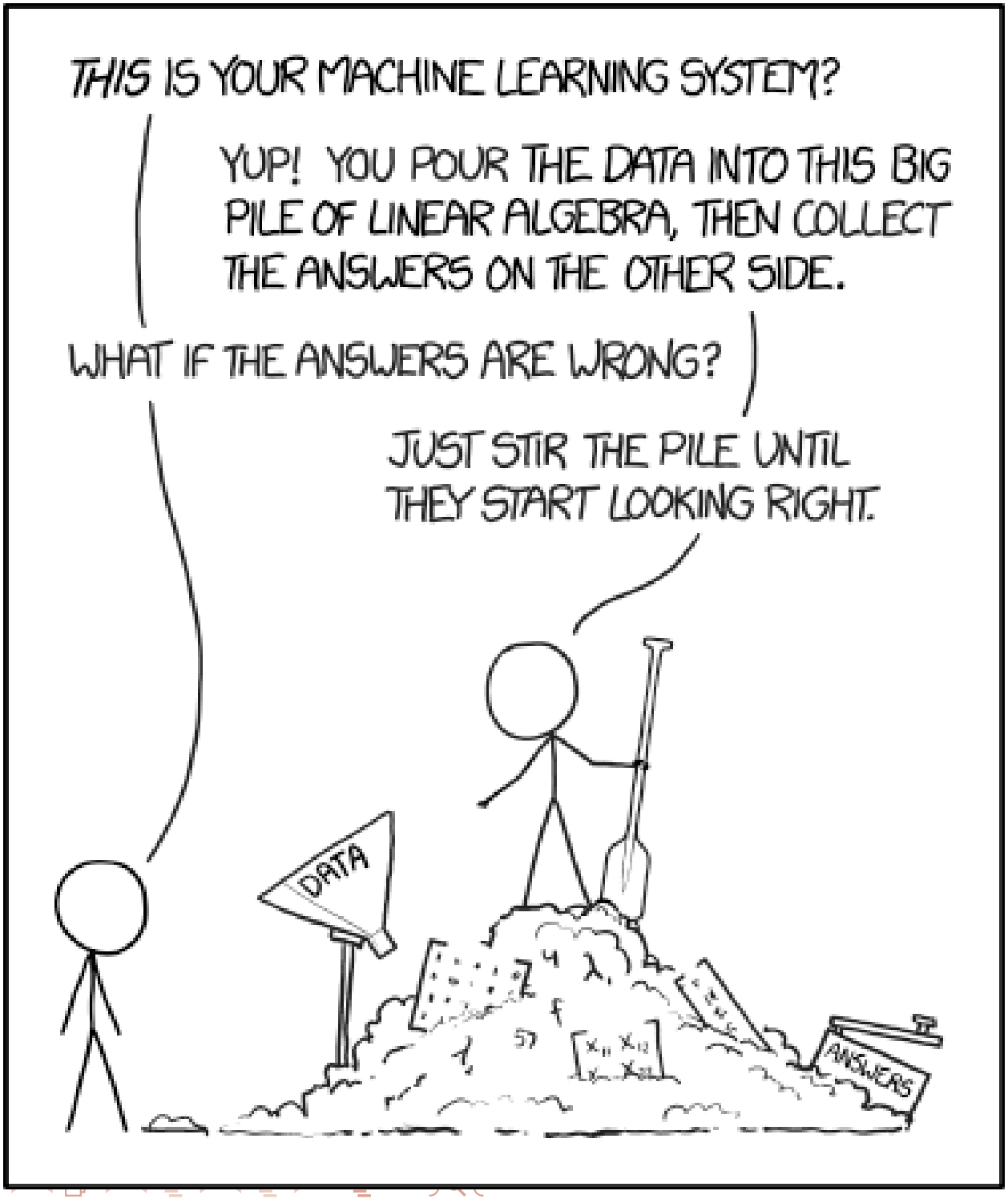
Optimisation strategies for machine learning - harnessing inexactness

Tristan van Leeuwen

Universiteit Utrecht



Universiteit Utrecht

Outline

- From Learning to Optimisation
- A Prototype algorithm
- Harnessing inexactness
- Wrap-up

Universiteit Utrecht

From Learning to Optimisation

Universiteit Utrecht

Supervised learning

We are given

- *samples* $\mathbf{x}^{(i)} \in \mathbb{R}^n$,
- *labels* $y^{(i)} \in \mathbb{R}$

for $i = 1, \ldots, m$.

Universiteit Utrecht

Supervised learning

We are given

- *samples* $\mathbf{x}^{(i)} \in \mathbb{R}^n$,
- *labels* $y^{(i)} \in \mathbb{R}$

for $i = 1, \ldots, m$.

The goal is to construct a *function*

$$f(\mathbf{x}^{(i)}) \approx y^{(i)}$$

Universiteit Utrecht

Supervised learning

We are given

- *samples* $\mathbf{x}^{(i)} \in \mathbb{R}^n$,
- *labels* $y^{(i)} \in \mathbb{R}$

for $i = 1, \ldots, m$.

