ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Data Mining
Learning from Large Data Sets

Lecture 4 – Large-scale supervised learning

263-5200-00L

Andreas Krause

Announcement

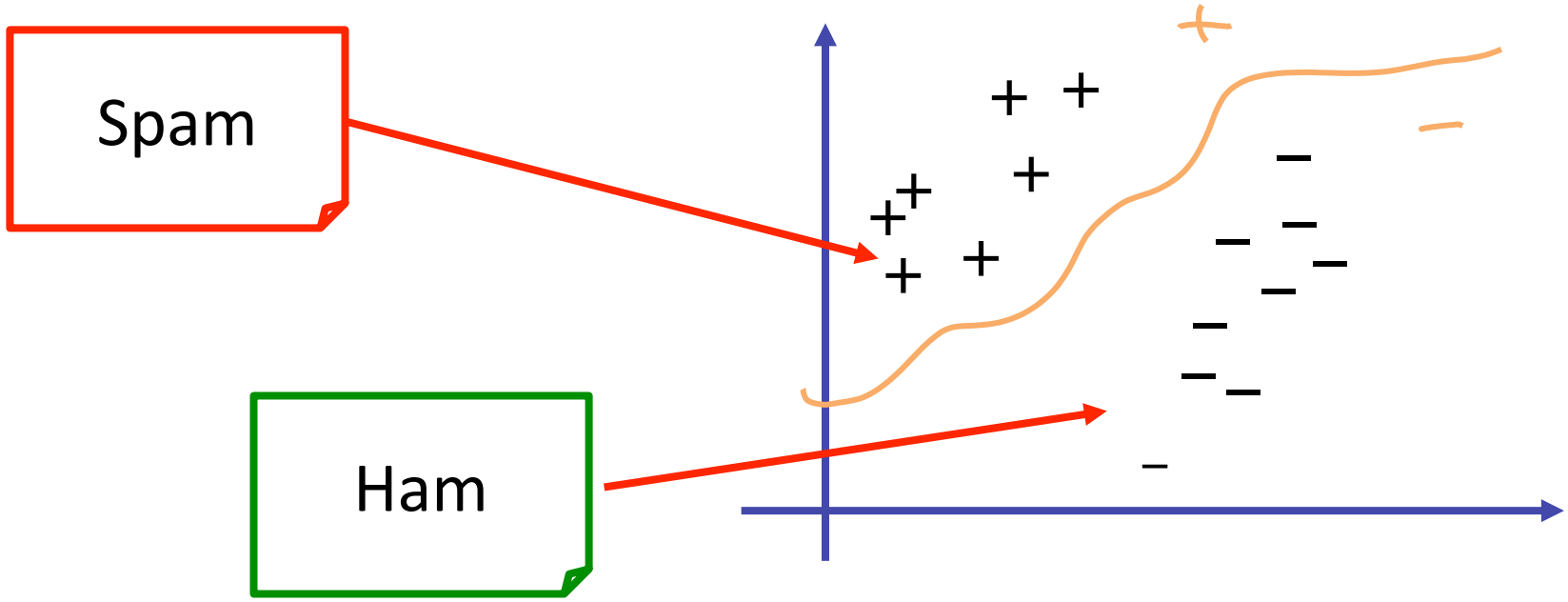
- Homework 2 out tomorrow

- Additional lecture on Friday 15:00 (not this week)?

Course organization

- **Retrieval**
  - Given a query, find "most similar" item in a large data set
  - *Applications*: GoogleGoggles, Shazam, …
- **Supervised learning** (Classification, Regression)
  - Learn a concept (function mapping queries to labels)
  - *Applications*: Spam filtering, predicting price changes, …
- **Unsupervised learning** (Clustering, dimension reduction)
  - Identify clusters, "common patterns"; anomaly detection
  - *Applications*: Recommender systems, fraud detection, …
- **Learning with limited feedback**
  - Learn to optimize a function that's expensive to evaluate
  - *Applications*: Online advertising, opt. UI, learning rankings, …

