ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

inf | Informatik Computer Science

Introduction to Machine Learning

Linear Classification

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

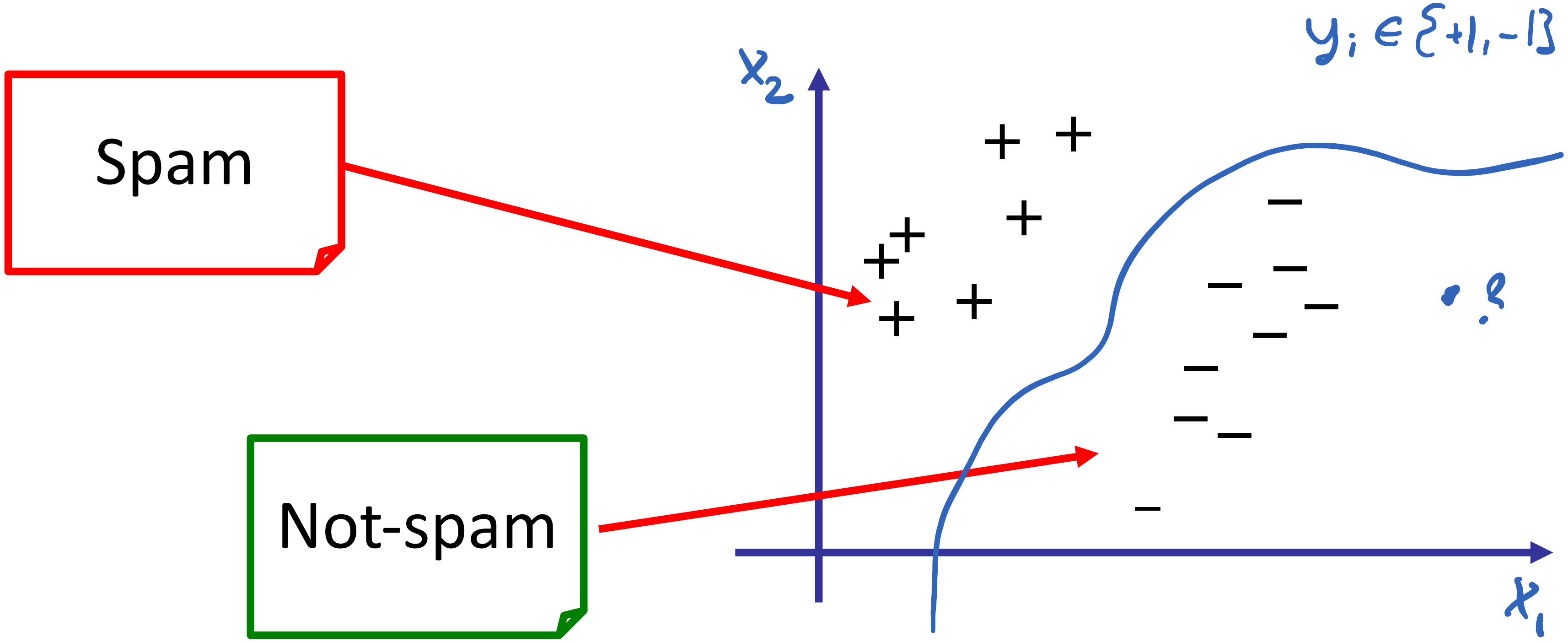
Classification

- Instance of supervised learning where *Y* is discrete (categorical)
- Want to assign data points *X*
  - Documents
  - Queries
  - Images
  - User visits
  - …

  a label Y (spam/not-spam; topic such as sports, politics, entertainment, click/no-click etc.)