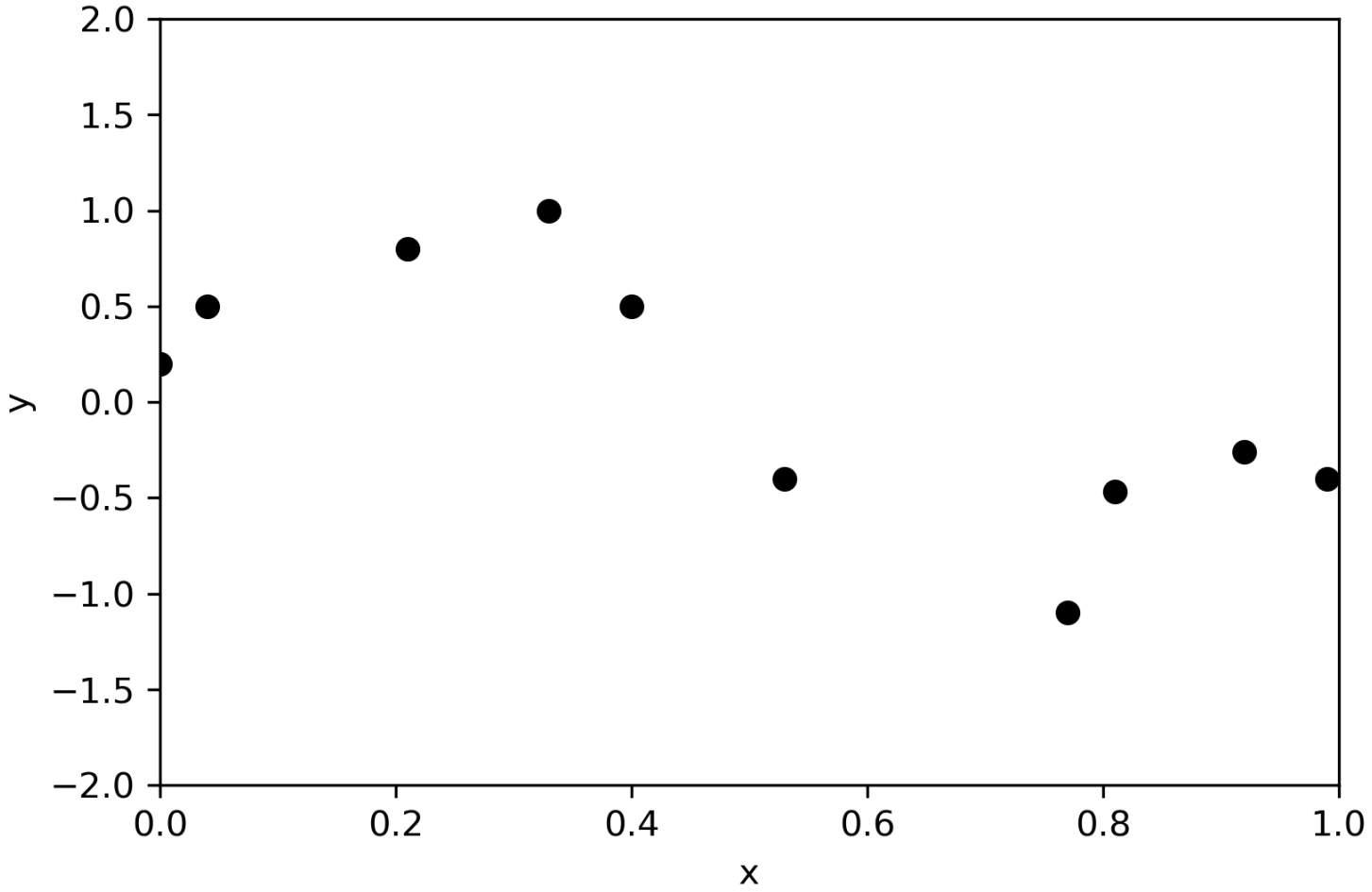
The goal is to construct a *function*

$$f(\mathbf{x}^{(i)}) \approx y^{(i)}$$

- Today, we focus on constructing f through optimisation.