Document classification

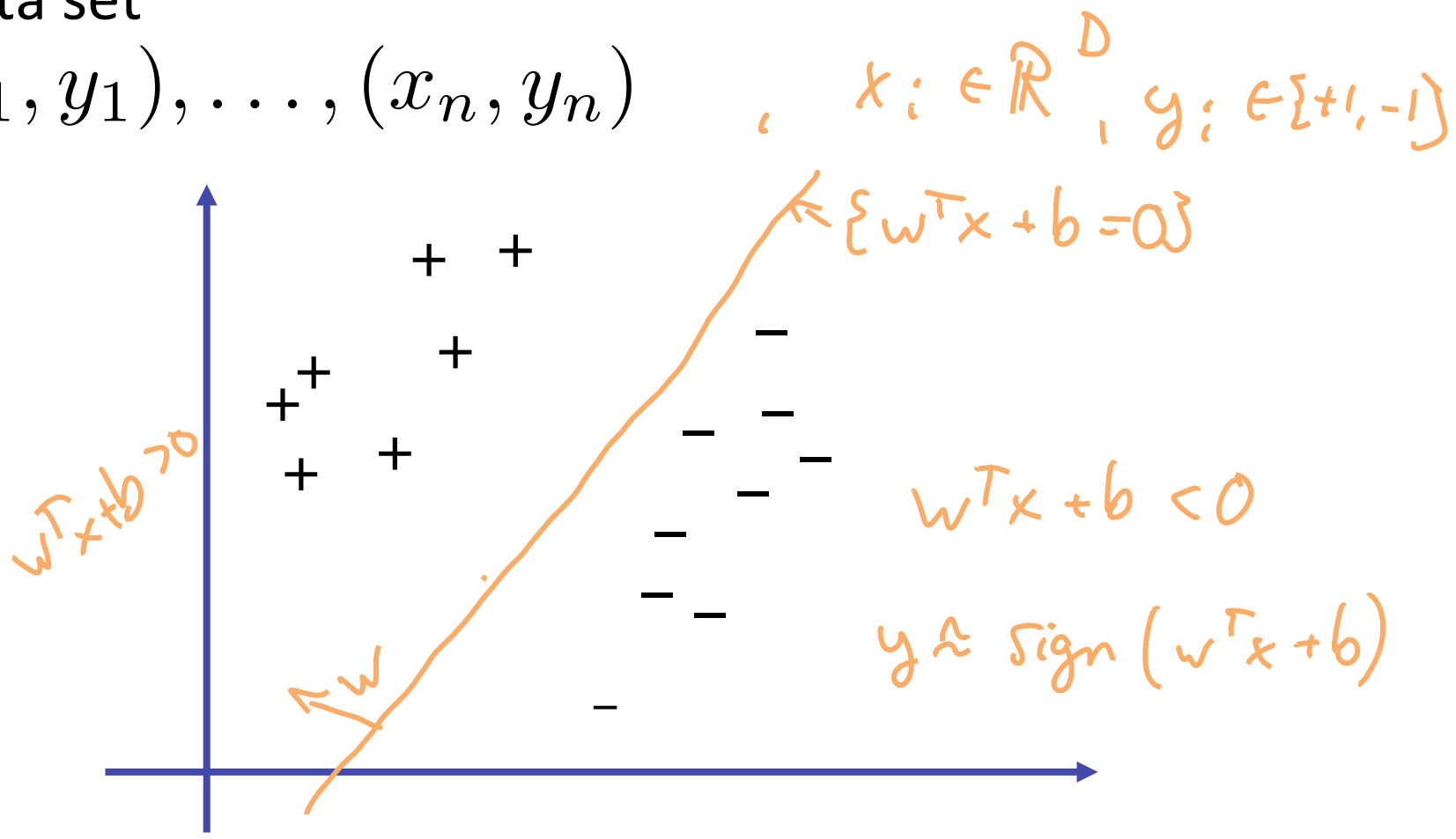


- **Input**: Labeled data set (e.g., rep. bag-of-words) with positive (+) and negative (-) examples
- **Output**: Decision rule (hypothesis)

Linear classifiers

- Data set

$$(x_1, y_1), \ldots, (x_n, y_n)$$

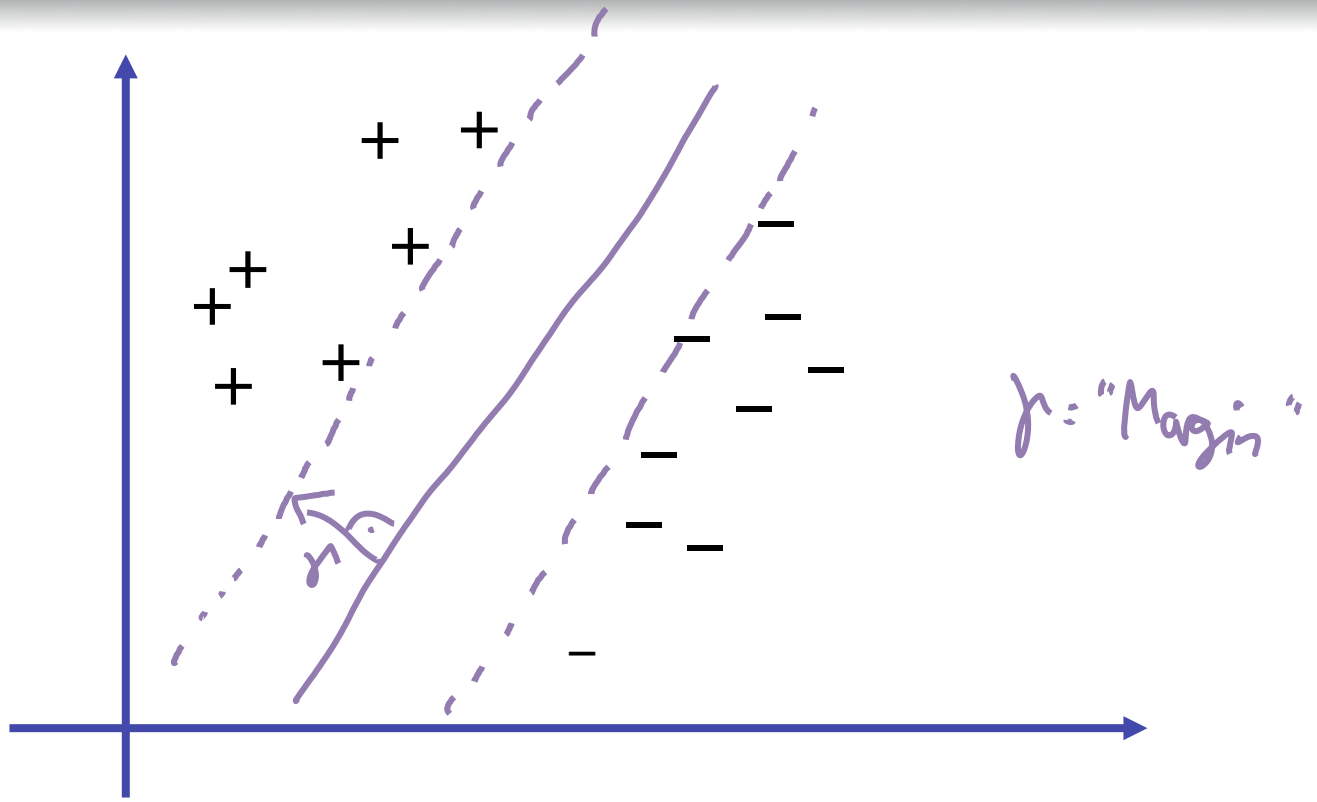


Why linear classification?

- Linear classification seems restrictive:

- In high-dimensional settings, often works extremely well
- Can be trained extremely efficiently, even on massive data
- We'll learn about the state of the art:
  Support Vector Machines (SVM)

