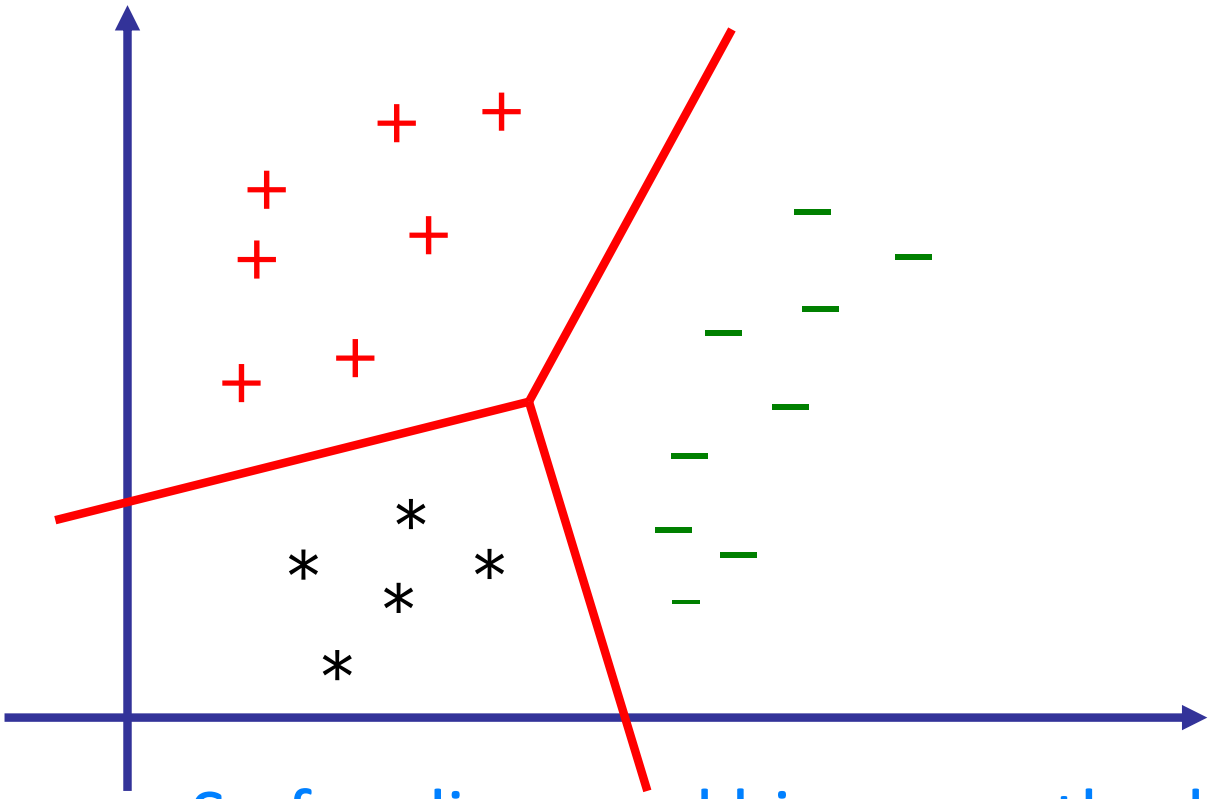
Multi-class problems

Prof. Andreas Krause
Learning and Adaptive Systems (las.ethz.ch)

Dealing with multiple classes

Given: $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ $\quad y_i \in \mathcal{Y} = \{1, \ldots, c\}$