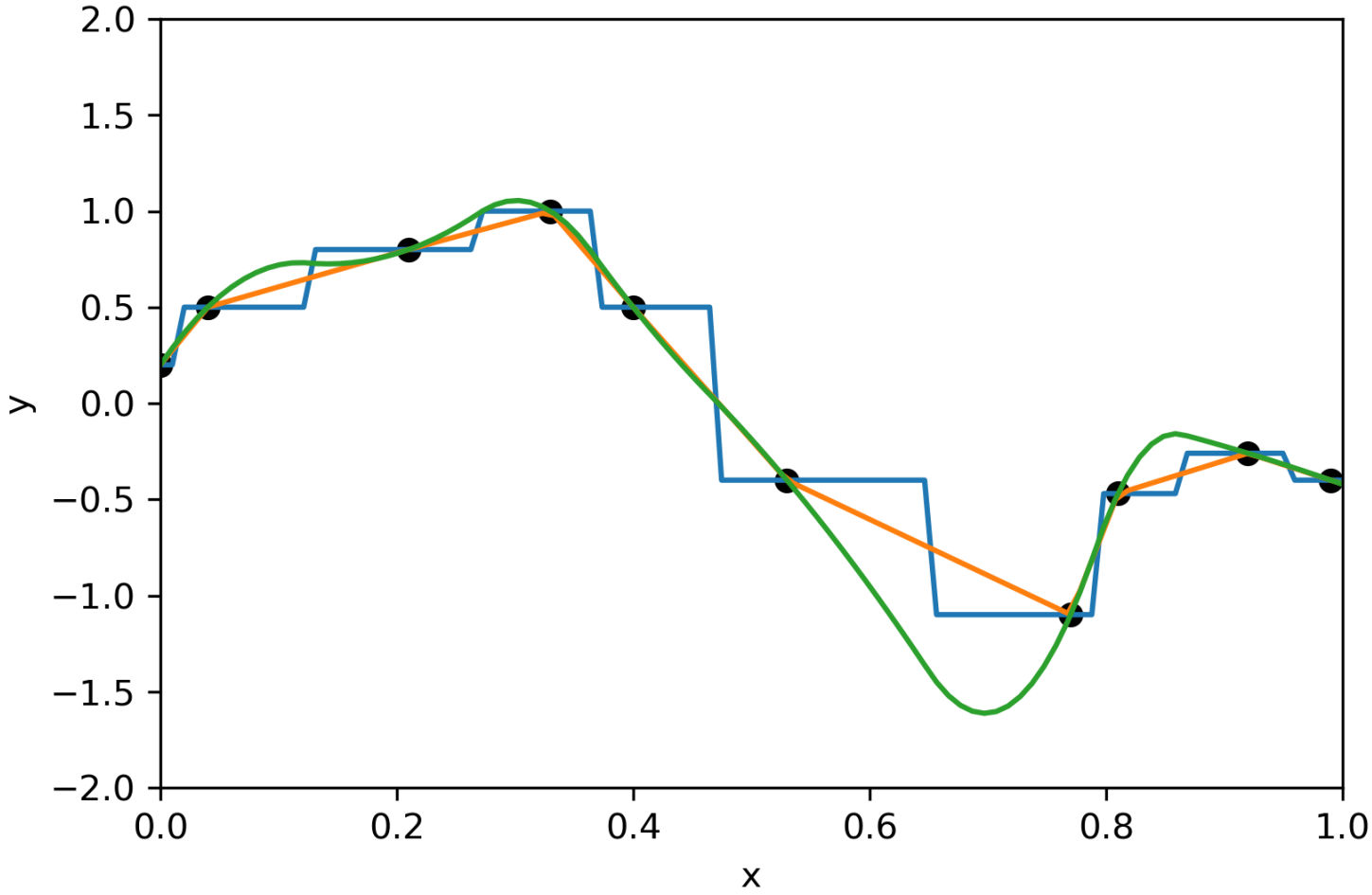
Universiteit Utrecht

Fit a line through a set of points



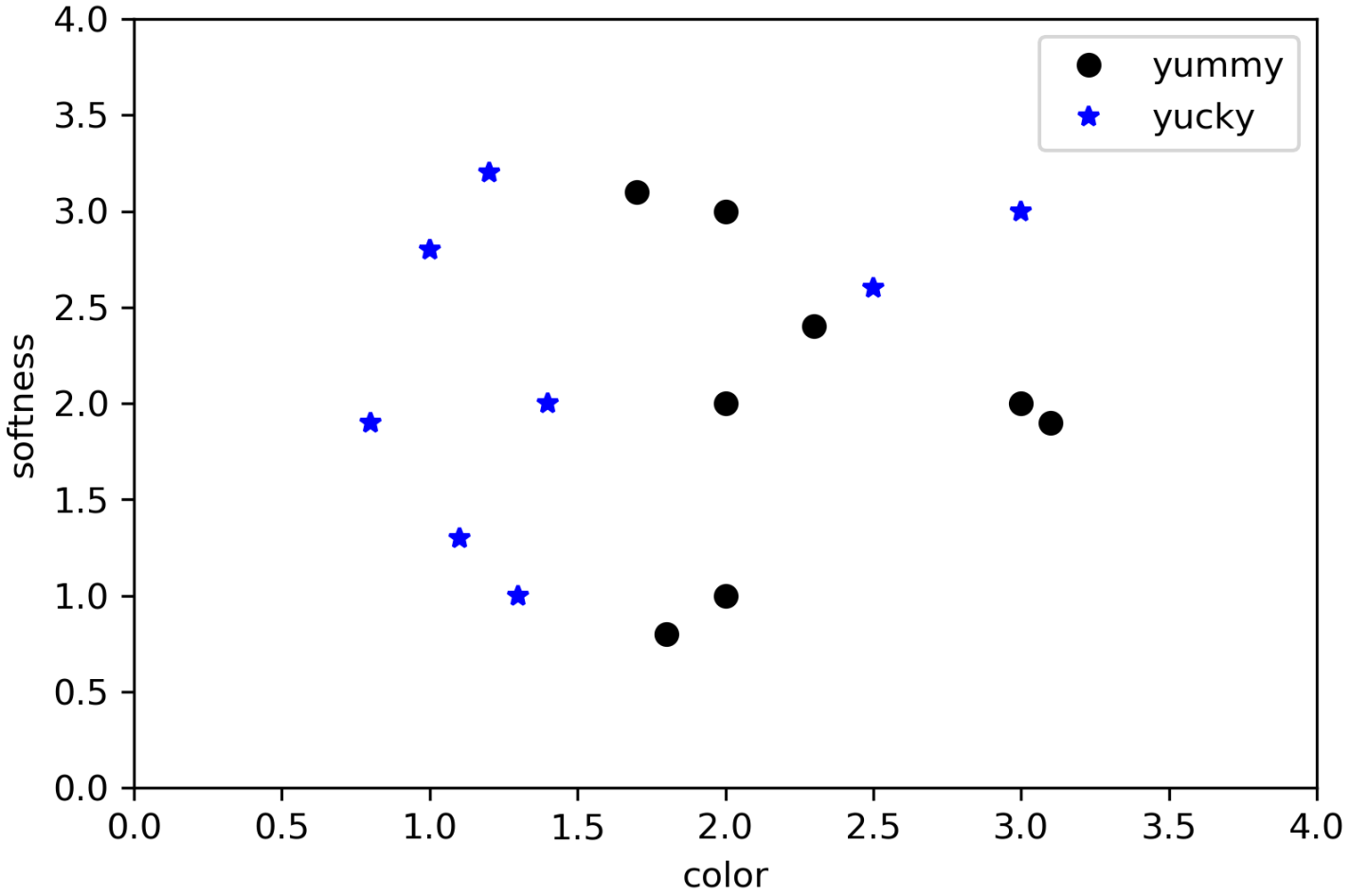
Universiteit Utrecht

Fit a line through a set of points



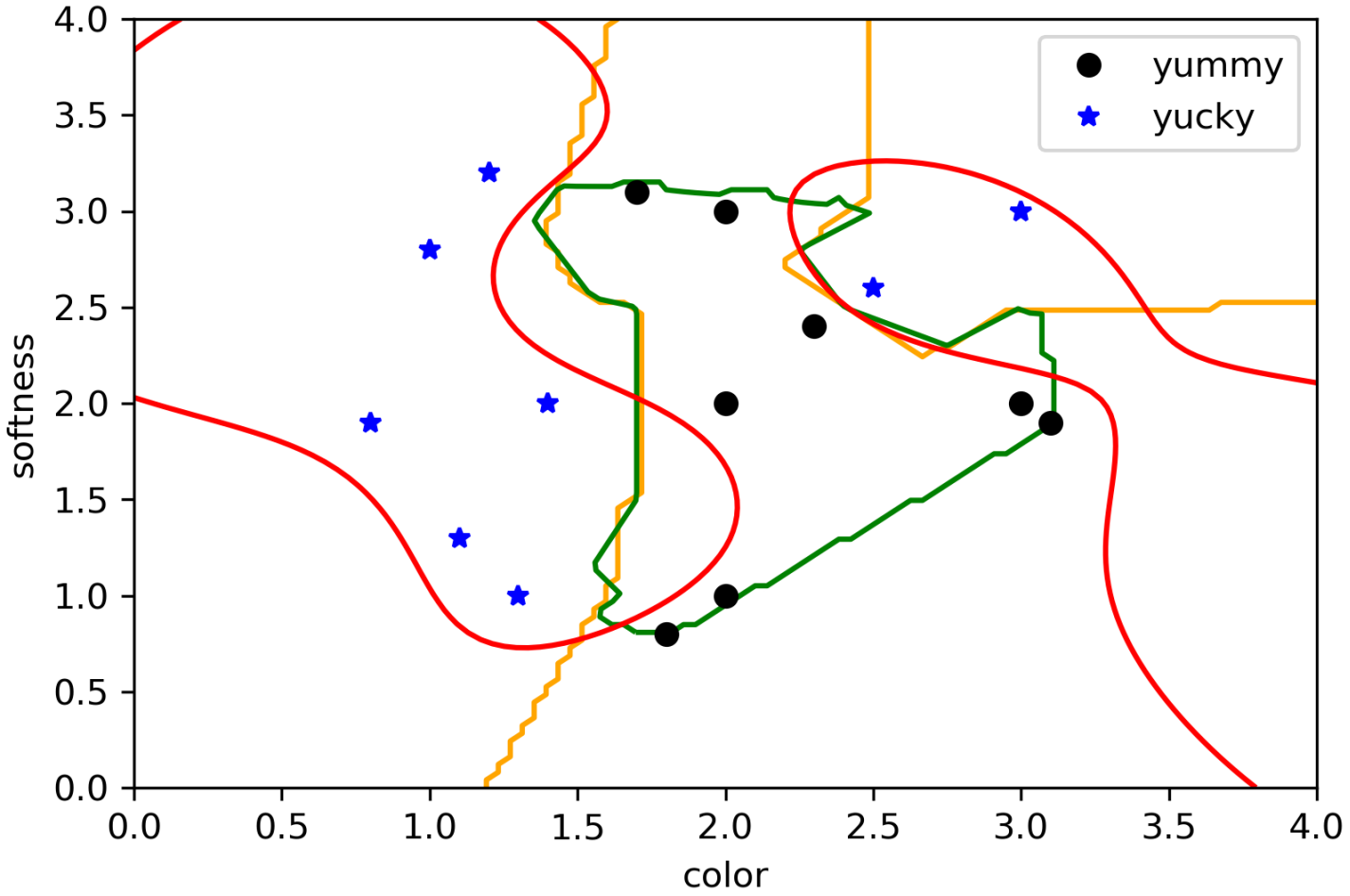
Universiteit Utrecht

Find out which papayas are yummy



Universiteit Utrecht

Find out which papayas are yummy



Universiteit Utrecht

Machine learning

Typical machine learning workflow consists of

- *preprocessing* of data
- *training* model on training data
- *testing* and *tuning* model on a test data set
- *validating* the final tuned model on a validation data set

Universiteit Utrecht

The supervised learning zoo

- Regression
- Neural networks
- Support Vector Machines
- Decision Trees
- ...