Large margin classification

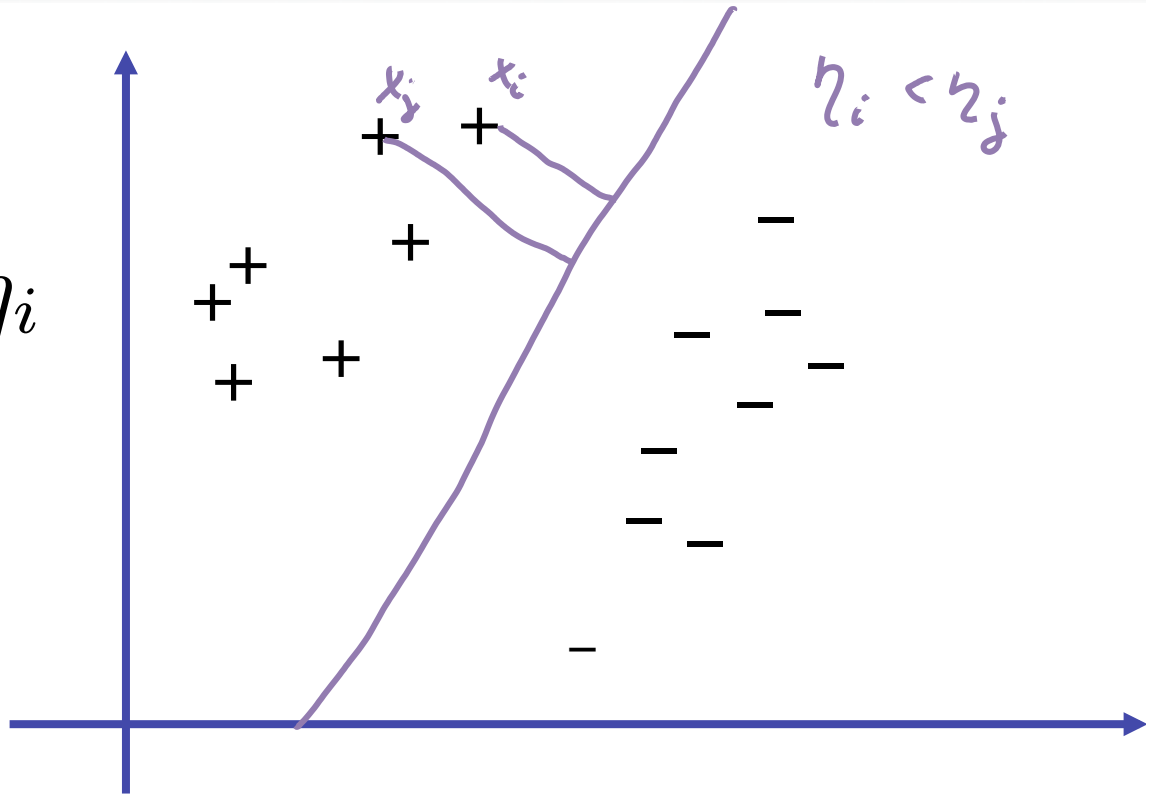


- Want to maximize "margin" to closest example!
- Turns out to be the "right" thing to do
  - Larger margin ➔ Better generalization

Large margin classification

- Confidence in example i:

