Machine Learning for Networking
ML4N

Luca Vassio
Gabriele Ciravegna
Zhihao Wang
Tailai Song

Recap

- Data: set of data points (**x**,y)
- Model: set $\mathcal{H}$ of hypothesis maps h(.)
- Loss: quality measure L((**x**,y),h)
- Learn hypothesis that incurs **minimum average loss** when predicting labels of **training datapoints** based on their features (**ERM**)
- If parametrized model $\widehat{\mathbf{w}} = \underset{\mathbf{w} \in \mathbb{R}^n}{\operatorname{argmin}} f(\mathbf{w})$

Recap

- Model validation
- Model selection
- Metrics for performance evaluation
- Hyper-parameter tuning
- ML process

What is missing?

- How to learn?
- How to find the best parameters **w**

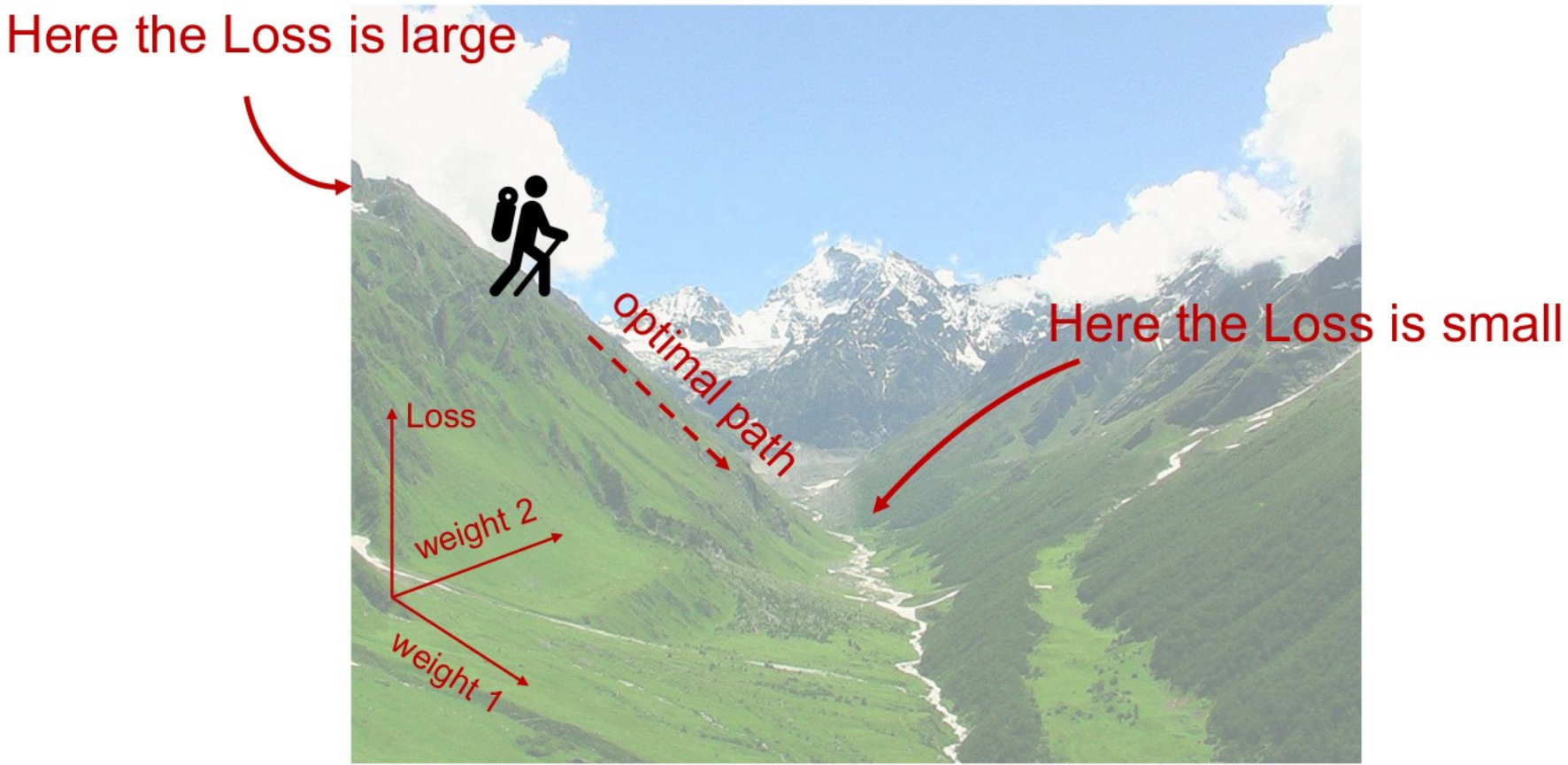
Gradient based learning

ERM on parametrized models

$$\min_{\mathbf{w}\in\mathbb{R}^n} f(\mathbf{w}) := (1/m)\sum_{i=1}^{m} \underbrace{L\left((\mathbf{x}^{(i)}, y^{(i)}), h^{(\mathbf{w})}\right)}_{f_i(\mathbf{w})}.$$

