Machine Learning Basics

He He

New York University

September 6, 2020

Table of Contents

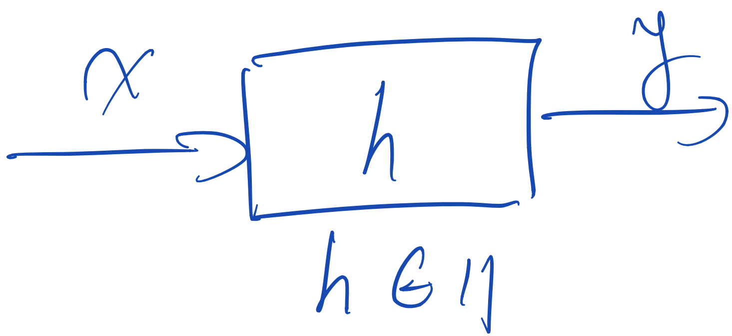
Rules vs data

Example: spam filter

- Rules

  Contains "Viagra"
  Contains "Rolex"
  Subject line is all caps
  ...

- Learning from data



Keys to success

- Availability of large amounts of (annotated) data

  Scraping, crowdsourcing, expert annotation

- Generalize to unseen samples

  Unknown data generating distribution: $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$

$(x, y) \sim D$

$\downarrow$ m samples

$\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$

training set

$\min \mathbb{E}_{D}[\text{error}(h)]$

test set

Empirical risk minimization (ERM)

Minimize the average loss on the training set over $\mathcal{H}$

$$\underbrace{\min_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \text{loss}(x^{(i)}, y^{(i)}, h)}_{\text{empirical risk}}$$

$$h(x^{(i)}) = y^{(i)}$$

Error decomposition

$$R(h) = E_D[\text{loss}(x, y, h)]$$

$$\hat{R}(h) = \frac{1}{m}\sum_{i=1}^{m} \text{loss}(x^{(i)}, y^{(i)}, h)$$

$$h^* = \min_h R(h) \quad \text{optimal}$$

$$h_H = \min_{h \in H} R(h) \quad \text{optimal in } H$$

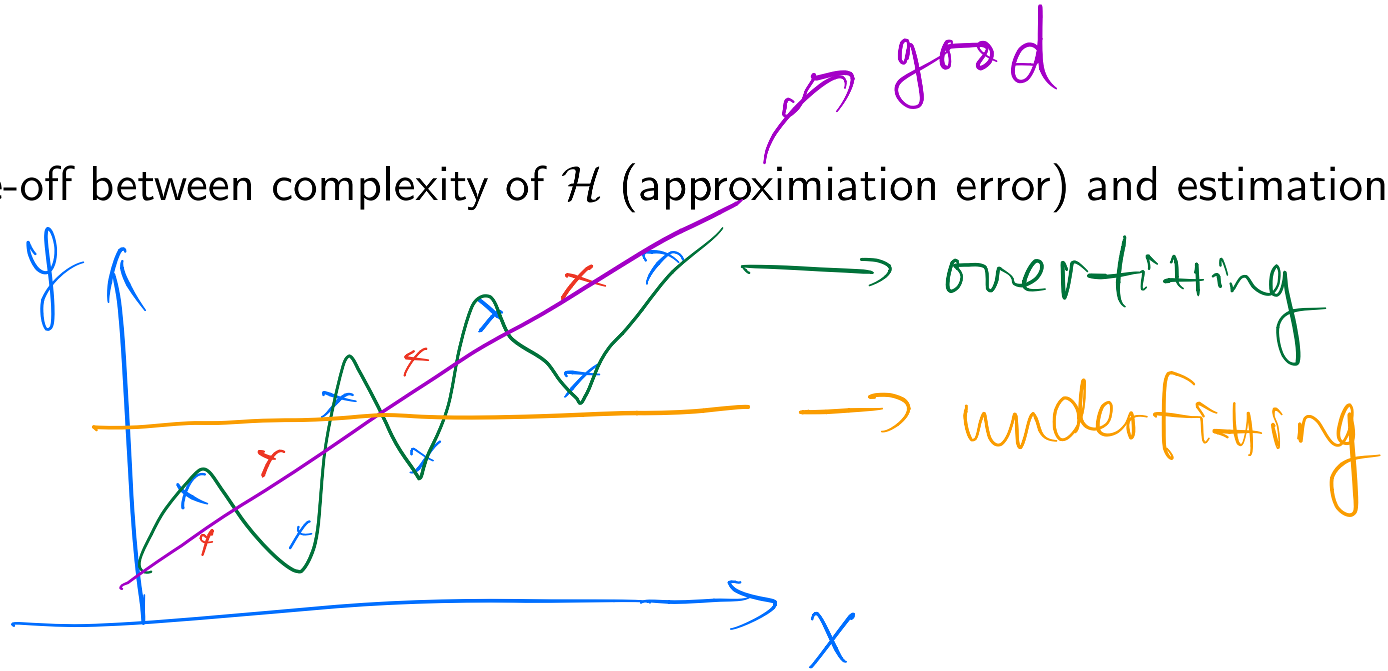
unk.