$$y_i(w^T x_i + b) \equiv \eta_i$$



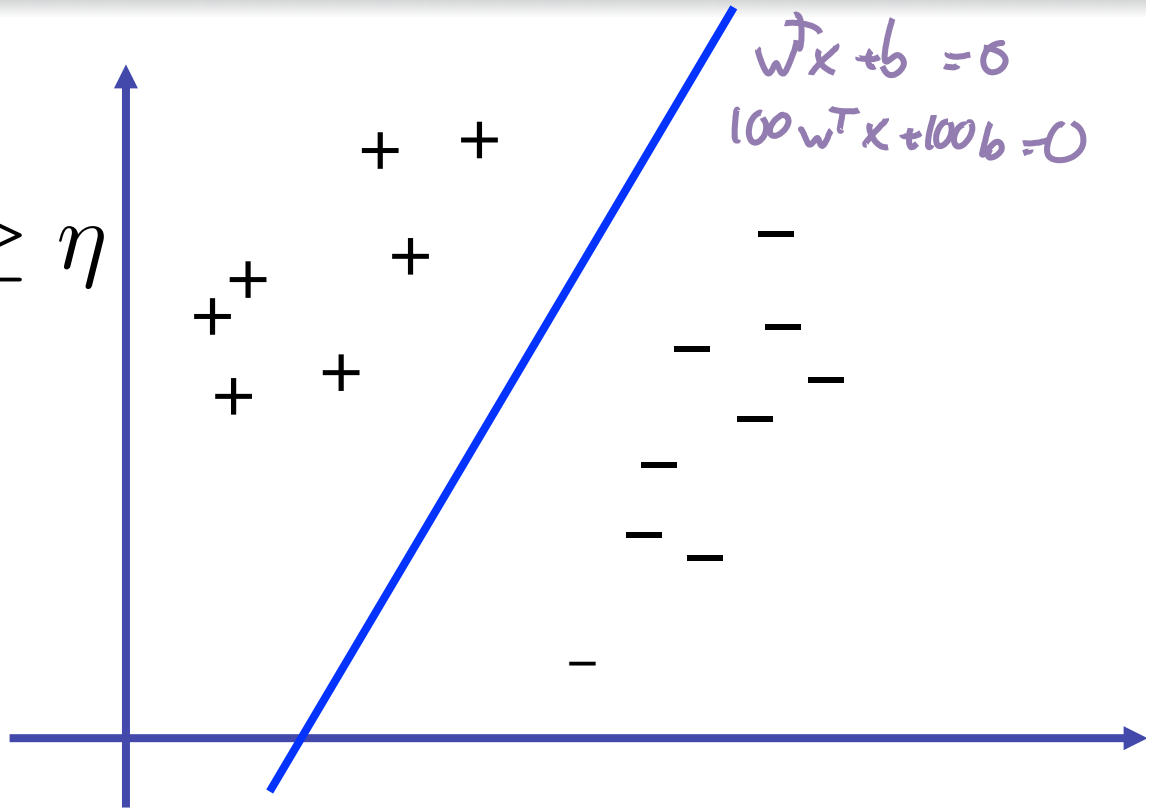
- Intuitively, want

$$\max_{w,b} \min_i \eta_i$$

Nonuniqueness

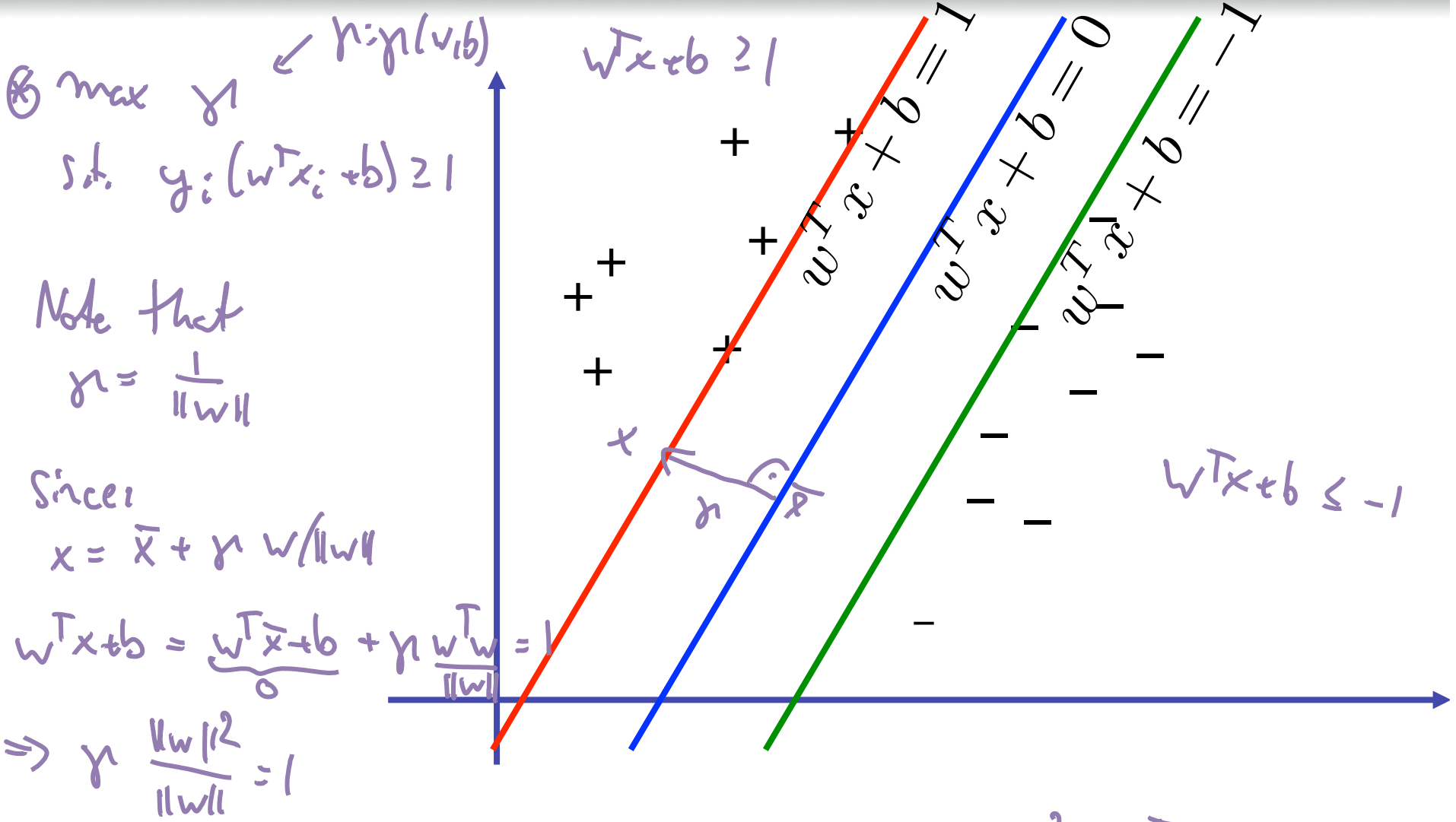
$$\max_{w,b,\eta} \eta \quad = \infty$$

$$\text{s.t.} (w^T x_i + b) y_i \geq \eta$$



So far, our notion of confidence is not yet well defined!
Need normalization!

Canonical hyperplanes



$\gamma_i : y_i (w^T x_i + b)$

$\max \gamma$

s.t. $y_i (w^T x_i + b) \geq 1$

Note that

$\gamma = \frac{1}{\|w\|}$

Since:

$x = \bar{x} + \gamma \, w/\|w\|$

$w^T x + b = \underbrace{w^T \bar{x} + b}_{0} + \gamma \frac{w^T w}{\|w\|} = 1$

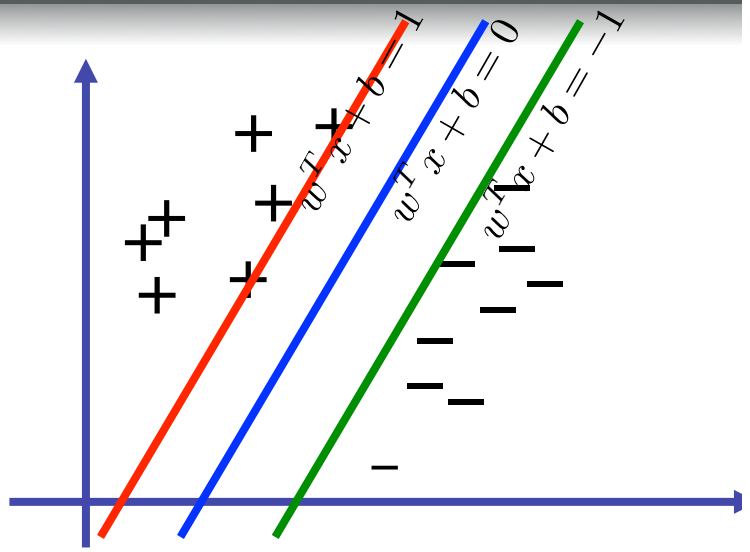
$\Rightarrow \gamma \frac{\|w\|^2}{\|w\|} = 1$

Thus (*) is equivalent to minimizing $\|w\|^2 = w^T w$

Support Vector Machine

$$\min_{w,b} w^T w$$

$$\text{s.t.} y_i(w^T x_i + b) \geq 1$$



- How can we solve this optimization?
- What about local minima?

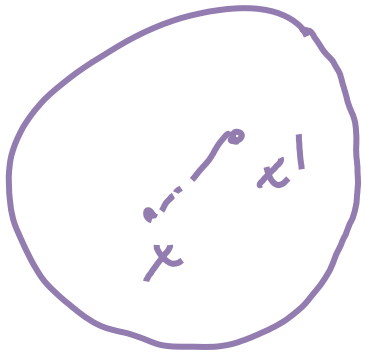
- This is a convex (quadratic) program

Convex sets

A subset $S \subseteq \mathbb{R}^d$

is called convex if

$$\forall x, x' \in S,\ \lambda \in (0,1): \underbrace{\lambda x + (1-\lambda)x' \in S}_{\text{Convex combination}}$$