- If each loss $f_i(\mathbf{w})$ is differentiable, $f(\mathbf{w})$ is also differentiable
- e.g., squared error loss, logistic loss

Minimizing the loss



Gradient based method

- Gradient based methods are iterative
- They construct a sequence of parameter vectors $\mathbf{w}^{(0)} \rightarrow \mathbf{w}^{(1)} \rightarrow \mathbf{w}^{(2)} \ldots$
- Hopefully the sequence converge to a minimizer of f(**w**)

$$f(\overline{\mathbf{w}}) = \bar{f} := \min_{\mathbf{w} \in \mathbb{R}^n} f(\mathbf{w}).$$

Gradient based method

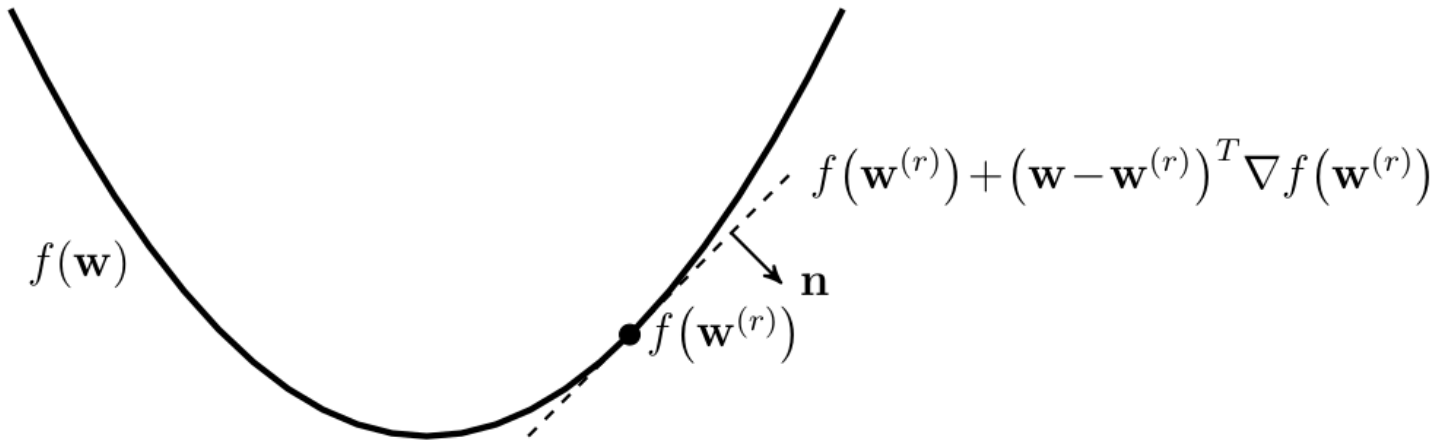


Figure 5.1: A differentiable function $f(\mathbf{w})$ can be approximated locally around a point $\mathbf{w}^{(r)}$ using a hyperplane whose normal vector $\mathbf{n} = (\nabla f\left(\mathbf{w}^{(r)}\right), -1)$ is determined by the gradient $\nabla f\left(\mathbf{w}^{(r)}\right)$ [119].