For now, focus on binary classification

Illustration of binary classification

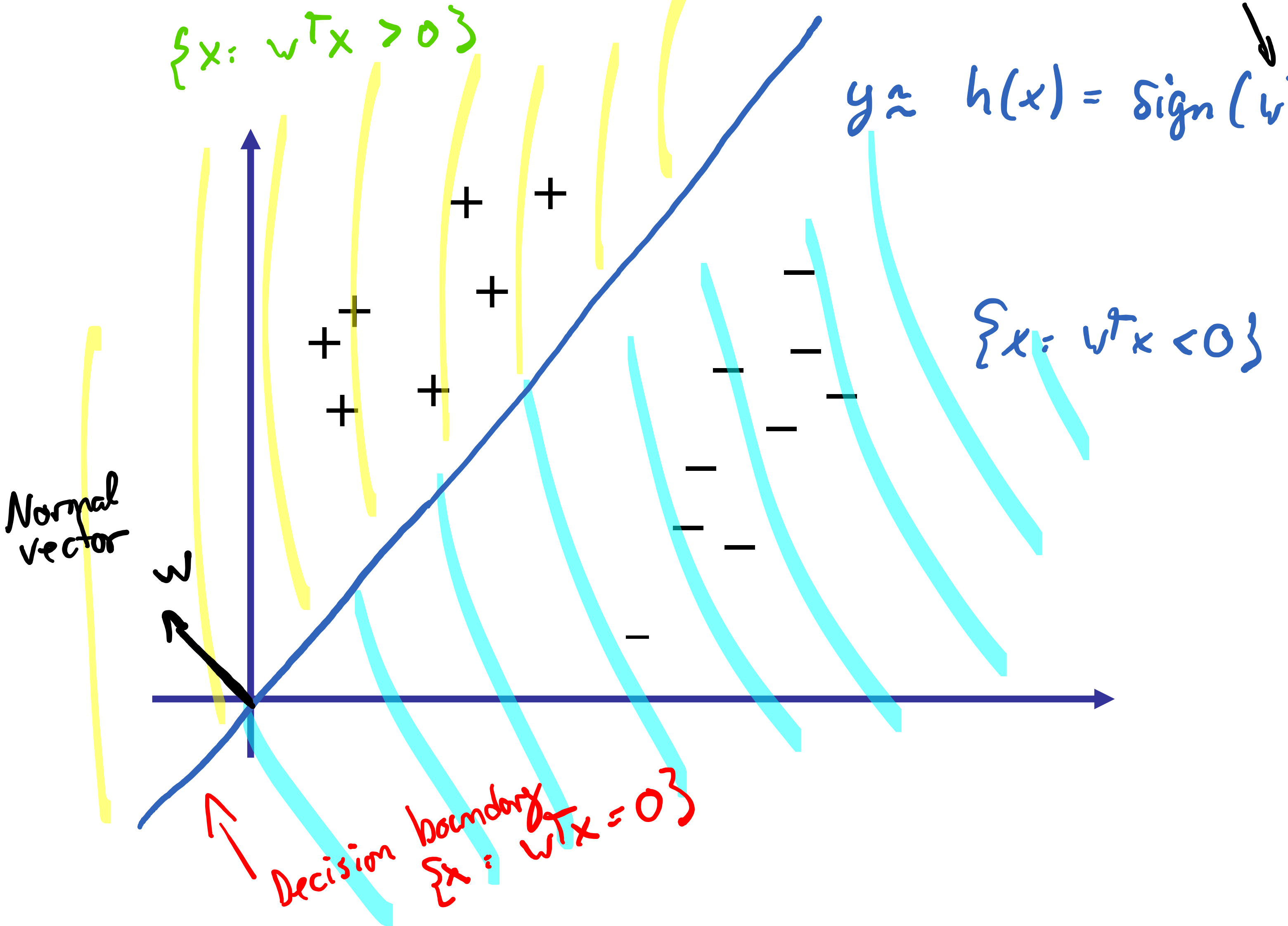


- **Input**: Labeled data set (e.g., rep. bag-of-words) with positive (+) and negative (-) examples
- **Output**: Decision rule (hypothesis)

Linear classifiers

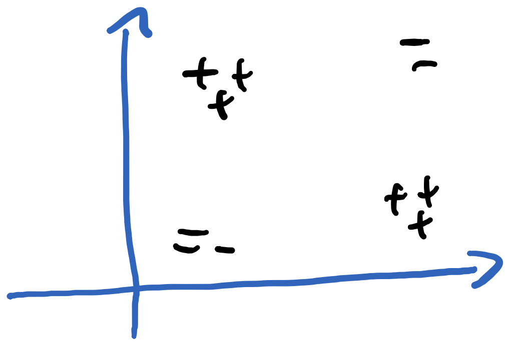
- Data set $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ $\quad w \in \mathbb{R}^d$

$y \approx h(x) = \text{sign}(w^T x)$



Why linear classification?