$$h_m = \min_{h \in H} \hat{R}(h) \quad \text{ERM sol.}$$

$$\underbrace{R(h_m) - R(h^*)}_{\text{excess risk}} = \underbrace{R(h_m) - R(h_H)}_{\text{estimation err.}} + \underbrace{R(h_H) - R(h^*)}_{\text{approximation err.}}$$

Overfitting vs underfitting

Trade-off between complexity of $\mathcal{H}$ (approximiation error) and estimation error



Question for us: how to choose a good $\mathcal{H}$ for certain domains

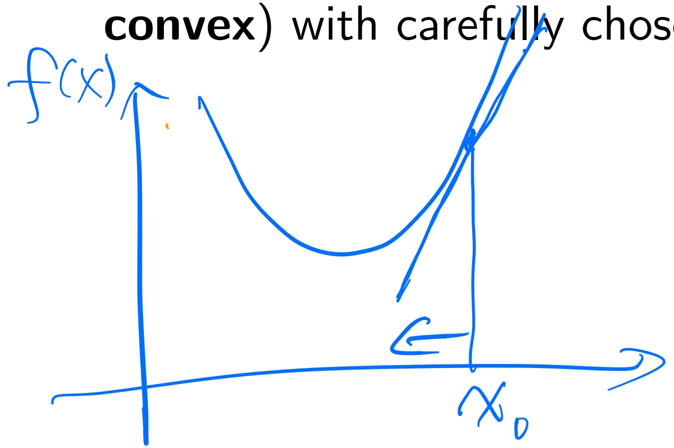
Table of Contents

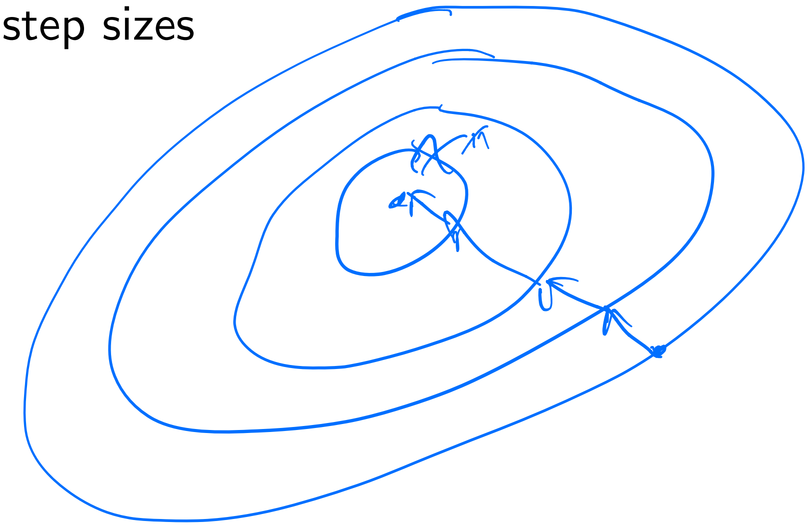
Overall picture

1. Obtain training data $D_{\text{train}} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$.
2. Choose a loss function L and a hypothesis class $\mathcal{H}$.
3. Learn a predictor by minimizing the empirical risk.