Universiteit Utrecht

Example: linear regression

$$f(\mathbf{x}) = b + \mathbf{a}^T \mathbf{x},$$

where $\mathbf{w} = (b, \mathbf{a}) \in \mathbb{R}^{n+1}$ is determined by solving

$$\min_{\mathbf{w}} \sum_{i=1}^{m} \left(f\left(\mathbf{x}^{(i)}\right) - y^{(i)} \right)^2.$$

Universiteit Utrecht

Example: binary classification

$$f(\mathbf{x}) = \sigma\left(b + \mathbf{a}^T\mathbf{x}\right),$$

where $\sigma(\cdot)$ is the *activation function* (e.g., $\sigma(y) = \text{sign}(y)$) and $\mathbf{w}$ is determined by

$$\min_{\mathbf{w}} \sum_{i=1}^{m} \left(1 - y^{(i)} f\left(\mathbf{x}^{(i)}\right)\right)^2.$$

Universiteit Utrecht

Example: Neural Network

$$f(\mathbf{x}) = f_k \circ f_{k-1} \circ \ldots \circ f_1(\mathbf{x}),$$

with

$$f_i(\mathbf{x}) = \sigma\left(A_i\mathbf{x} + \mathbf{b}_i\right).$$

- The network architecture is determined by the size and structure of the matrices A_i
- The weights A_i and $\mathbf{b}_i$ are determined by *training*:

$$\min_{\mathbf{w}} \sum_{i=1}^{m} \ell\left(f\left(\mathbf{x}^{(i)}\right), y^{(i)}\right).$$

Universiteit Utrecht

Example: Support vector machines

$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i k(\mathbf{x}, \mathbf{x}^{(i)}),$$

and the weights are determined by solving

$$\min_{\mathbf{w}} \ell\left(f\left(\mathbf{x}^{(i)}\right), y^{(i)}\right) + \mathbf{w}^T K \mathbf{w},$$

with $k_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$

Universiteit Utrecht

Optimisation

Most supervised learning methods lead to a structured optimisation problem:

$$\min_{\mathbf{w}} \sum_{i=1}^{m} \ell\left(f\left(\mathbf{x}^{(i)}\right), y^{(i)}\right) + r(\mathbf{w}),$$

- $\mathbf{w}$ parametrizes the function
- ℓ measures the difference between prediction and label for the i^{th} sample
- r is a regularisation term

Universiteit Utrecht

Optimisation

Most supervised learning methods lead to a structured optimisation problem:

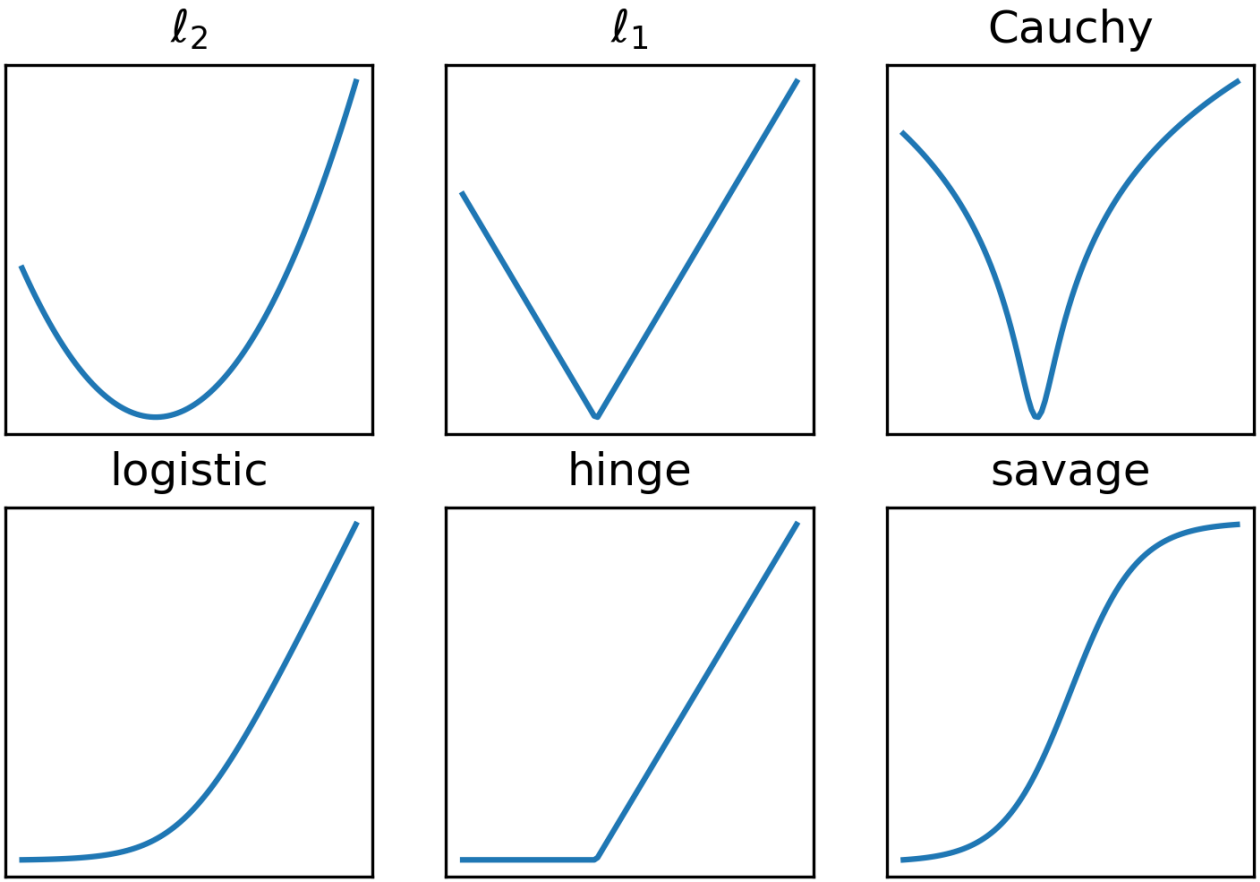
$$\min_{\mathbf{w}} \sum_{i=1}^{m} \ell\left(f\left(\mathbf{x}^{(i)}\right), y^{(i)}\right) + r(\mathbf{w}),$$

- $\mathbf{w}$ parametrizes the function
- ℓ measures the difference between prediction and label for the i^{th} sample
- r is a regularisation term

- Evaluation of the cost function may be computationally expensive
- Computing the gradient exactly may be difficult