- Linear classification seems restrictive



- Especially in high-dimensional settings / when using the right features, often works quite well!
- Prediction is typically very efficient

Finding linear separators

- Want to write the search for a classifier as optimization problem
- What should we optimize?
- **First Idea**: Seek ***w*** that minimizes #mistakes

$D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$

$x_i \in \mathbb{R}^d,\ y_i \in \{+1, -1\}$

$$\hat{w} = \operatorname*{argmin}_{w \in \mathbb{R}} \underbrace{\sum_{i=1}^{n} \underbrace{[y_i \neq \text{sign}(w^T x_i)]}_{= \begin{cases} 1 & \text{if } y_i \neq \text{sign}(w^T x_i) \\ 0 & \text{otw} \end{cases}}}_{}$$

$\ell_{0/1}(w; x_i, y_i)$

$$\text{Sign}(z) = \begin{cases} +1 & \text{if } z \geq 0 \\ -1 & \text{otw.} \end{cases}$$

Optimization problem

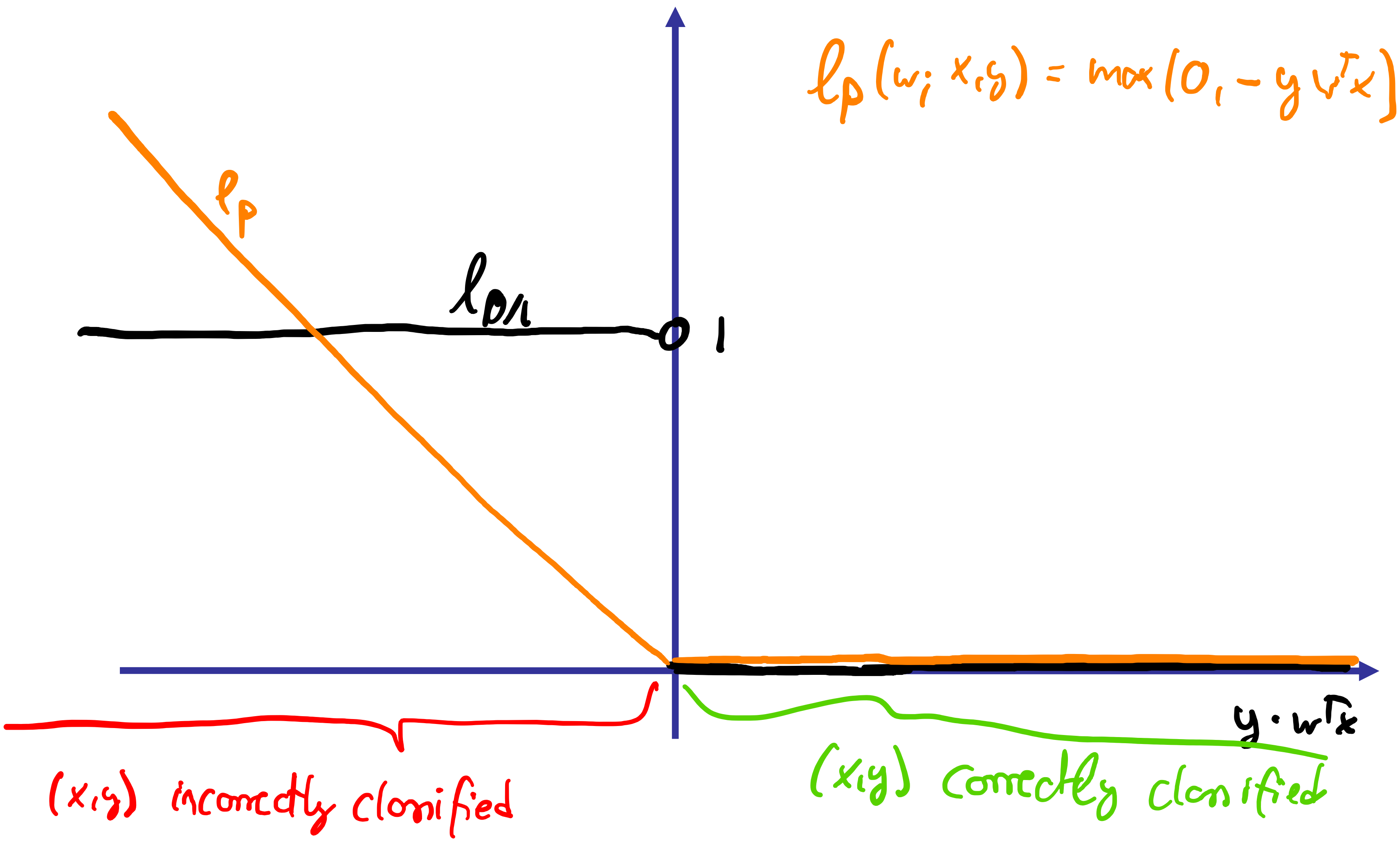
- **Goal**:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} [y_i \neq \text{sign}(\mathbf{w}^T \mathbf{x}_i)]$$

$$= \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \ell_{0/1}(\mathbf{w}; \mathbf{x}_i, y_i)$$