Gradient descent

- $w \leftarrow w - \eta \nabla_w F(w)$
- Converge to a local minimum (also global minimum if $F(w)$ is **convex**) with carefully chosen step sizes





$$x^t \leftarrow x^{t-1} - \alpha \nabla_x F(x)$$

step size

Stochastic gradient descent

- **Gradient descent (GD)**

$$w \leftarrow w - \eta \nabla_w \underbrace{\overbrace{\sum_{i=1}^{n} L(x^{(i)}, y^{(i)}, f_w)}^{\text{ERM obj.}}}_{\text{training loss}}$$

- **Stochastic gradient descent (SGD)**

$$\text{For each } (x, y) \in D_{\text{train}} :$$
$$w \leftarrow w - \eta \nabla_w \underbrace{L(x, y, f_w)}_{\text{example loss}}$$

GD vs SGD

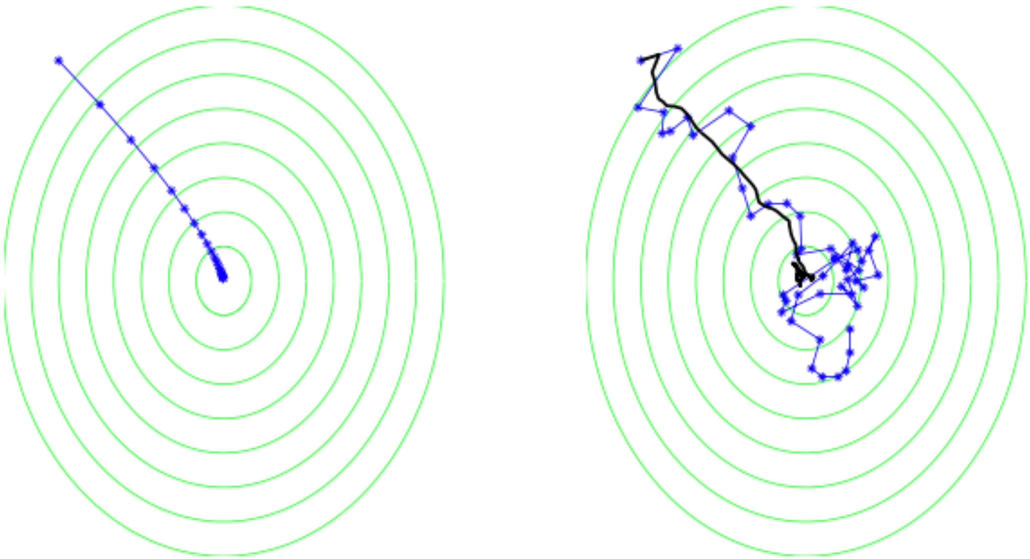


Figure: Minimize $1.25(x+6)^2 + (y-8)^2$

(Figure from "Understanding Machine Learning: From Theory to Algorithms".)

Stochastic gradient descent

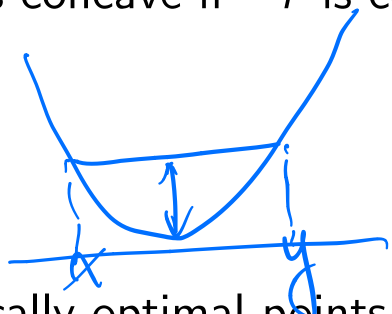
- Each update is efficient in both time and space
- Can be slow to converge
- Popular in large-scale ML, including non-convex problems
- In practice,

  Randomly sample examples.
  Fixed or diminishing step sizes, e.g. $1/t$, $1/\sqrt{t}$.
  Stop when objective does not improve.

Convex optimization (unconstrained)

- A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if for all $x, y \in \mathbb{R}^d$ and $\theta \in [0, 1]$ we have
$$f(\theta x + (1-\theta)y) \le \theta f(x) + (1-\theta) f(y) \; .$$
- f is concave if $-f$ is convex.

$f''(x) \geq 0$ "curvature"



- Locally optimal points are also globally optimal.
- For unconstrained problems, x is optimal iff $\nabla f(x) = 0$.