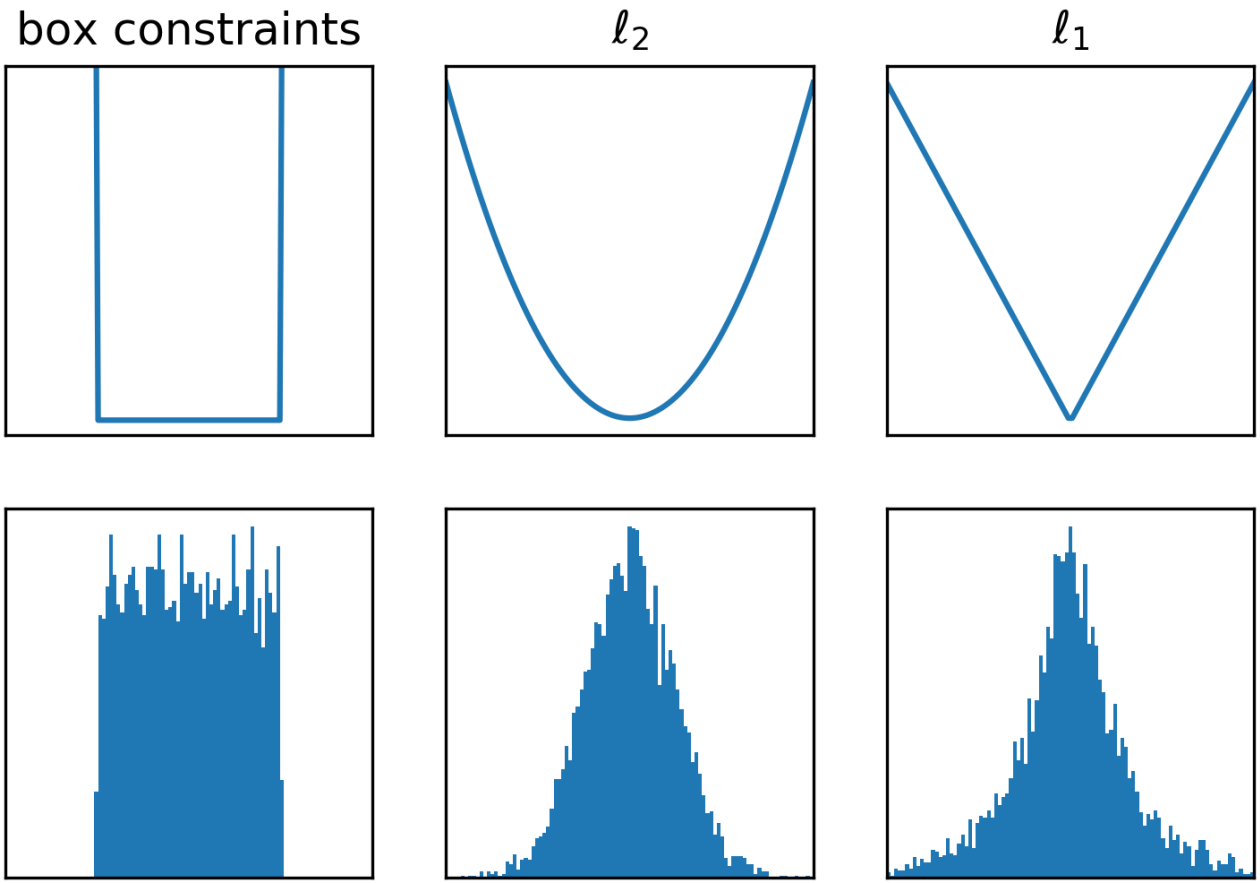
Universiteit Utrecht

The loss function



Universiteit Utrecht

The regulariser



Structure of the cost function

$$c(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} c_i(\mathbf{w}).$$

- smoothness and convexity of c are important properties when designing optimisation algotithms.

Universiteit Utrecht

Structure of the cost function

$$c(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} c_i(\mathbf{w}).$$

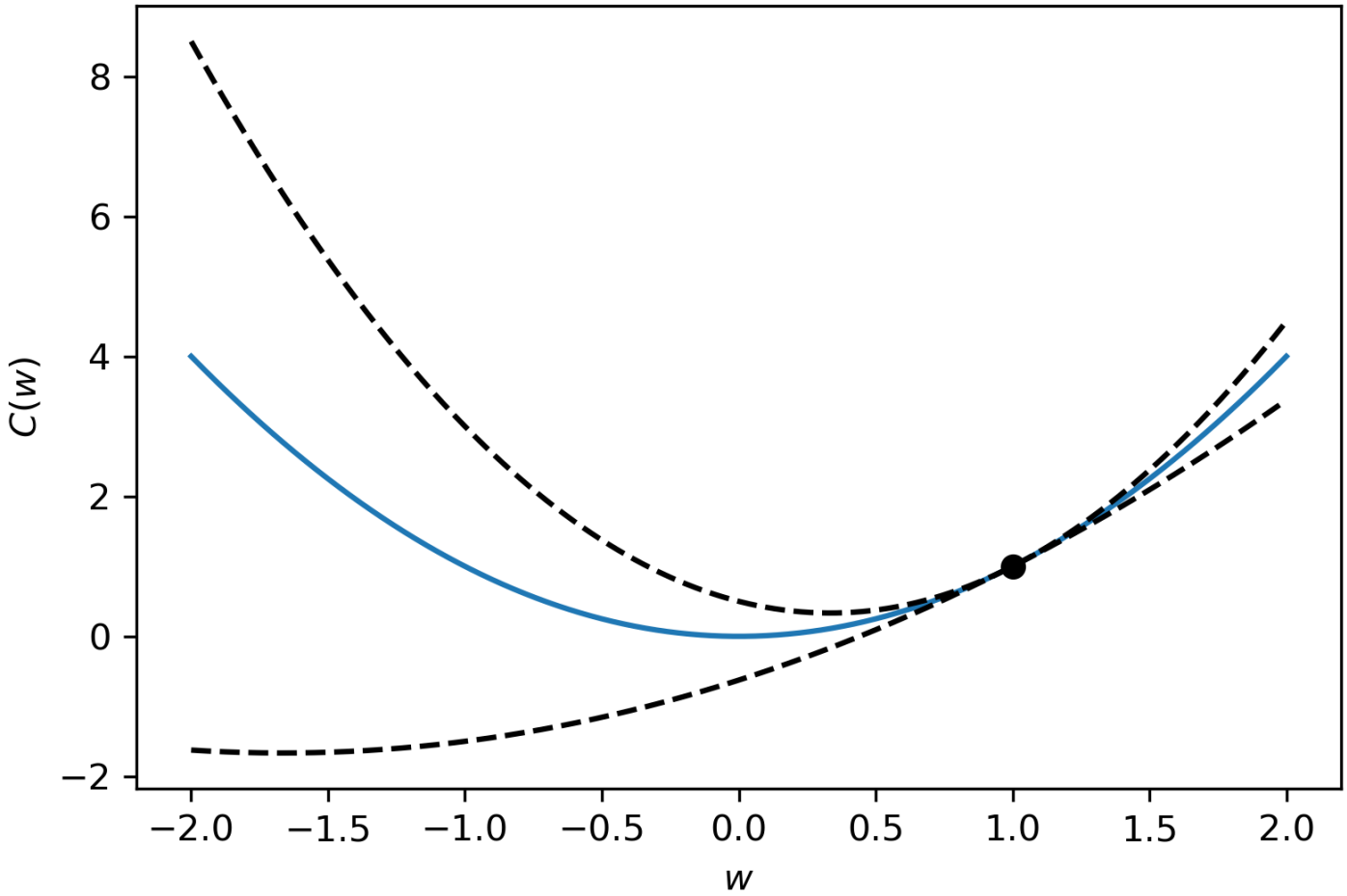
- smoothness and convexity of c are important properties when designing optimisation algotithms.

We assume that c is

- Smooth ($\|\nabla c(\mathbf{w}) - \nabla c(\mathbf{w}')\| \leq L\|\mathbf{w} - \mathbf{w}'\|$)
- Strongly convex ($\|\nabla c(\mathbf{w}) - \nabla c(\mathbf{w}')\| \geq \mu\|\mathbf{w} - \mathbf{w}'\|$)

Universiteit Utrecht

Smooth, strongly convex functions



Universiteit Utrecht

Practical aspects

- Many cost functions in ML are only *locally* convex
- We can make any convex function strongly convex by adding $\beta\|\mathbf{w}\|^2$
- We can make any convex function smooth by computing the *Moreau envelope*

Universiteit Utrecht

A Prototype algorithm

Universiteit Utrecht

Simplified setting

- Assume c is strongly convex and smooth:

$$\mu\|\mathbf{w} - \mathbf{w}^*\|^2 \leq c(\mathbf{w}) - c(\mathbf{w}^*) \leq L\|\mathbf{w} - \mathbf{w}^*\|^2.$$

- Computational cost of evaluating c is linear in m
- Goal is to find a minimizer $\mathbf{w}_k$ for which $|c(\mathbf{w}_k) - c(\mathbf{w}_*)| \leq \epsilon$.