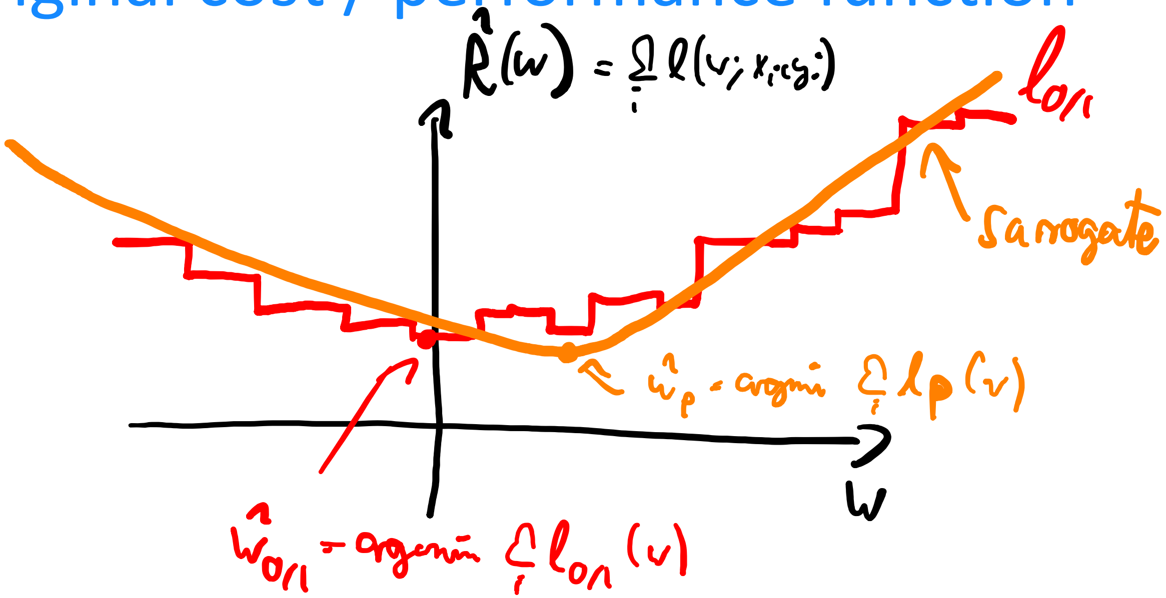
- **Challenge**: in contrast to squared loss, 0/1 loss is not convex (not even differentiable!)

A surrogate loss function



Key concept: Surrogate losses

- Replace intractable cost function that we care about (e.g., 0/1-loss) by tractable loss function (e.g., Perceptron loss) for sake of optimization / model fitting
- When evaluating a model (e.g., via cross-validation), use original cost / performance function



Surrogate optimization problem

- Instead of
$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \ell_{0/1}(\mathbf{w}; \mathbf{x}_i, y_i)$$

- Solve
$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \frac{1}{n} \boxed{\sum_{i=1}^{n} \ell_P(\mathbf{w}; \mathbf{x}_i, y_i)} \quad \hat{R}(w)$$

where $\ell_P(\mathbf{w}; y_i, \mathbf{x}_i) = \max(0, -y_i \mathbf{w}^T \mathbf{x}_i)$

is the Perceptron loss

Gradient descent

- Compute gradient of the Perceptron loss function

$$\hat{R}(w) = \sum_{i=1}^{n} \max(0, -y_i w^T x_i)$$

$$\nabla \hat{R}(w) = \sum_{i=1}^{n} \underbrace{\nabla_w \max(0, -y_i w^T x_i)}_{(*)}$$

$$(*) = \nabla_w \ell_p(w; x_i, y_i)$$

$$(*) = \begin{cases} 0 & \text{if } y_i w^T x_i \geq 0 \quad \text{i.e., correctly classified} \\ -y_i \cdot x_i & \text{otw.} \end{cases}$$

$$w_{t+1} \leftarrow w_t + \eta_t \sum_{\substack{i:(x_i,y_i) \text{ incorrectly} \\ \text{classified by } w}} y_i x_i$$

Stochastic gradient decent

- Computing the gradient requires summing over all data
- For large data sets, this is inefficient
- Moreover, our initial estimates are likely very wrong, and we can get a good (unbiased) gradient estimate by evaluating the gradient on few points

- Extreme case: Evaluate on only one randomly chosen point!