Gradient based method

- Approximate locally f(**w**) around **w**$^{(r)}$

$$f(\mathbf{w}) \approx f\left(\mathbf{w}^{(r)}\right) + \left(\mathbf{w} - \mathbf{w}^{(r)}\right)^T \nabla f\left(\mathbf{w}^{(r)}\right) \text{ for } \mathbf{w} \text{ sufficiently close to } \mathbf{w}^{(r)}.$$

- Since we want to **minimize** f, $\left(\mathbf{w} - \mathbf{w}^{(r)}\right)^T \nabla f\left(\mathbf{w}^{(r)}\right)$ should be **negative**
- If the gradient is positive, decrease **w**
- If the gradient is negative, enlarge **w**

Gradient descent step

- Given current guess $\mathbf{w}^{(r)}$, the next guess is:

$$\mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - \alpha \nabla f(\mathbf{w}^{(r)}) \qquad \text{with a sufficiently small step size } \alpha > 0$$

- Gradient descent step
- α is called learning rate

Gradient descent step

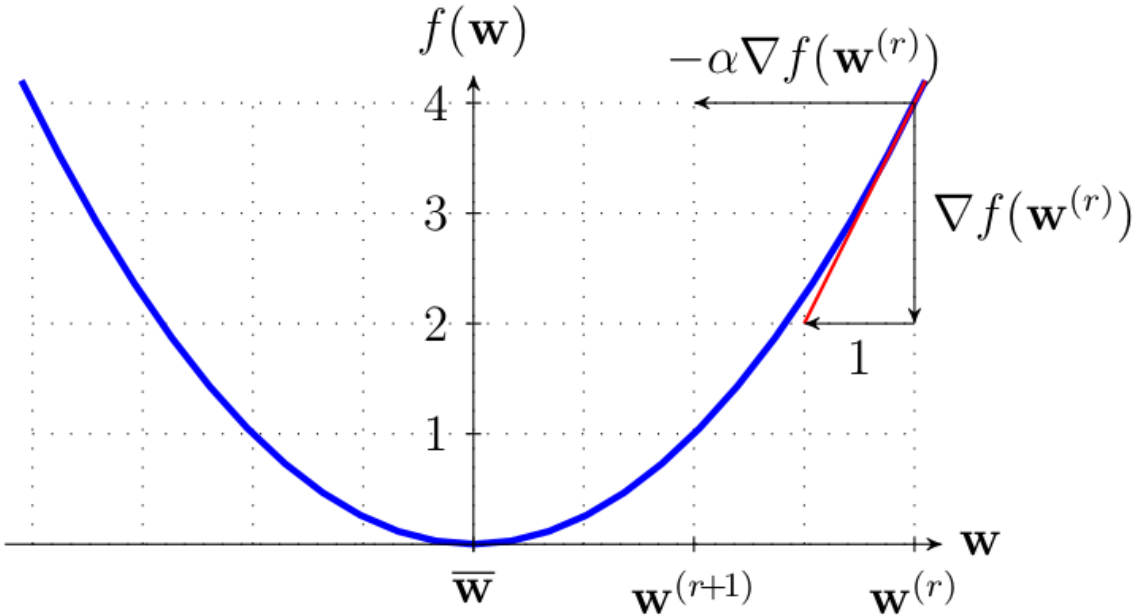


Figure 5.2: A GD step (5.6) updates a current guess or approximation $\mathbf{w}^{(r)}$ for the optimal parameter vector $\overline{\mathbf{w}}$ (5.3) by adding the correction term $-\alpha\nabla f(\mathbf{w}^{(r)})$. The updated parameter vector $\mathbf{w}^{(r+1)}$ is (typically) an improved approximation of the minimizer $\overline{\mathbf{w}}$.

Example: GD for linear regression

- ERM for linear regression and squared error loss

$$\overline{\mathbf{w}} = \underset{\mathbf{w}\in\mathbb{R}^n}{\operatorname{argmin}} f(\mathbf{w}) \text{ with } f(\mathbf{w}) = (1/m)\sum_{i=1}^{m}\left(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)}\right)^2.$$

Example: GD for linear regression

- ERM for linear regression and squared error loss

$$\overline{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} f(\mathbf{w}) \text{ with } f(\mathbf{w}) = (1/m)\sum_{i=1}^{m}\left(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)}\right)^2.$$

- Compute gradient

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

- GD step

$$\mathbf{w}^{(r)} := \mathbf{w}^{(r-1)} + \alpha(2/m)\sum_{i=1}^{m}\left(y^{(i)} - \left(\mathbf{w}^{(r-1)}\right)^T\mathbf{x}^{(i)}\right)\mathbf{x}^{(i)}.$$