Universiteit Utrecht

Simplified setting

- Assume c is strongly convex and smooth:

$$\mu\|\mathbf{w} - \mathbf{w}^*\|^2 \leq c(\mathbf{w}) - c(\mathbf{w}^*) \leq L\|\mathbf{w} - \mathbf{w}^*\|^2.$$

- Computational cost of evaluating c is linear in m
- Goal is to find a minimizer $\mathbf{w}_k$ for which $|c(\mathbf{w}_k) - c(\mathbf{w}_*)| \leq \epsilon$.

What is the computational complexity in terms of m and ϵ ?

Universiteit Utrecht

Gradient descent

The basic iteration

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \nabla c\left(\mathbf{w}_k\right),$$

- requires m evaluations at each iteration
- converges to a minimum at a *linear* rate ($\mathcal{O}(\rho^k)$)

Universiteit Utrecht

Convergence

In terms of the values we have

$$c(\mathbf{w}_{k+1}) - c(\mathbf{w}_*) \leq \left(1 - 2\alpha_k\mu + \alpha_k^2\mu L\right)\left(c(\mathbf{w}_k) - c(\mathbf{w}_*)\right).$$

- Convergence ensured when $0 < \alpha_k < 2/L$.
- Convergence may be arbitrarily slow when c is ill-conditioned ($\mu \ll L$)

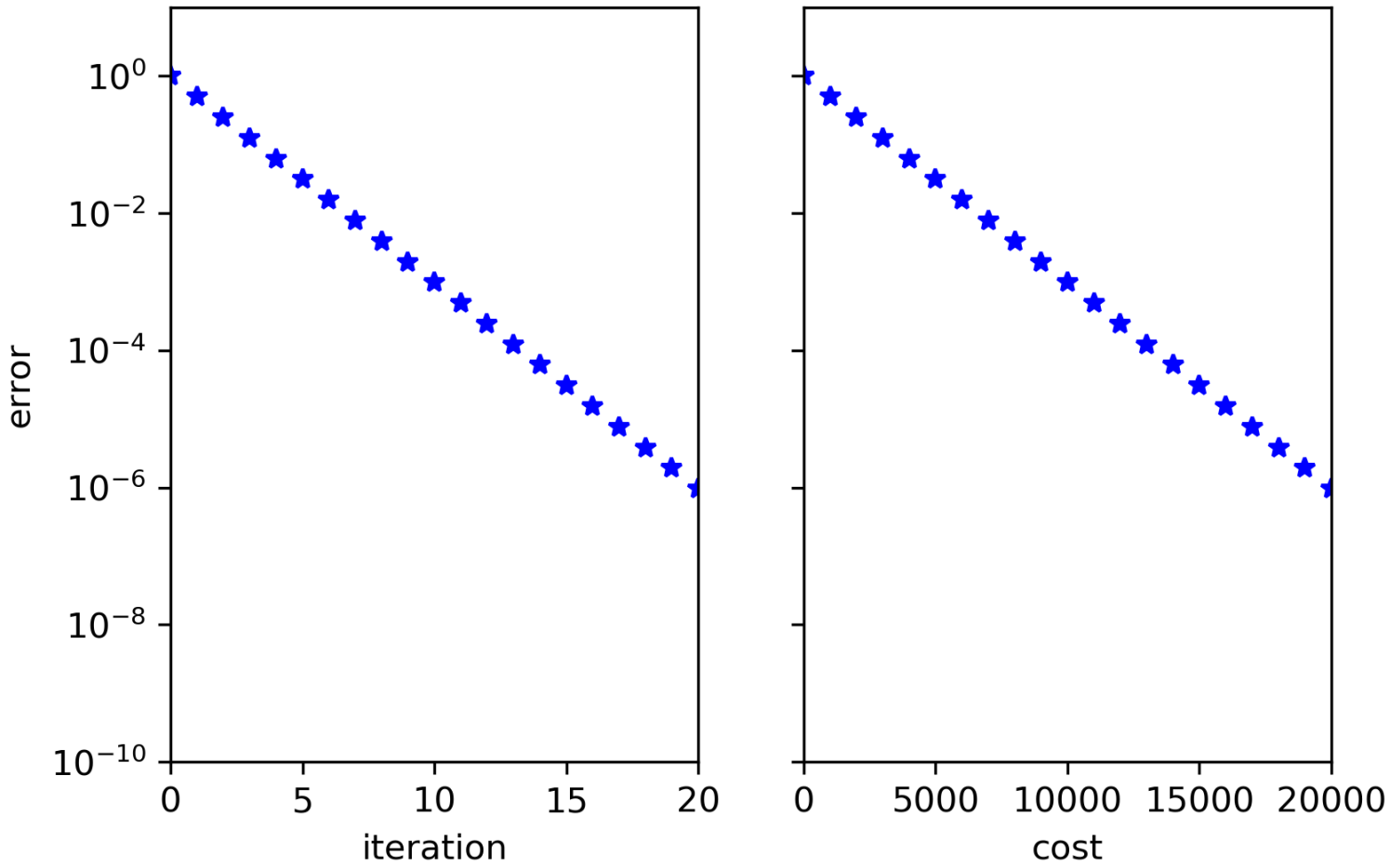
Universiteit Utrecht

Computational complexity

- Assuming a linear rate of convergence we need $\mathcal{O}(\log \epsilon^{-1})$ iterations
- The overall computational cost is linear in the sample size: $\mathcal{O}(m \cdot \log \epsilon^{-1})$

Universiteit Utrecht

Computational complexity



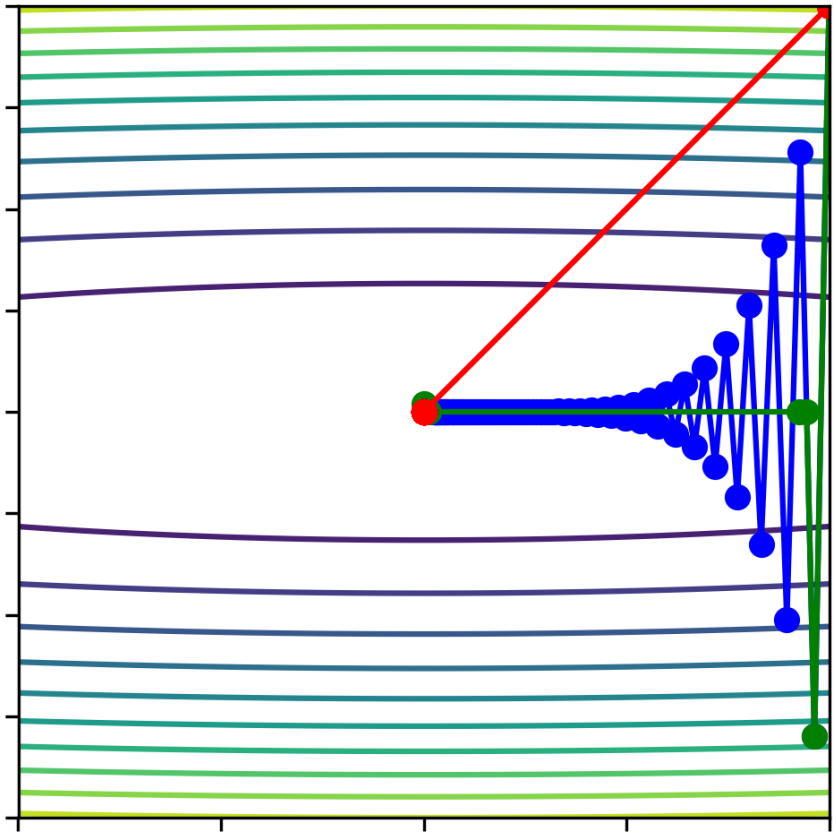
Universiteit Utrecht

Practical aspects

- We don't know L, μ; they need to be estimated
- Performance (*constants*) can be improved by adaptive stepsize (linesearch)
- Convergence *rate* can be improved by using (Quasi)-Newton methods

Universiteit Utrecht

Faster convergence



Universiteit Utrecht

Harnessing inexactness

Universiteit Utrecht

Sample average approximation

Approximate the cost function

$$c(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} c_i(\mathbf{w}) \approx \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} c_i(\mathbf{w}),$$

with $\mathcal{I} \subseteq \{1, 2, \ldots, m\}$.