→ Stochastic gradient descent

Stochastic Gradient Descent

- 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

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

- Hereby, η_t is called learning rate
- Guaranteed to converge under mild conditions, if

$$\underline{\sum_t \eta_t = \infty} \quad \text{and} \quad \sum_t \eta_t^2 < \infty$$

E.g. $\eta_t = \frac{1}{t}$

$\eta_t = \min(c, \frac{c'}{t})$

The Perceptron algorithm

- Is just stochastic gradient descent (SGD) on the Perceptron loss function ℓ_P with learning rate 1

- **Theorem**: If the data is linearly separable, the Perceptron will obtain a linear separator

Perceptron Demo

Mini-batch SGD

- Using single points might have large variance in the gradient estimate, and hence lead to slow convergence.
- Can reduce variance by averaging over the gradients w.r.t. multiple randomly selected points (mini-batches)

Demo: SGD for regression

Adaptive learning rates

- Similar as in gradient descent, the learning rate is very important
- There exist various approaches for adaptively tuning the learning rate. Often times, these even use a different learning rate per feature
- Examples: AdaGrad, RMSProp, Adam, …

Supervised learning summary so far

Representation/ features | Linear hypotheses; nonlinear hypotheses with nonlinear feature transforms

Model/ objective: | Loss-function + Regularization

Squared loss, 0/1 loss, Perceptron loss

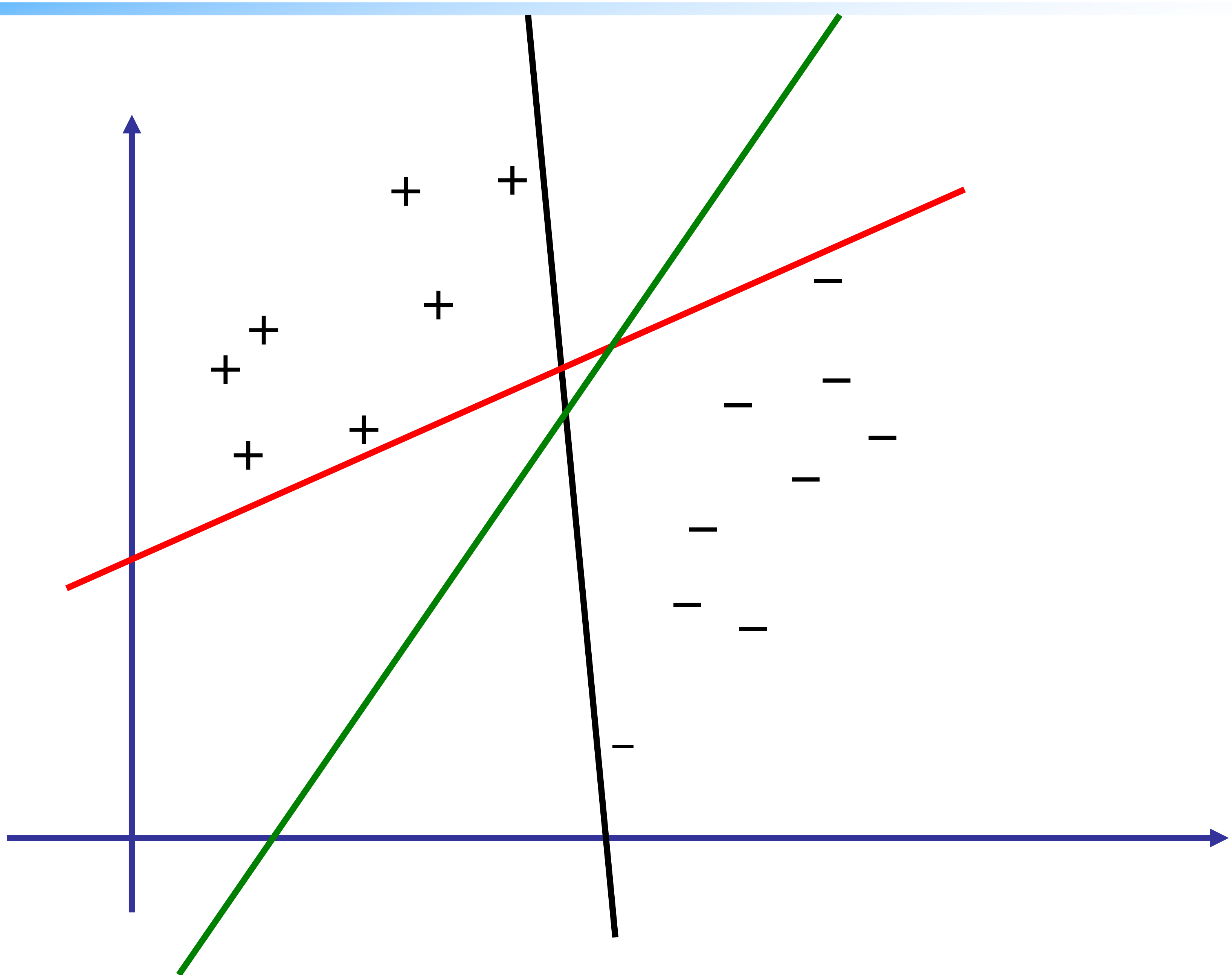
L^2 norm

Method: | Exact solution, Gradient Descent, (mini-batch) SGD

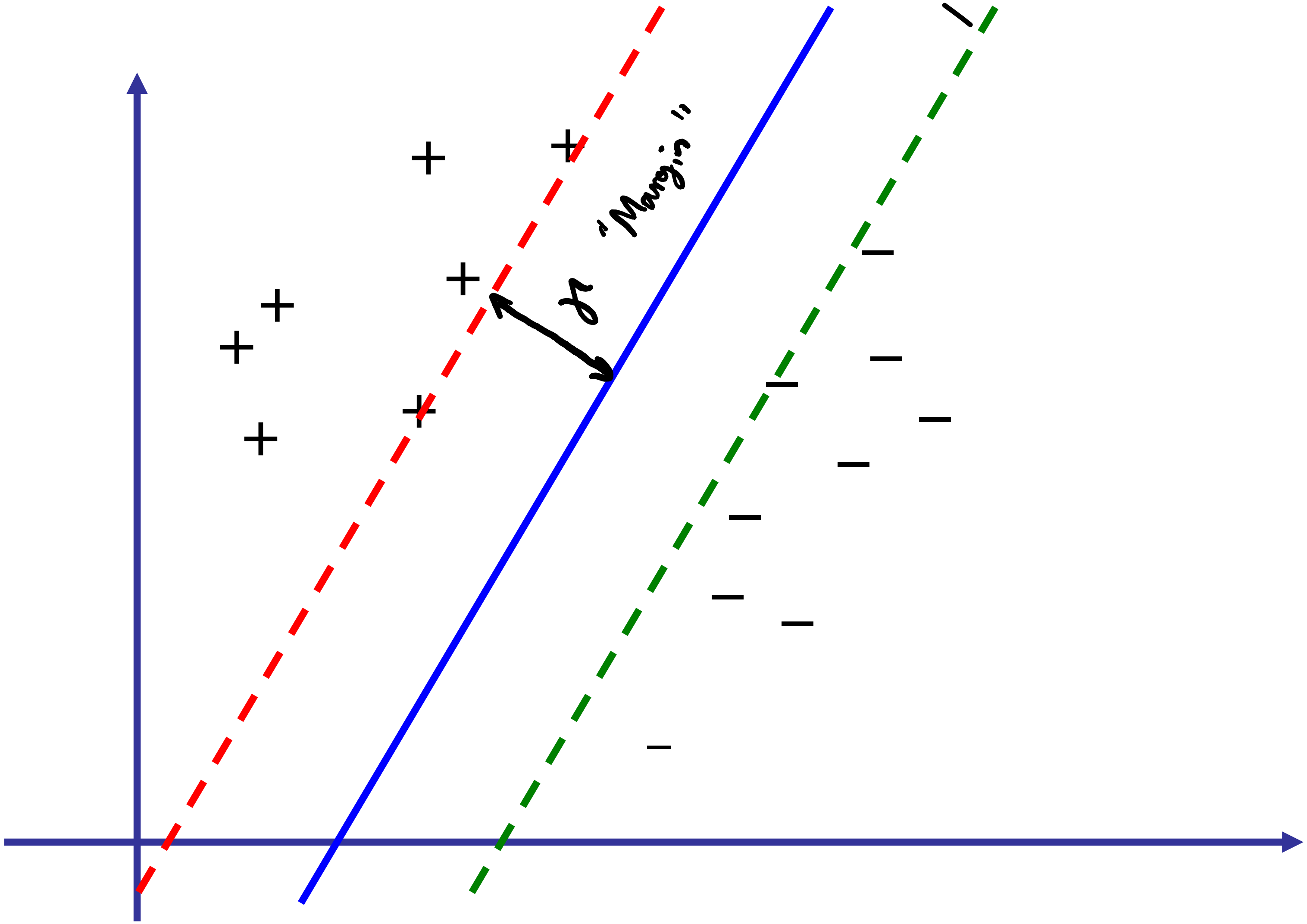
Evaluation metric: | Mean squared error, Accuracy