Convex

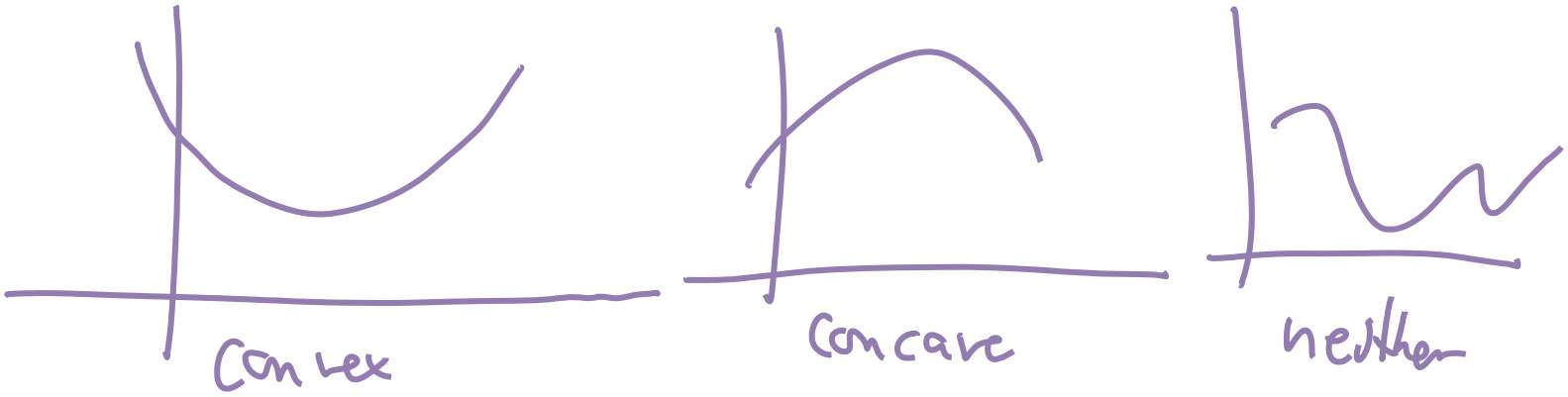
not convex

Convex functions

A function $f : \mathbb{R}^d \longrightarrow \mathbb{R}$

is called convex if

$$\forall\, x, x' \in \mathbb{R}^d,\ \lambda \in [0,1]: \quad f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x')$$



Convex optimization

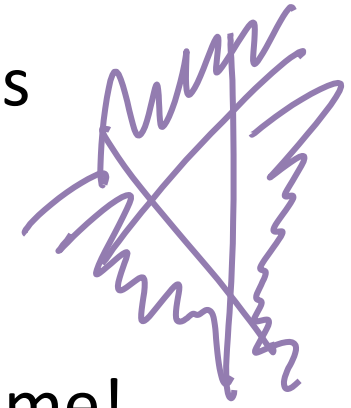
- Given a convex function f and a convex set S

  want to solve $\min f(x)$ s.t. $x \in S$

- This is called a convex optimization problem

- Often, S specified using linear inequalities

$$S = \{x \in \mathbb{R}^d : a_i^T x \leq b_i\}$$



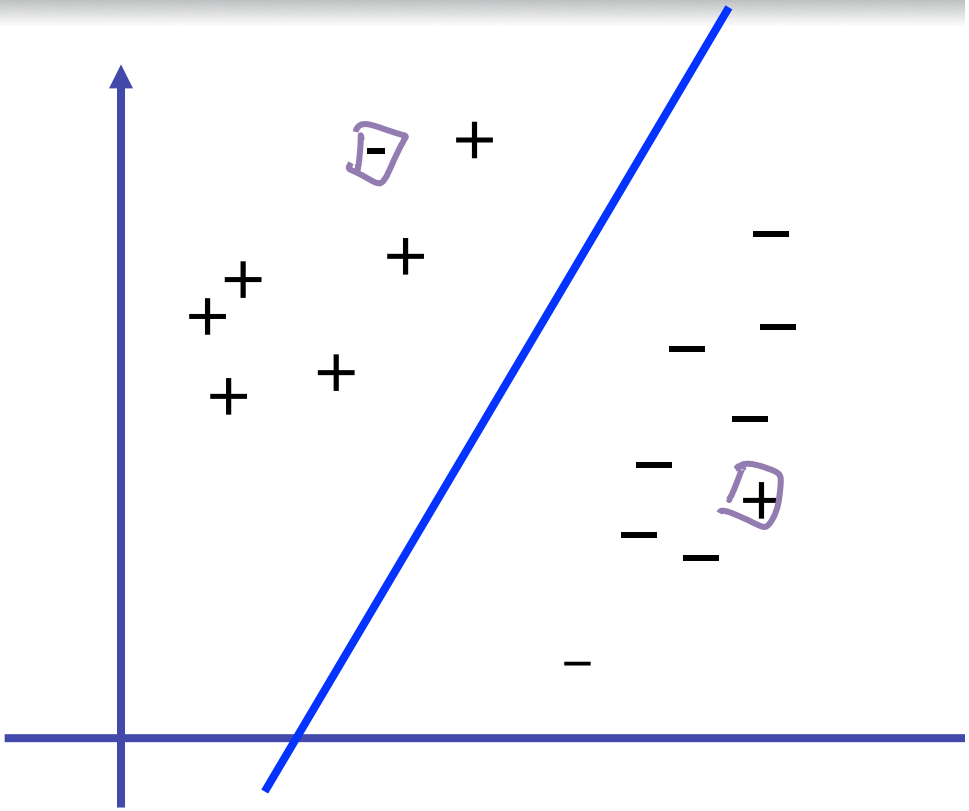
- Can solve such problems in polynomial time!

Dealing with noise

$$\min_{w,b} w^T w \quad + C \#\text{mistakes}$$

$$\text{s.t.} y_i(w^T x_i + b) \geq 1$$

Not convex!