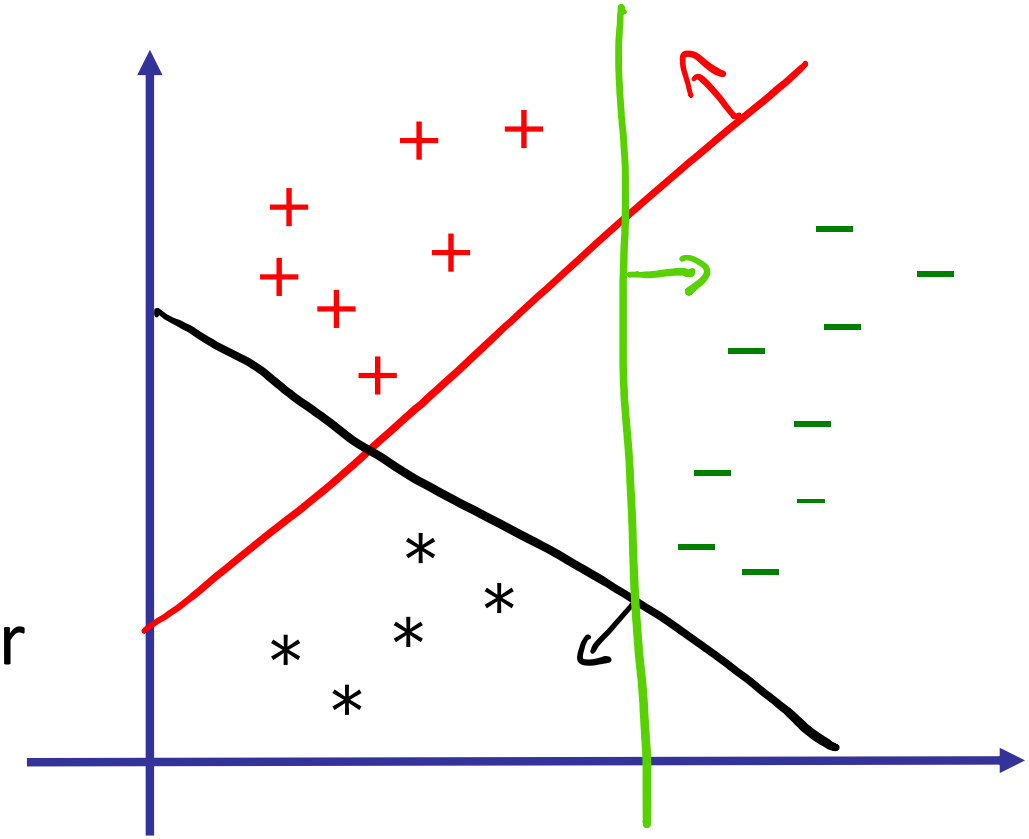
Want: $f : \mathcal{X} \to \mathcal{Y}$ $\quad \mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$