Model selection: | K-fold Cross-Validation, Monte Carlo CV

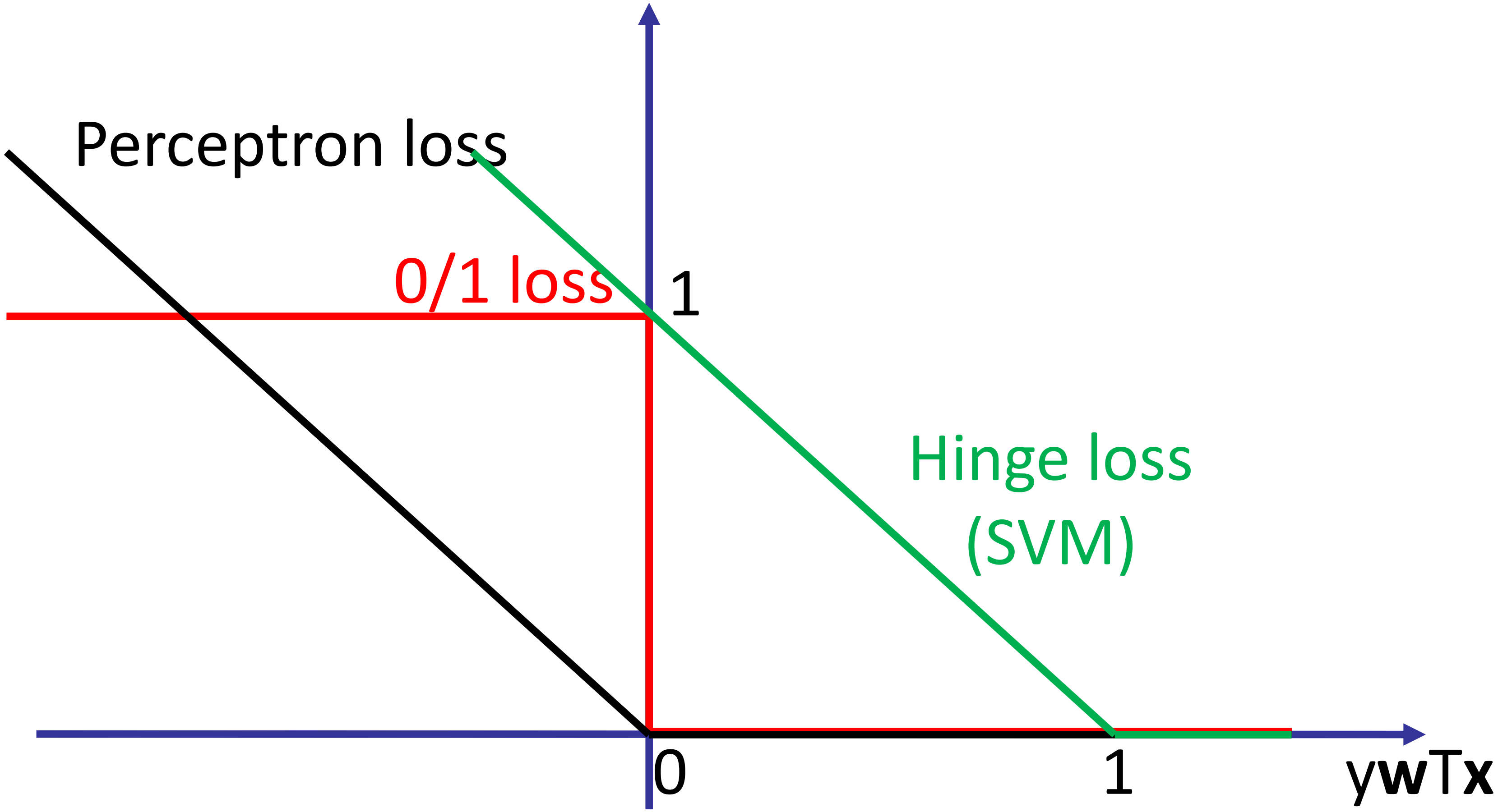
Which of these separators will the Perceptron "prefer"?