Learning rate

Large step size

Small step size

Choosing learning rate

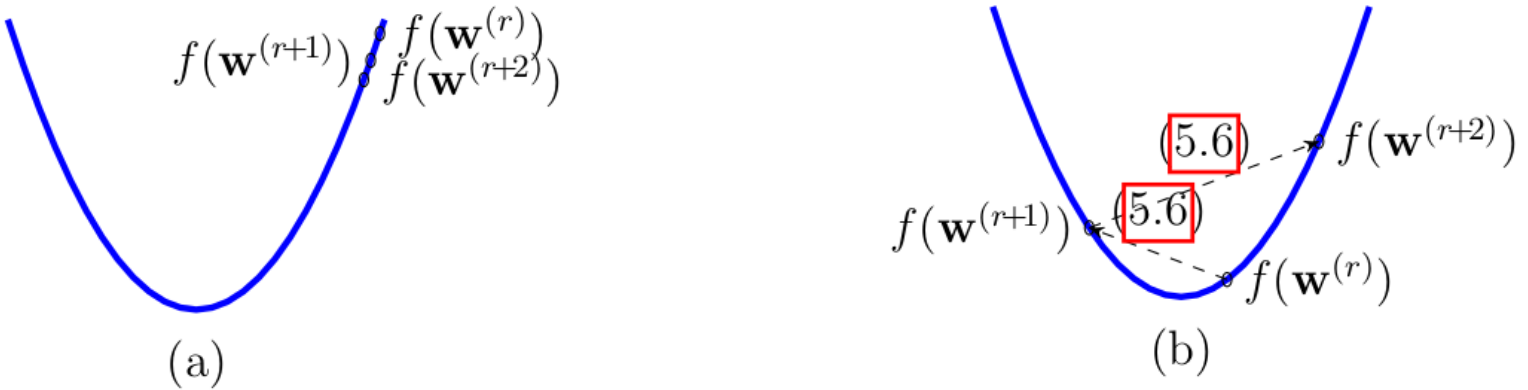


Figure 5.3: Effect of choosing bad values for the learning rate α in the GD step(5.6). (a) If the learning rate α in the GD step (5.6) is chosen too small, the iterations make very little progress towards the optimum or even fail to reach the optimum at all. (b) If the learning rate α is chosen too large, the iterates $\mathbf{w}^{(r)}$ might not converge at all (it might happen that $f(\mathbf{w}^{(r+1)}) > f(\mathbf{w}^{(r)})$!).

Choosing learning rate

- In some cases, theoretical bounds and optimal values can be given
- **Number of GD steps is smaller for standardized data**

When to stop?

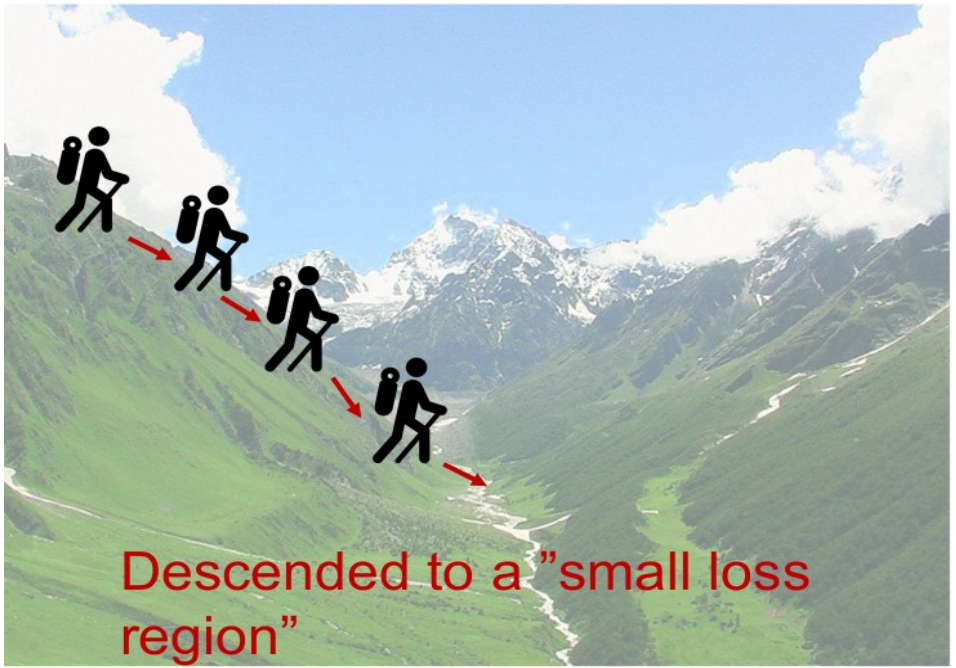
- Fixed number of iterations (epochs)
- When empirical risk is not improving anymore or not improving enough:

  $f(\mathbf{w}^{(r-1)}) - f(\mathbf{w}^{(r)})$ less than a threshold

- When validation error keep decreasing $\tilde{f}(\mathbf{w}^{(r)})$

Number of iterations (epochs)

N.o. epochs is large



N.o. epochs is smal

Stochastic gradient descent

- Gradient of empirical risk

$$\nabla f(\mathbf{w}) = (1/m) \sum_{i=1}^{m} \nabla f_i(\mathbf{w}) \text{ with } f_i(\mathbf{w}) := L\left((\mathbf{x}^{(i)}, y^{(i)}), h^{(\mathbf{w})}\right).$$

- The sum can be challenging
  - m can be billions
  - data can be stored in different places (or does not fit in memory)

Stochastic gradient descent

- Approximations of GD: $g(\mathbf{w}) \approx \nabla f(\mathbf{w})$

Stochastic gradient descent

- Approximations of GD: $g(\mathbf{w}) \approx \nabla f(\mathbf{w})$

  1. Iteratively replace sum with a (random) component (a single sample)

$$g(\mathbf{w}) := \nabla f_{\hat{i}}(\mathbf{w}). \qquad \mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - \alpha \nabla f_{\hat{i}_r}(\mathbf{w}^{(r)}),$$

  2. Iteratively replace sum with a (random) batch of samples

$$\mathcal{B} = \{i_1, \ldots, i_B\} \text{ (a "batch")} \qquad g(\mathbf{w}) = (1/B) \sum_{i' \in \mathcal{B}} \nabla f_{i'}(\mathbf{w})$$

Advanced gradient based techniques

- E.g., : Keep memory of gradient at previous iteration (improved local approximation)

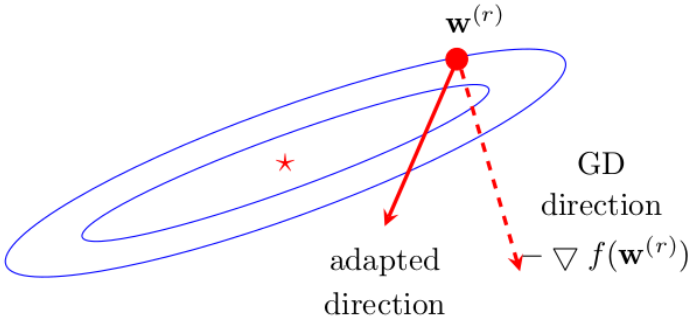


Figure 5.5: Advanced gradient-based methods use improved (non-linear) local approximations of the objective function $f(\mathbf{w})$ to "nudge" the update direction towards the optimal parameter vector $\overline{\mathbf{w}}$. The update direction of plain vanilla GD (5.6) is the negative gradient $-\nabla f\big(\mathbf{w}^{(r)}\big)$. For some objective functions the negative gradient might be only weakly correlated with the straight direction from $\mathbf{w}^{(r)}$ towards the optimal parameter vector $(\star)$.

Any questions?

Self-assessment quiz

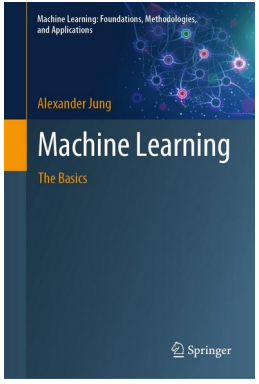
- Formulate a gradient-descent step for ERM in logistic regression
- Then perform a single step of it on this training set, starting from $\mathbf{w}^{(0)}=[0, 0, 0]^T$

Train data

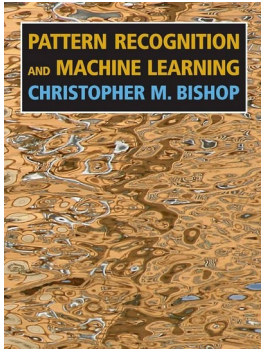
Feature 1	Feature 2	Feature 3	Label
10	5	1	-1
7	0	1	1
7	0	2	1

References: readings

- Chapters 5



- Chapter 3-5



Slide acknowledgments

- Alexander Jung – Aalto University