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

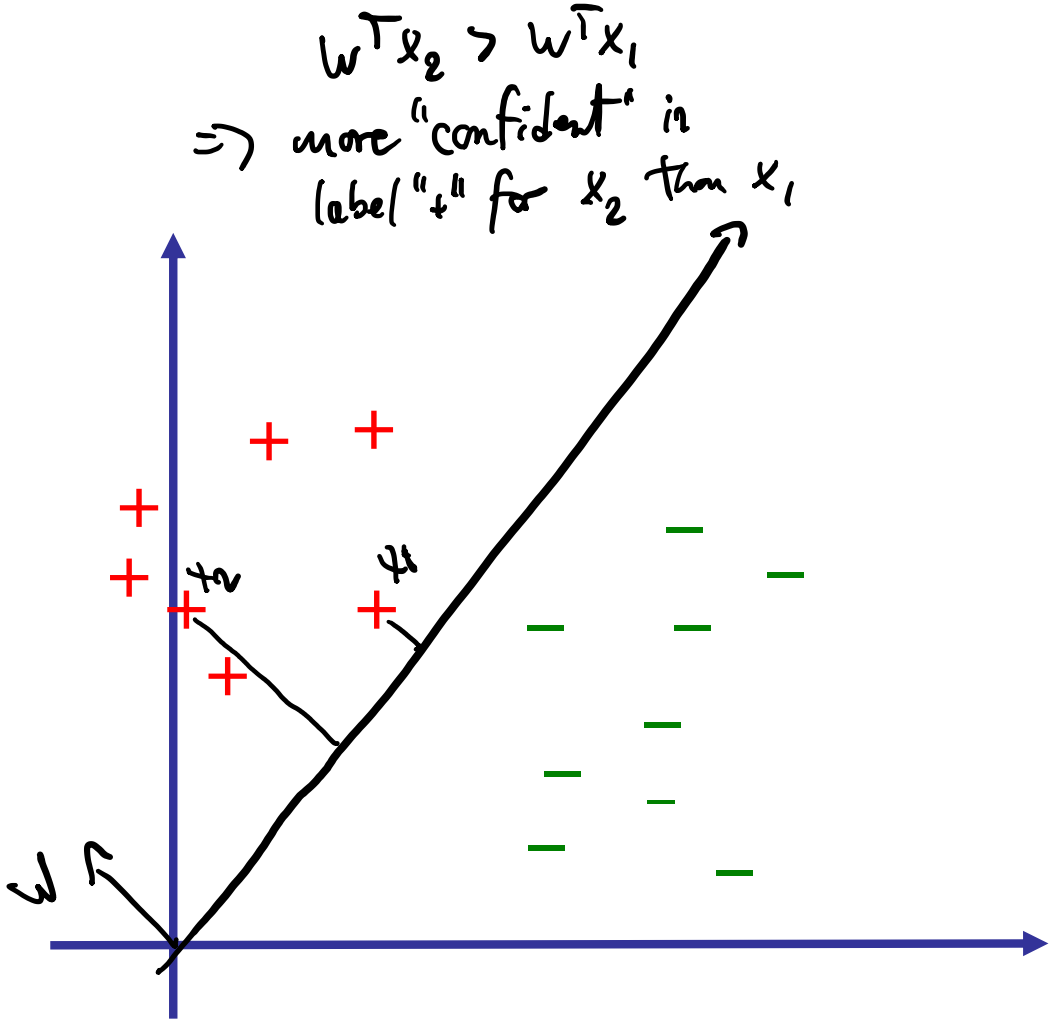


For each class, fit $f^{(i)}: X \longmapsto \mathbb{R}$

$f^{(i)}(x) = w^{(i)T} x$

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

Confidence in classification



Note: for $\alpha > 0$,

$$\text{sign}((\alpha w)^T x) = \text{sign}(w^T x)$$

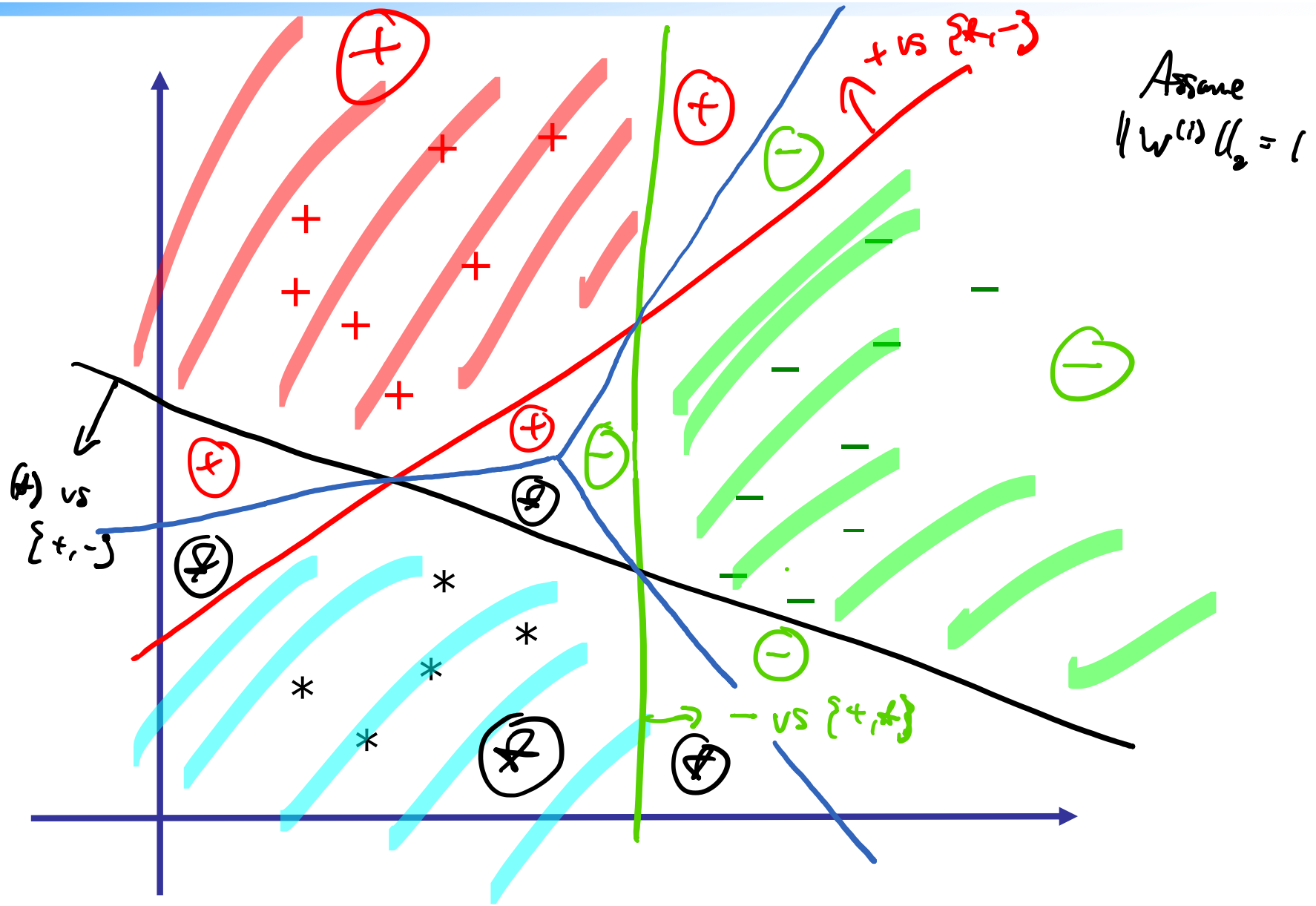
Thus αw and w implement same decision boundary, but different "confidence"