Table of Contents

Zero-one loss

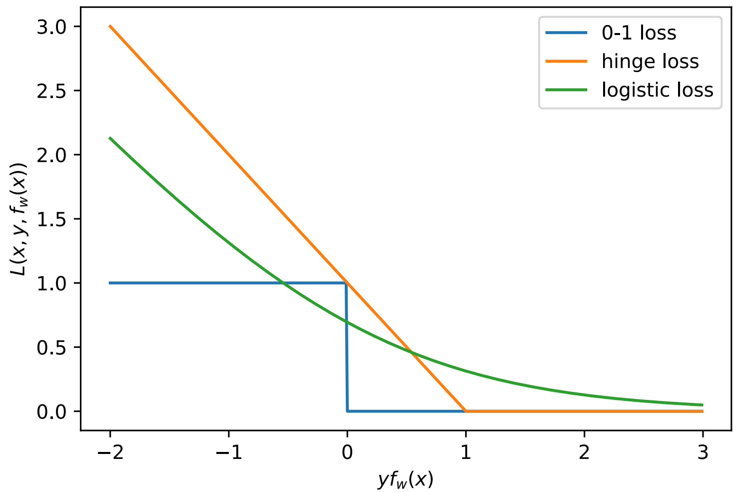
- Settings

  Binary classification: $y \in \{+1, -1\}$.
  Scorer $f_w \colon \mathcal{X} \to \mathrm{R}$ parametrized by $w \in \mathrm{R}^d$.
  Output prediction: $\operatorname{sign}(f_w(x))$.

- Zero-one (0-1) loss

$\mathbb{I}(\operatorname{sign}(f_w(x)) = y)$

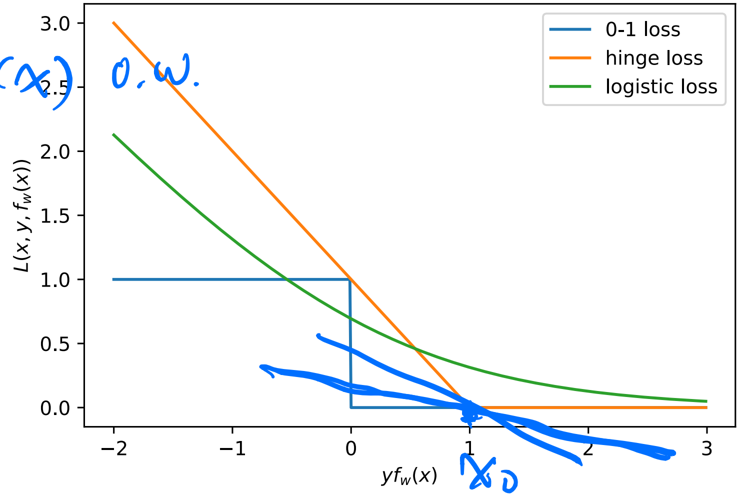
$$L(x, y, f_w) = \mathbb{I}\left[\underbrace{y f_w(x)}_{\text{(functional) margin}} \le 0\right]$$



Hinge loss

$$L(x, y, f_w) = \max(\underline{1 - yf_w(x)}, 0)$$

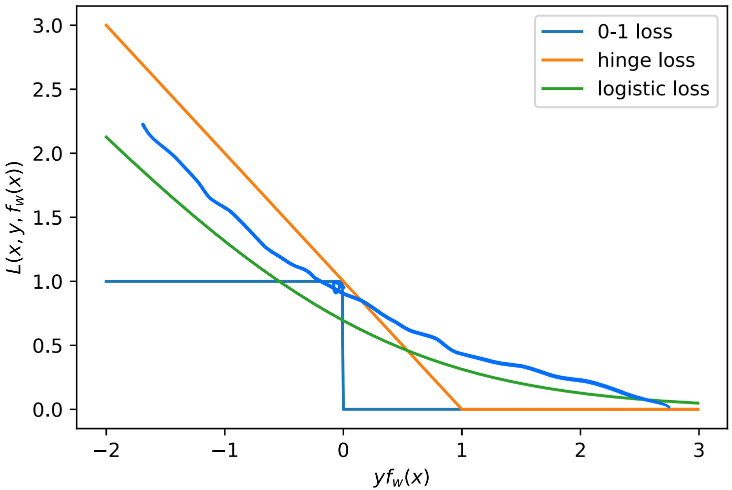
$$\nabla_w L = \begin{cases} 0 & \text{if } yf_w(x) \geq 1 \\ -y\nabla_w f(x) & \text{o.w.} \end{cases}$$



Subgradient: $f(x) \geq \underline{x_0 + g^T(x - x_0)}$

Logistic loss

$$L(x, y, f_w) = \log(1 + e^{-yf_w(x)}) \cdot \frac{1}{\log 2}$$



Summary

- Bias-complexity trade-off: choose hypothesis class based on prior knowledge
- Learning algorithm: empirical risk minimization
- Optimization: stochastic gradient descent