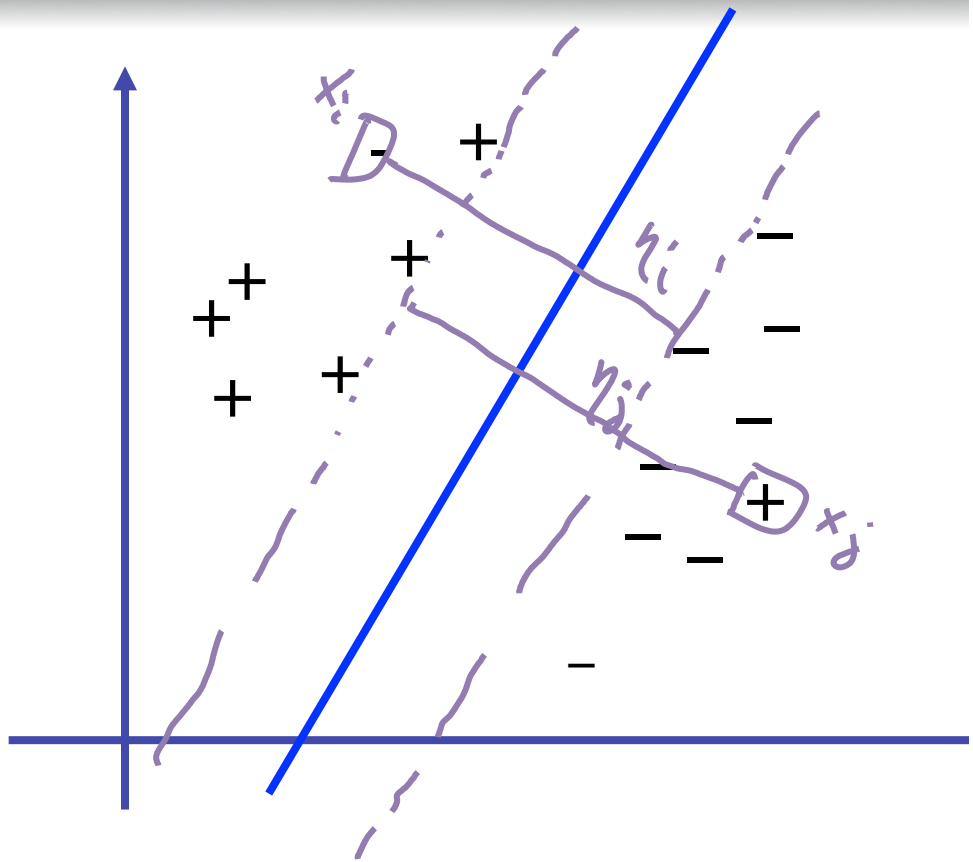


Dealing with noise: Slack variables

$$\min_{w,b} w^T w + C \sum_i \eta_i$$

$$\text{s.t.} y_i(w^T x_i + b) \geq 1 - \eta_i$$

Convex!



Dealing with massive data sets

- Are we done??
- Complexity of quadratic programming
  - Naïve implementations: $\Omega(n^3)$
- What if the data doesn't even fit in memory??
- Will see how one can reformulate the SVM optimization problem so that one can solve it on web-scale problems…

Hinge loss

- SVM with slack variables

$$\min_{w,b,\xi \geq 0} \; w^T w + C \sum_i \xi_i$$

$$\text{s.t.} \; y_i (w^T x_i + b) \geq 1 - \xi_i$$

- Equivalent formulation:

$$\xi_i = \begin{cases} 0 & \text{if} \quad y_i (w^T x_i + b) \geq 1 \\ 1 - y_i (w^T x_i + b) & \text{otherwise} \end{cases}$$

$$= \max \left(0, 1 - y_i (w^T x_i + b)\right)$$

Hinge loss

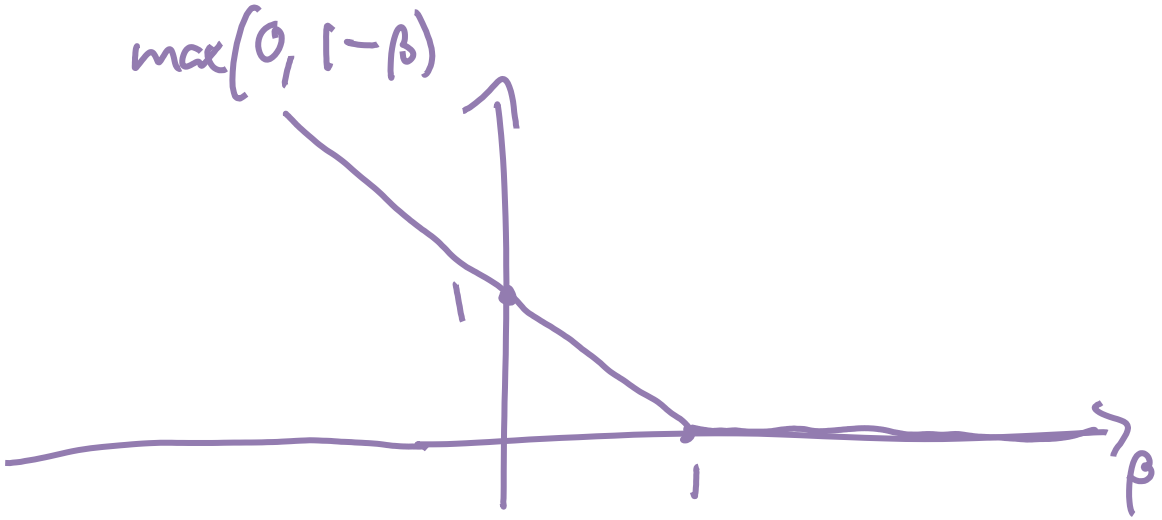
$$\min_{w,b} w^T w + C \sum_i \max\left(0, 1 - y_i(w^T x_i + b)\right)$$

"Margin"

Hinge loss

β

Data fit



Yet another SVM formulation

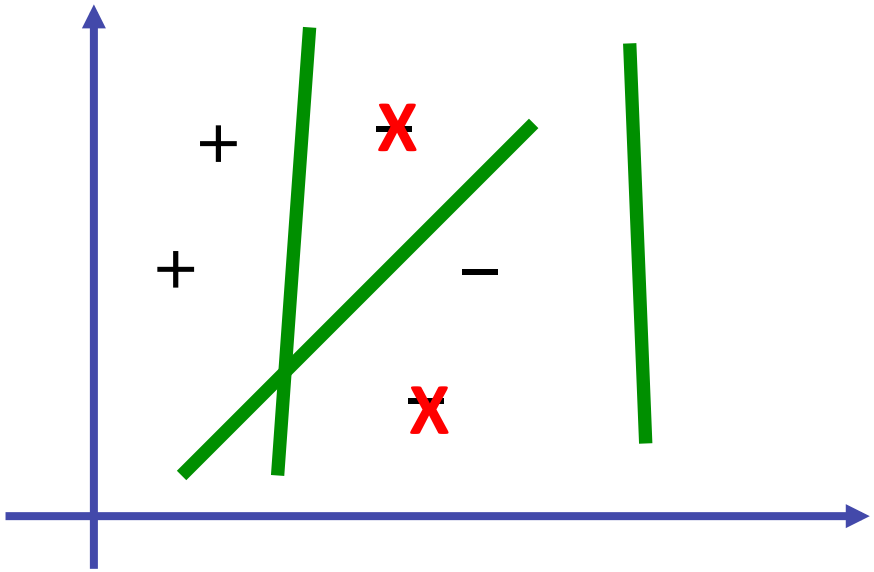
$$\min_{w,b} w^T w + C \sum_i \max\left(0, 1 - y_i(w^T x_i + b)\right)$$

$\Leftrightarrow \min \lambda w^T w + \sum_i \max\left(0, 1 - y_i(w^T x_i + b)\right)$

$\Leftrightarrow \min \sum_i \max\left(0, 1 - y_i(w^T x_i + b)\right)$

s.t. $\|w\| \leq \frac{1}{\lambda}$

Online classification



X: Classification error

- Data arrives sequentially
- Need to classify one data point at a time
- Use a different decision rule (lin. separator) each time
- Can't remember all data points!