Solution: 1) $w \leftarrow \frac{w}{\|w\|_2}$

(renormalize to unit length)

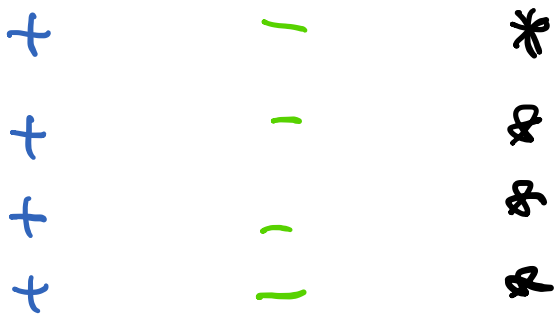
2) In practice, when using regularization, magnitude of $\|w\|_2$ is kept under control

One-vs-all (OvA) decision boundary



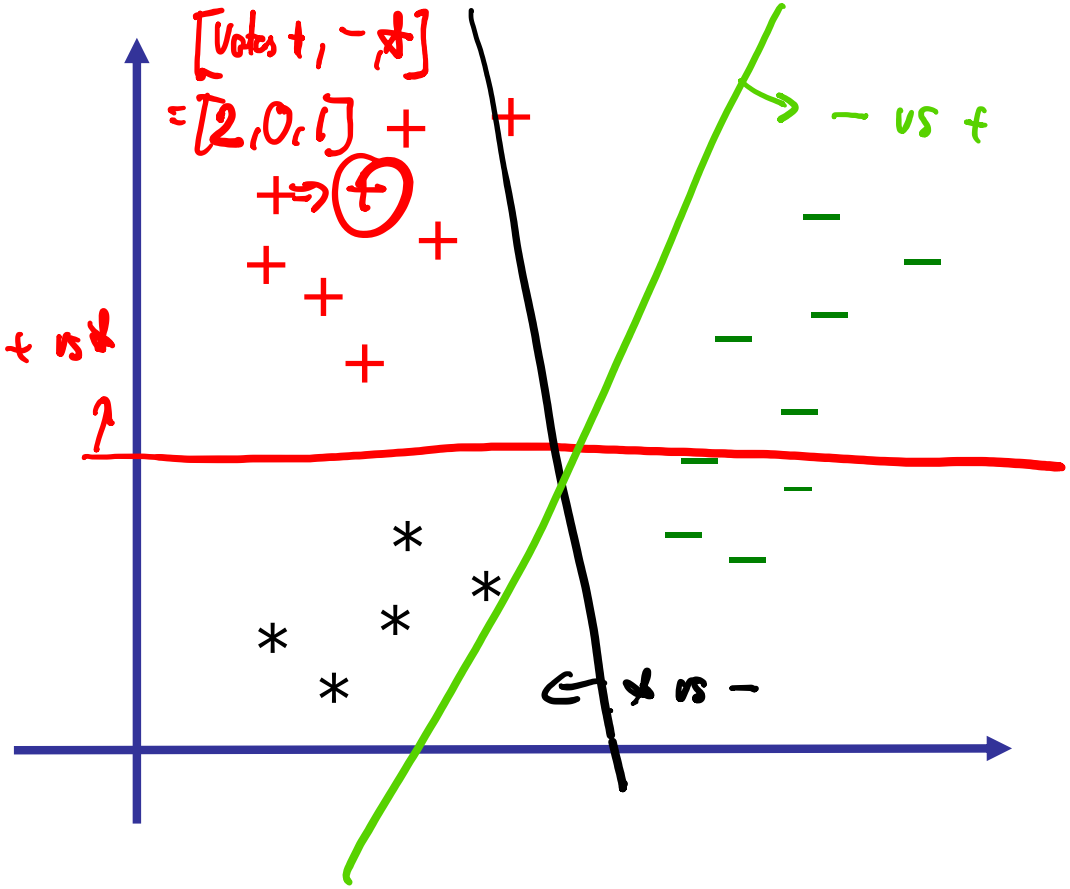
Challenges with one-vs-all

- 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)