Support Vector Machines (SVMs): "max. margin" linear classification



Hinge vs. Perceptron loss



Hinge loss upper bounds #mistakes; encourages „margin“

$$\ell_H(\mathbf{w};\mathbf{x},y) = \max\{0, 1 - y\mathbf{w}^T\mathbf{x}\}$$

SVM vs. Perceptron

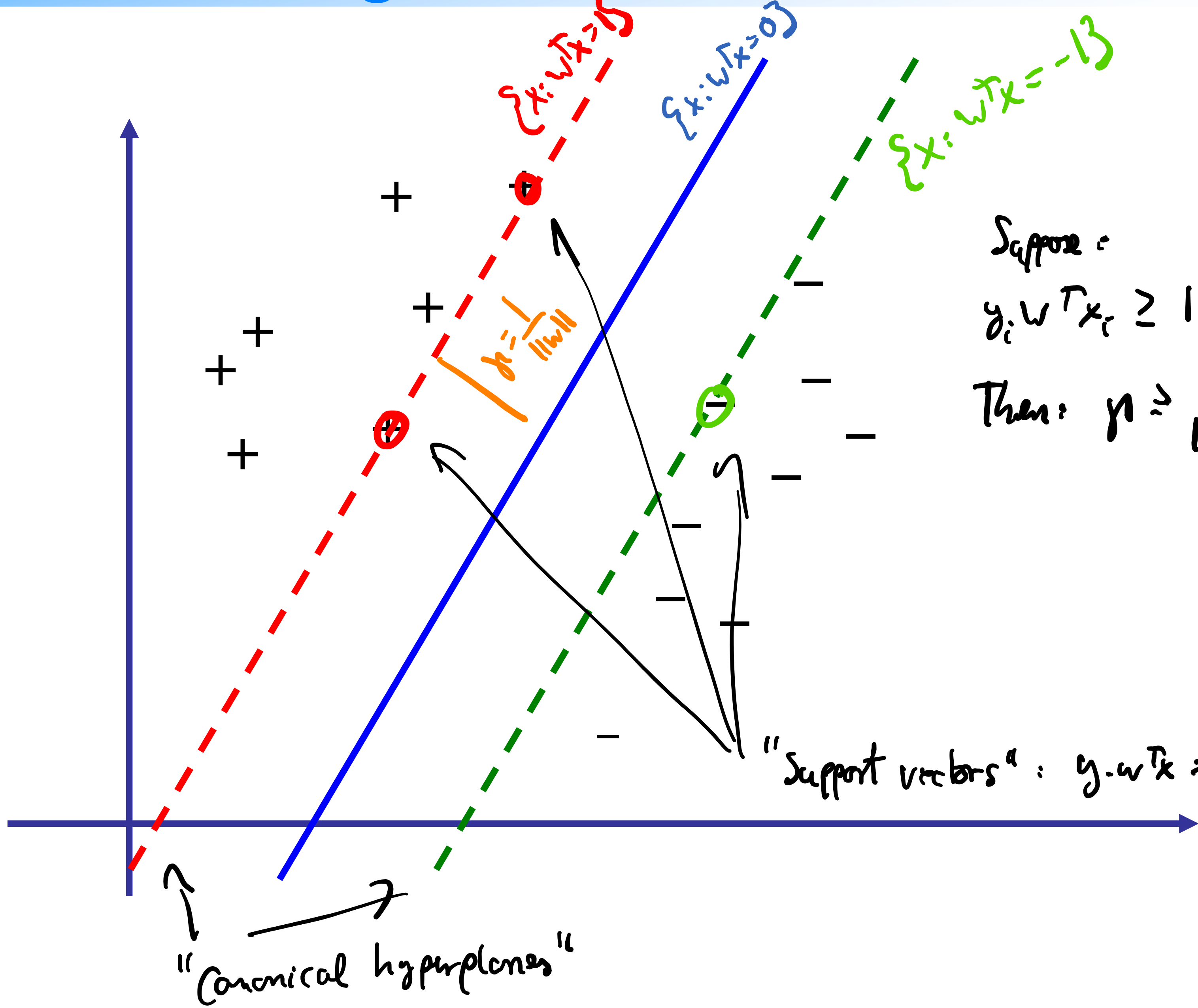
- Perceptron:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \max\{0, -y_i \mathbf{w}^T \mathbf{x}_i\}$$

- Support vector machine (SVM):

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \max\{0, 1 - y_i \mathbf{w}^T \mathbf{x}_i\} + \lambda ||\mathbf{w}||_2^2$$

Support Vector Machines (SVMs): "max. margin" linear classification



Suppose:

$y_i w^T x_i \geq 1 \quad \forall i$

Then: $\gamma \geq \frac{1}{\|w\|}$

Support vector machines

- Widely used, very effective linear classifer
- Almost like Perceptron. Only differences:
  - Optimize slightly different, shifted loss (hinge loss)
  - Regularize weights (like ridge regression)
- Can optimize via stochastic gradient descent
- Safe choice for learning rate: $\eta_t = \frac{1}{\lambda t}$
- More details in Advanced Machine Learning lecture