Online SVM optimization

- Keep track of hyperplane parameters w
- Each round
  - New data point arrives
  - Classify according to $\text{sign}(w_t^T x_t + b_t)$
  - Incur loss $\ell_t = \max(0, 1 - y_t(w_t^T x_t + b_t))$
  - Update w_t and b_t based on (x_t, y_t)
- Best we could have done:

$$L^* = \min_{w:||w||\leq 1/\lambda} \sum_{t=1}^{T} \max\Big(0, 1 - y_t(w_t^T x_t + b_t)\Big)$$

- Our regret: $R_T = \sum_{t=1}^{T} \ell_t - L^*$

Generally: Online convex programming

- Input:
  - Feasible set $S \subseteq R^d$
  - Starting point $w_0 \in S$
- Each round t do
  - Pick new feasible point $w_t \in S$
  - Receive convex function $f_t : S \to \mathbb{R}$
  - Incur loss $\ell_t = f_t(w_t)$
- Regret:

$$R_T = \left(\sum_{t=1}^{T} \ell_t\right) - \min_{w \in S} \sum_{t=1}^{T} f_t(w)$$

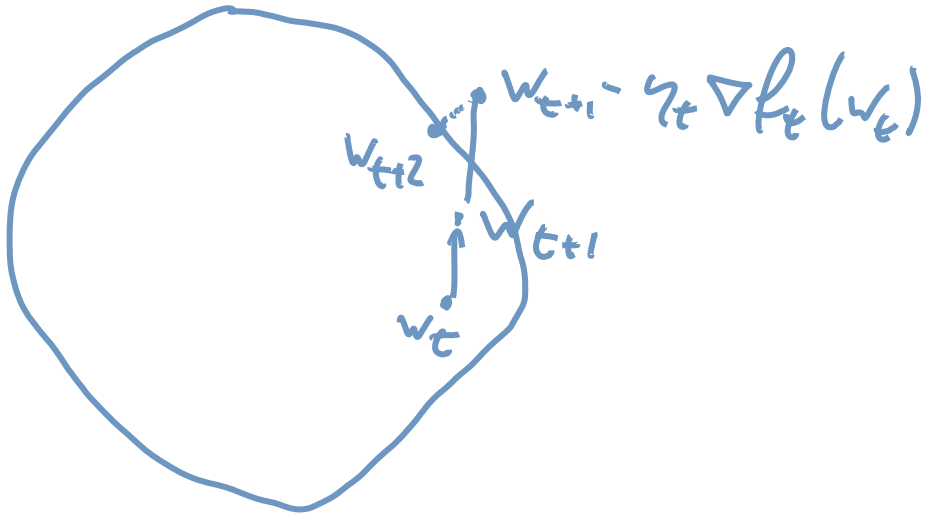
Solve: $\min \sum_t f_t(x)$ s.t. $x \in S$

E.g., SVM

Online convex programming

- Simple update rule:

$$w_{t+1} = w_t - \eta_t \nabla f_t(w_t)$$



$$\text{Proj}_S(x) = \operatorname*{argmin}_{x' \in S} \|x' - x\|_2$$

- How well does this simple algorithm do??

No regret algorithms

- An online algorithm is called no-regret if

$$R_T/T \to 0$$

for any sequence of functions $f_1, \ldots, f_T$

- For SVMs, this means:
  - The average error compared to solving the (expensive) quadratic program goes to zero
  - This is independent of how we process the data set!!!

Regret for online convex programming

Theorem [Zinkevich '03]

Let $f_1, \ldots, f_T$ be an arbitrary sequence of convex functions with feasible set S

Set $\eta_t = 1/\sqrt{t}$

Then, the regret of online convex programming is bounded by

$$R_T \leq \frac{||S||^2\sqrt{T}}{2} + \left(\sqrt{T} - \frac{1}{2}\right)||\nabla f||^2$$

Yet another SVM formulation

$$\min_{w,b} w^T w + C \sum_i \max\big(0, 1 - y_i(w^T x_i + b)\big)$$

$$\min_{w,b} \sum_i \max\big(0, 1 - y_i(w^T x_i + b)\big)$$

$$\text{s.t.} ||w||_2 \leq \frac{1}{\lambda}$$

Online convex programming for SVM

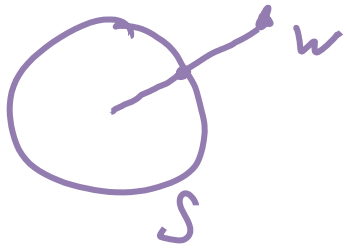
$$w_{t+1} = \text{Proj}_S\left(w_t - \eta_t \nabla f_t(w_t)\right)$$

- Feasible set: $S = \{w : ||w|| \leq \frac{1}{\lambda}\}$

- Projection:

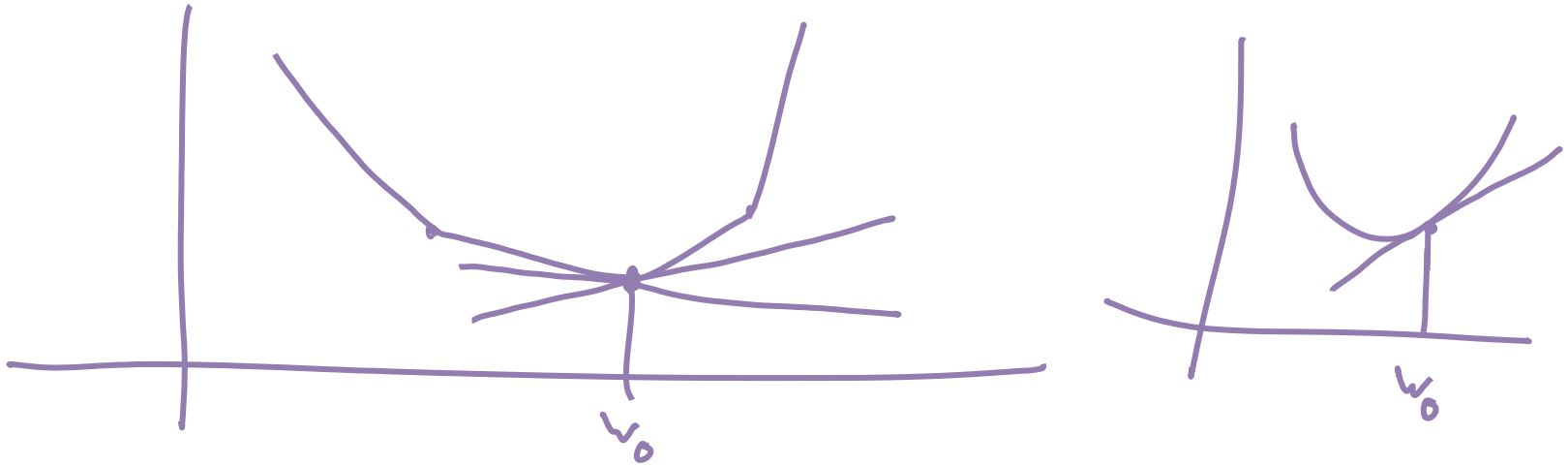
$w \in \mathbb{R}^d$

$$\text{Proj}_S(w) = \begin{cases} w & \text{if } w \in S \\ \frac{w}{||w||} \cdot \frac{1}{\lambda} & \text{if } w \notin S \end{cases}$$



- Gradient: $f_t(w) = \max(0, 1 - y_t(w^T x_t))$

Subgradients



f convex

$v \in \partial f(w_0)$ iff: $\forall w \in S: \; f(w) \geq f(w_0) + v^T(w - w_0)$

Subgradient for SVM

- Hinge loss: $f_t(w) = \max(0, 1 - y_t(w^T x_t + b))$

- Subgradient:

If	$\partial_w f_t(w)$
$1 - y_t(w^T x_t + b) < 0$	0
$1 - y_t(w^T x_t + b) > 0$	$-y_t x_t$

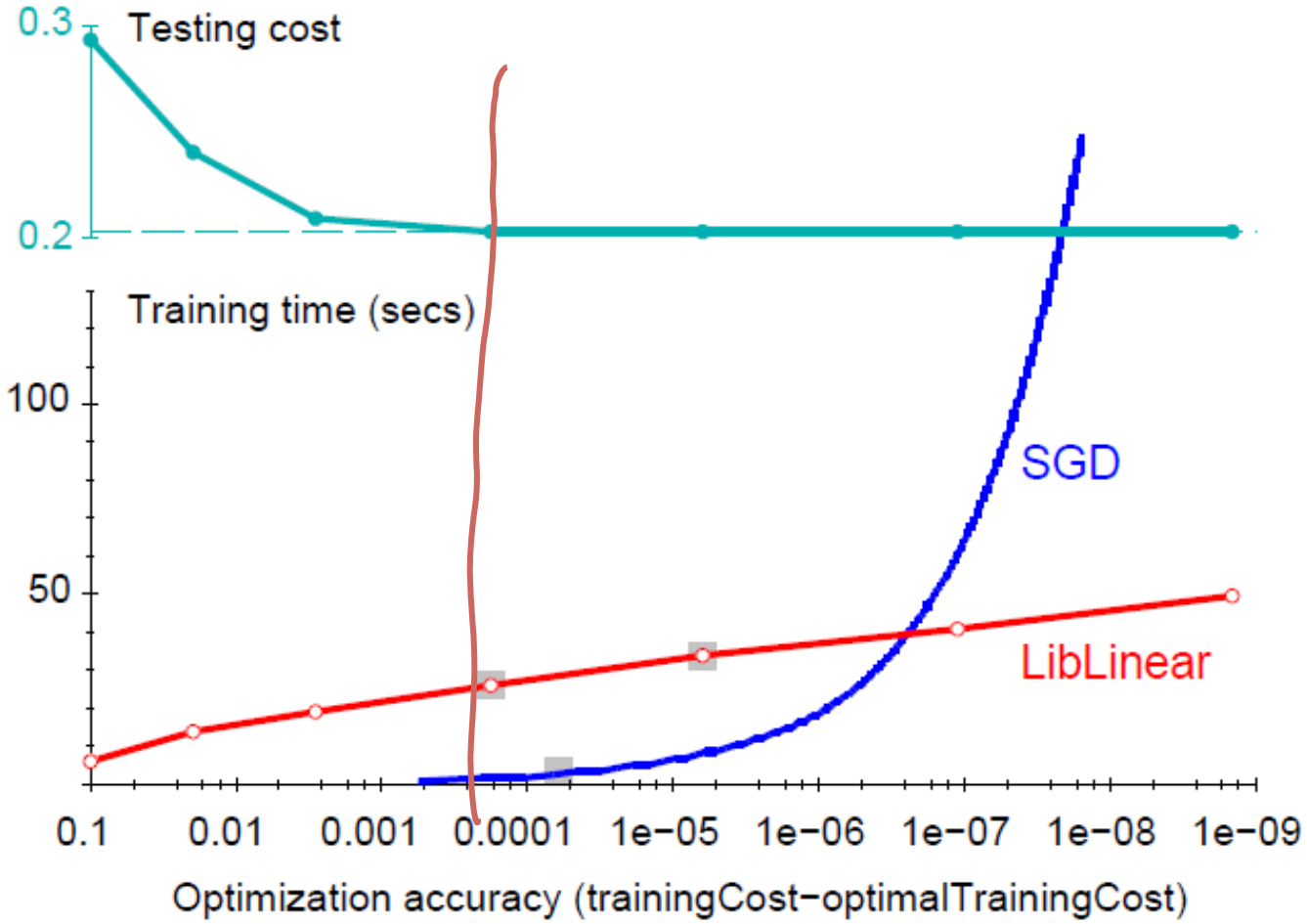
$$w_{t+1} = \text{Proj}_S\left(w_t - \eta_t \partial_w f_t(w_t)\right)$$

Example [Bottou]

- Stochastic gradient descent
  - Online convex programming with training samples picked at random
- Data set:
  - Reuters RCV1
  - 780k training examples, 23k test examples
  - 50k dimensions

	Training Time	Primal cost	Test Error
SVMLight	23,642 secs	*0.2275*	*6.02%*
SVMPerf	66 secs	0.2278	6.03%
SGD	1.4 secs	*0.2275*	*6.02%*

Error


0.3
Testing cost
0.2
Training time (secs)
100
50
SGD
LibLinear
0.1
0.01
0.001
0.0001
1e−05
1e−06
1e−07
1e−08
1e−09
Optimization accuracy (trainingCost−optimalTrainingCost)