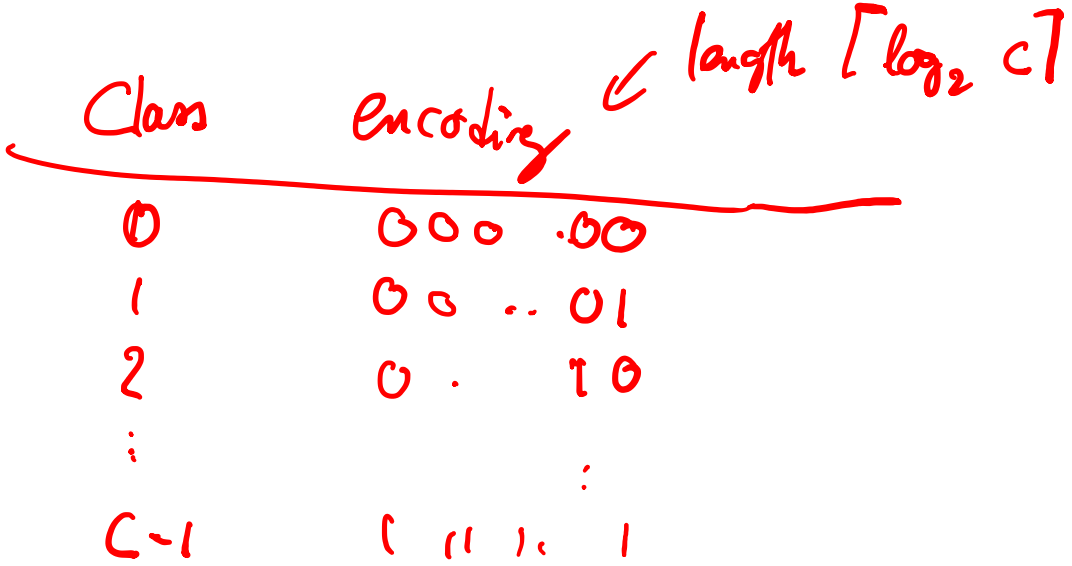
Alternative methods

- 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*?



Multi-class vs. coding

- 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 c)* classifiers!

- Can use ideas from coding theory to do multi-class classification

Example: Error correcting output codes

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 +1 +1

Example: Error correcting output codes

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

Multi-class SVMs

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

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

Predict $\hat{y} \in \underset{i \in \{1..c\}}{\operatorname{argmax}} \; w^{(i)T} x$

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

$w^{(y)T} x \geq w^{(i)T} x + 1 \quad \forall i \in \{1..c\} \setminus \{y\}$

Confidence in correct label

Confidence in any other label

$\equiv \quad w^{(y)T} x \geq \max_{i \in \{1..c\} \setminus \{y\}} w^{(i)T} x + 1 \qquad (*)$

Multi-class Hinge Loss

$$\ell_{MC-H}(\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(c)}; \mathbf{x}, y) =$$

$$\max\left(0, 1 + \max_{j \in \{1,\ldots,y-1,y+1,\ldots,c\}} \mathbf{w}^{(j)T}\mathbf{x} - \mathbf{w}^{(y)T}\mathbf{x}\right)$$

$= 0$ iff. (*) is satisfied

$$\nabla_{w^{(i)}} \ell_{MC-H}(w^{(1:c)}; x, y) = \begin{cases} 0 & \text{if } (*) \text{ is sat. } \underline{\text{or}} \ i \neq y \text{ and } i \neq \underset{j}{\operatorname{argmax}}\, w^{(j)T}x \\ -x & \text{if } \neg(*) \text{ and } i = y \\ +x & \text{otw} \end{cases}$$