Universiteit Utrecht

Sample average approximation

Approximate the cost function

$$c(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} c_i(\mathbf{w}) \approx \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} c_i(\mathbf{w}),$$

with $\mathcal{I} \subseteq \{1, 2, \ldots, m\}$.

- Use same the approximation for the gradient

$$\nabla c(\mathbf{w}) = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} \nabla c_i(\mathbf{w}) + \mathbf{e}$$

- Cost per iteration is proportional to $m' = |\mathcal{I}|$

Universiteit Utrecht

Sampling schemes

An (arbitrary) *deterministic* approximation yields

- $\|\mathbf{e}\|^2 = \mathcal{O}\left(\left(\frac{m-m'}{m}\right)^2\right)$

Universiteit Utrecht

Sampling schemes

An (arbitrary) *deterministic* approximation yields

- $\|\mathbf{e}\|^2 = \mathcal{O}\left(\left(\frac{m-m'}{m}\right)^2\right)$

Sampling uniformly at random with replacement yields an *unbiased* approximation with

- $\mathbb{E}(\|\mathbf{e}\|^2) = \mathcal{O}\left(\frac{1}{m'}\right),$

Universiteit Utrecht

Sampling schemes

An (arbitrary) *deterministic* approximation yields

- $\|\mathbf{e}\|^2 = \mathcal{O}\left(\left(\frac{m-m'}{m}\right)^2\right)$

Sampling uniformly at random with replacement yields an *unbiased* approximation with

- $\mathbb{E}(\|\mathbf{e}\|^2) = \mathcal{O}\left(\frac{1}{m'}\right)$,

Sampling uniformly at random *without replacement* yields a *biased* approximation with

- $\mathbb{E}(\|\mathbf{e}\|^2) = \mathcal{O}\left(\frac{m-m'}{mm'}\right)$.

Universiteit Utrecht

Gradient descent with errors

Use a modified iteration instead

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \mathbf{g}_k,$$

where

$$\mathbf{g}_k = \nabla c\left(\mathbf{w}_k\right) + \mathbf{e}_k.$$

Universiteit Utrecht

Convergence with errors

In terms of the values we have

$$c(\mathbf{w}_{k+1}) - c(\mathbf{w}_*) \leq \rho_k \left(c(\mathbf{w}_k) - c(\mathbf{w}_*)\right) + \lambda_k \nabla c\left(\mathbf{w}_k\right)^T \mathbf{e}_k + \frac{\alpha_k^2 L}{2} \|\mathbf{e}_k\|^2,$$

with

- $\rho_k = 1 - 2\alpha_k \mu + \alpha_k^2 \mu L$
- $\lambda_k = \alpha_k(\alpha_k L - 1)$

Universiteit Utrecht

Convergence with errors

In terms of the values we have

$$c(\mathbf{w}_{k+1}) - c(\mathbf{w}_*) \leq \rho_k \left(c(\mathbf{w}_k) - c(\mathbf{w}_*)\right) + \lambda_k \nabla c\left(\mathbf{w}_k\right)^T \mathbf{e}_k + \frac{\alpha_k^2 L}{2} \|\mathbf{e}_k\|^2,$$

with

- $\rho_k = 1 - 2\alpha_k \mu + \alpha_k^2 \mu L$
- $\lambda_k = \alpha_k(\alpha_k L - 1)$

Can we ensure convergence?

Universiteit Utrecht

Convergence and complexity

- We can control the error by increasing the samplesize or decreasing the stepsize
- We expect a tradeoff between accuracy, complexity and speed of convergence

Universiteit Utrecht

Convergence and complexity - deterministic

With fixed step size $\alpha_k \equiv 1/L$ we have

$$c(\mathbf{w}_{k+1}) - c(\mathbf{w}_*) \leq \rho\left(c(\mathbf{w}_k) - c(\mathbf{w}_*)\right) + \frac{1}{2L}\left(\frac{m - m'_k}{m}\right)^2.$$

Universiteit Utrecht

Convergence and complexity - deterministic

With fixed step size $\alpha_k \equiv 1/L$ we have

$$c(\mathbf{w}_{k+1}) - c(\mathbf{w}_*) \leq \rho\,(c(\mathbf{w}_k) - c(\mathbf{w}_*)) + \frac{1}{2L}\left(\frac{m - m'_k}{m}\right)^2.$$

- linear convergence with increasing samplesize $m - m'_k = \mathcal{O}(\gamma^{k/2})$
- overall complexity is *not* asymptotically better; $\mathcal{O}(m \cdot \log \epsilon^{-1})$

Universiteit Utrecht

Convergence and complexity - stochastic

With fixed sample size $m'_k \equiv 1$ we have

$$\mathbb{E}\left(c(\mathbf{w}_{k+1}) - c(\mathbf{w}_*)\right) \leq \rho_k \mathbb{E}\left(c(\mathbf{w}_k) - c(\mathbf{w}_*)\right) + \frac{\alpha_k^2 L}{2} \mathbb{E}\|\mathbf{e}_k\|^2.$$

- sublinear convergence with diminishing stepsize $\alpha_k = \mathcal{O}(1/k)$
- overall complexity is $\mathcal{O}(\epsilon^{-1})$ (independent of m)

Universiteit Utrecht

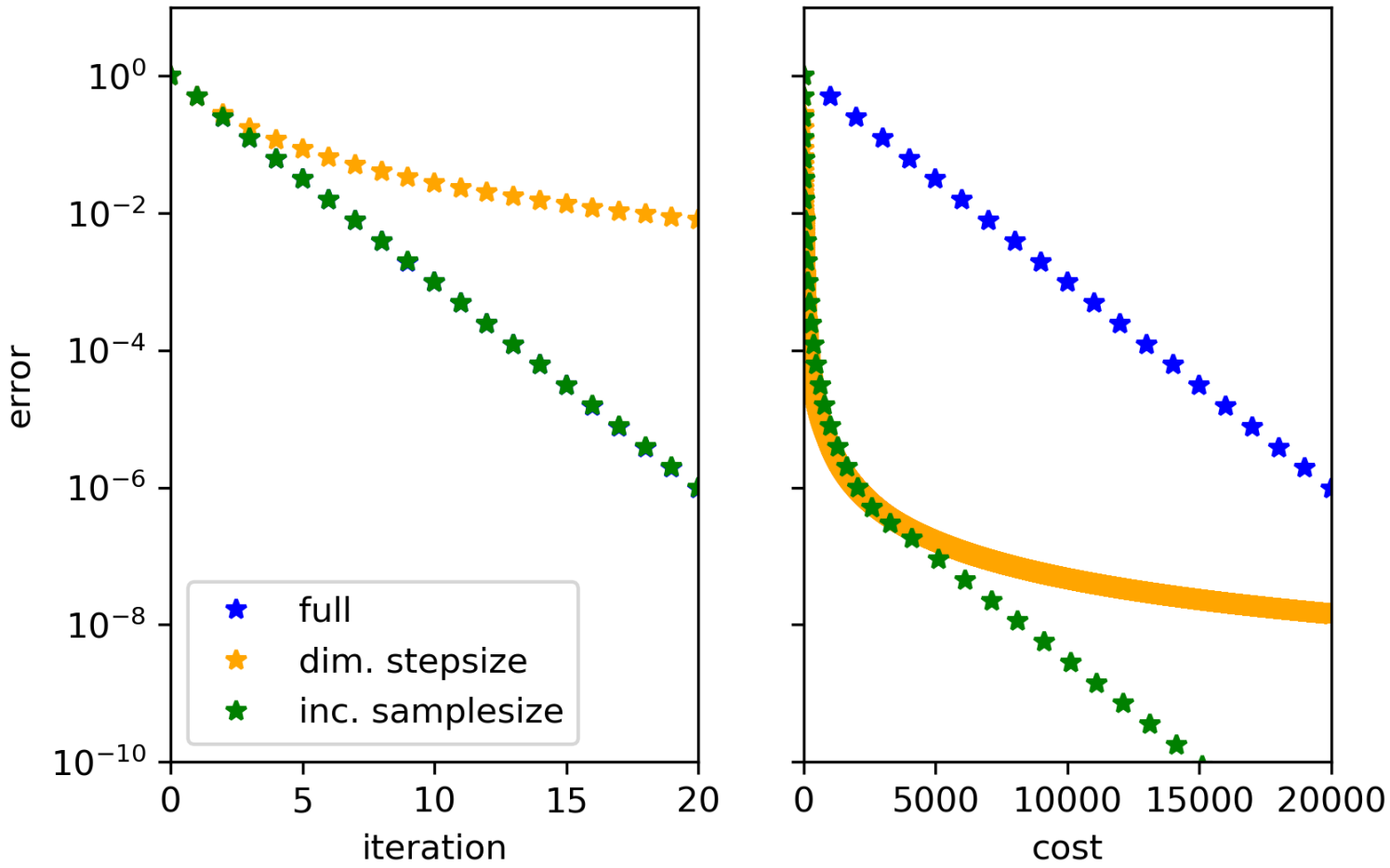
Convergence and complexity - hybrid

With increasing sample size and fixed step size we have

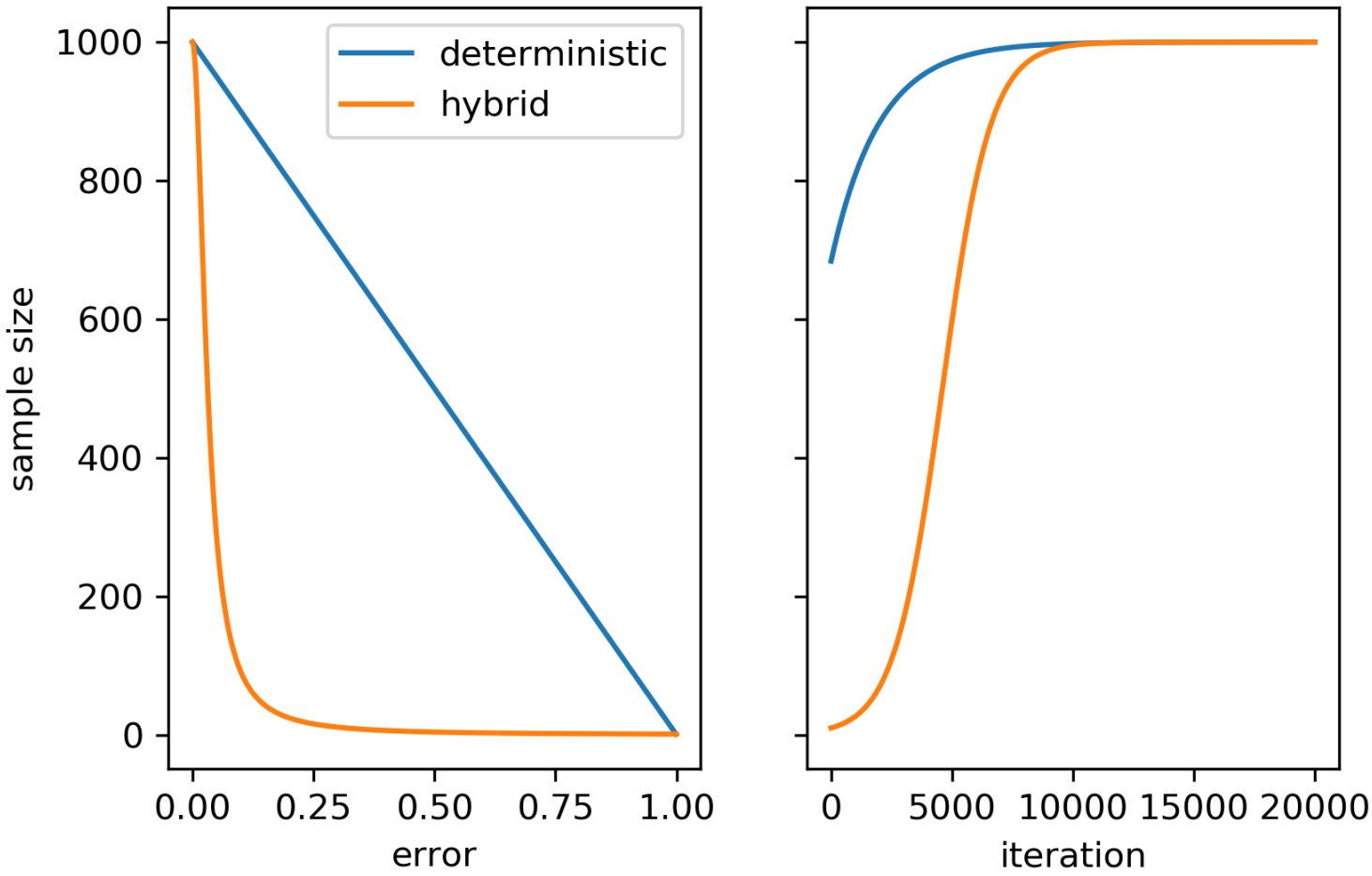
- Linear convergence with increasing sample size $(m - m'_k)/m'_k = \mathcal{O}(\gamma^k)$
- Overall complexity *is* asymptotically better; $\mathcal{O}(\frac{m \cdot \log(\epsilon^{-1})}{1+\epsilon \cdot m})$

Universiteit Utrecht

Convergence - example



Complexity - example



Universiteit Utrecht

Practical aspects

- How fast should we increase the sample size
- Adaptive step sizes
- Second order information

Universiteit Utrecht

Wrap-up

Universiteit Utrecht

Take home message

- Supervised learning give rise to structured optimisation problems
- Inexactness is a powerful tool to speed up computations
- Careful analysis is needed to guarantee convergence
- Many practical issues remain to be resolved

Universiteit Utrecht

Further reading

Friedlander, M. P., & Schmidt, M. (2012). Hybrid Deterministic-Stochastic Methods for Data Fitting. SIAM Journal on Scientific Computing, 34(3), A1380–A1405. https://doi.org/10.1137/110830629

Curtis, F. E., & Nocedal, J. (2018). Optimization Methods for Large-Scale Machine Learning. SIAM Review, 60(2), 223–311. https://doi.org/10.1137/16M1080173