Note: Confusion matrices

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

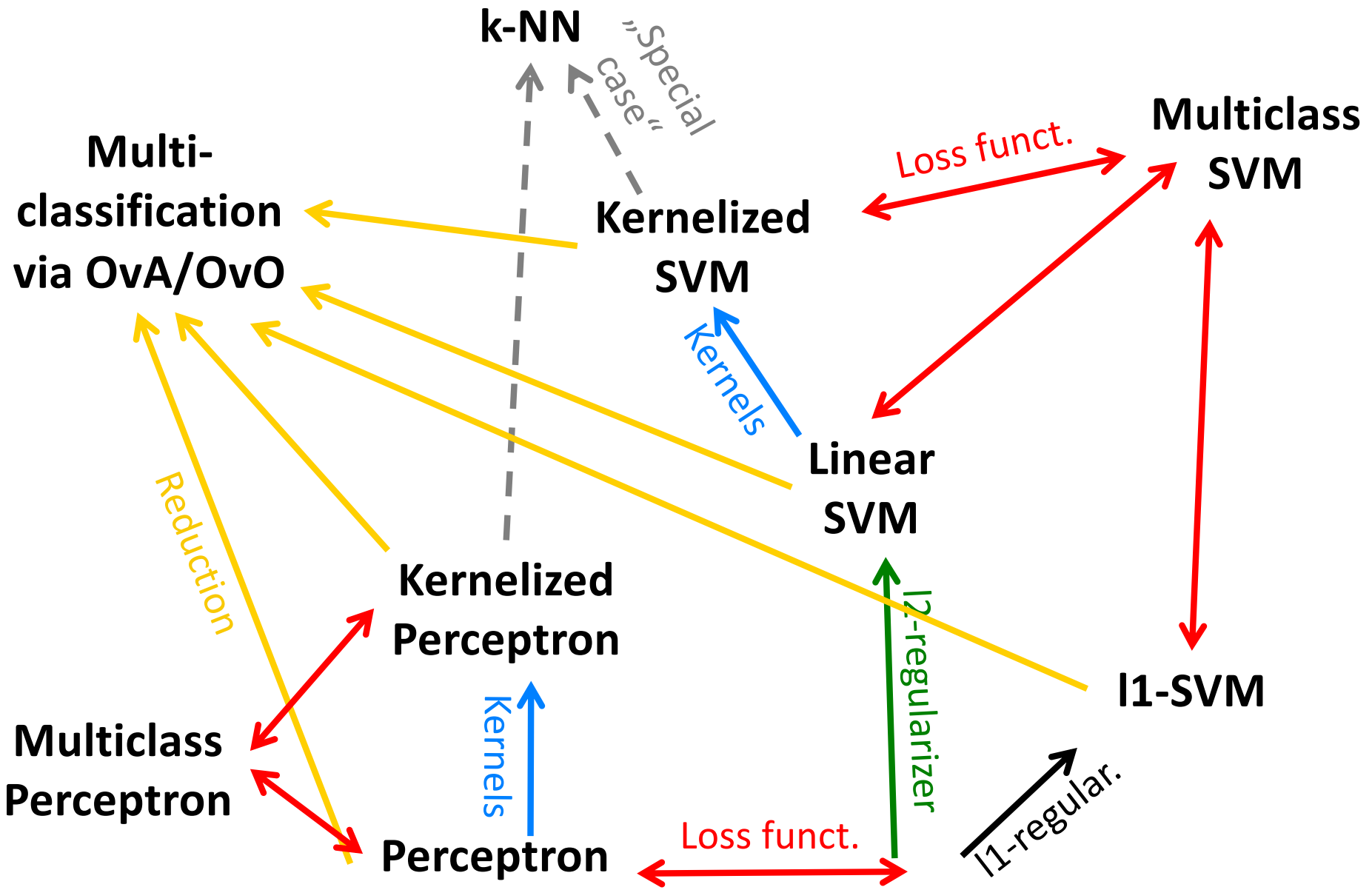
True label

Predicted label	Cat	Dog	Elefant
Cat	5	2	0
Dog	3	7	0
Elefant	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; nonlinear hypotheses with nonlinear feature transforms, kernels		
Model/ objective:	Loss-function Squared loss, 0/1 loss, Perceptron loss, Hinge loss, cost sensitive losses, multi-class hinge loss	+	Regularization L^2 norm, L^1 norm
Method:	Exact solution, Gradient Descent, (mini-batch) SGD, Reductions		
Evaluation metric:	Mean squared error, Accuracy, F1 score, AUC, Confusion matrices		
Model selection:	K-fold Cross-Validation, Monte Carlo CV		