Machine Learning

Lecture 14: Generalization 1

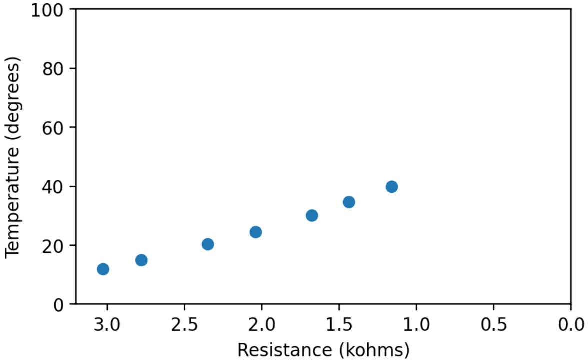
Hao Tang

October 26, 2022

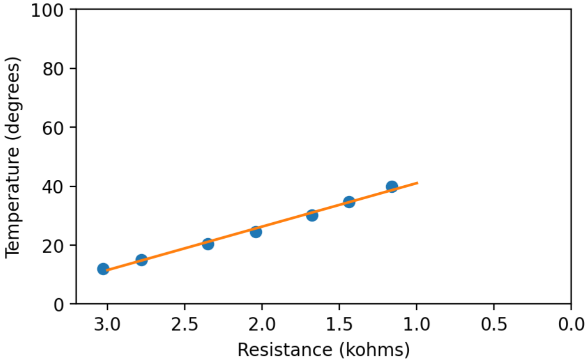
Programming with data

- Minimizing a loss function on a data set produces a program with data.
- How do we know if a program learned with data is correct?

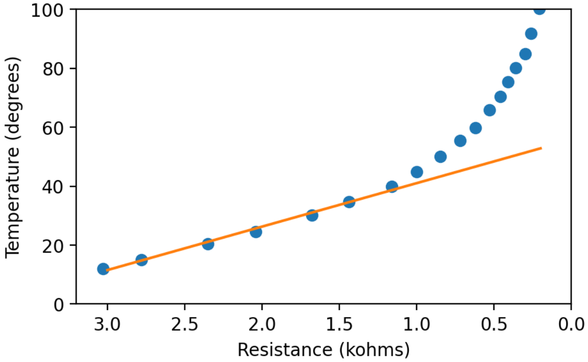
Failure case 1



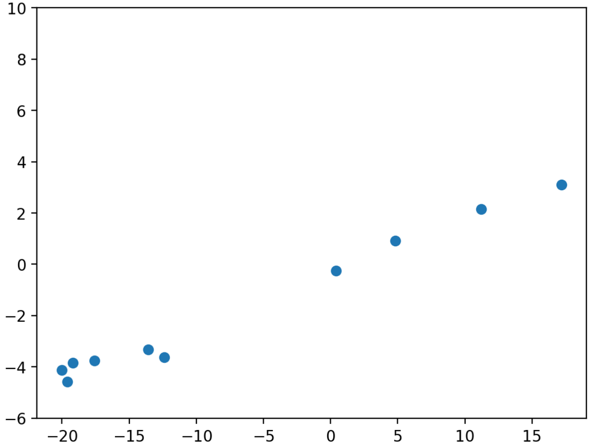
Failure case 1



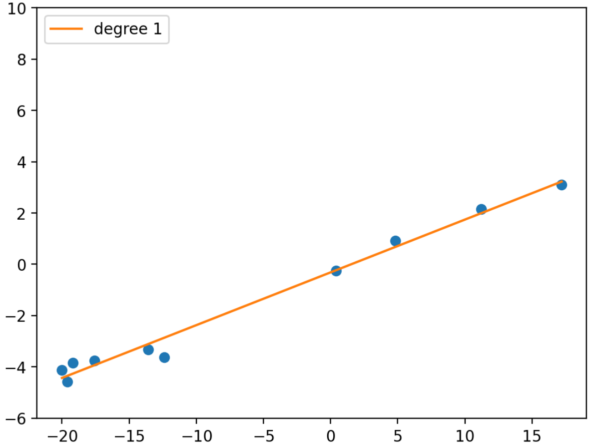
Failure case 1



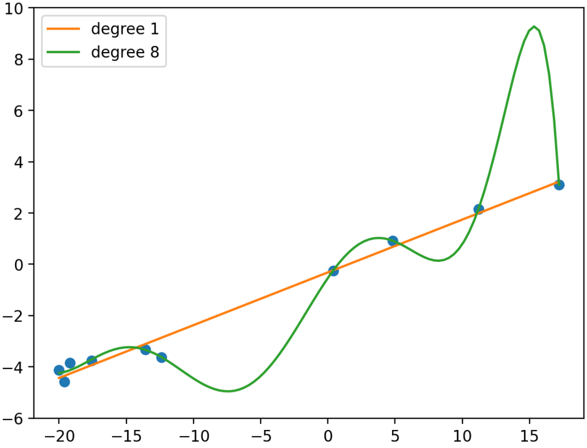
Failure case 2



Failure case 2



Failure case 2



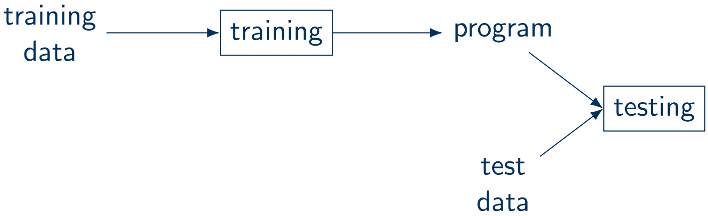
Programming with data

- A program written with data is correct if produces the indended results on unseen data.
- **Generalization** is defined as being (approximately) correct on unseen data (most of the time).

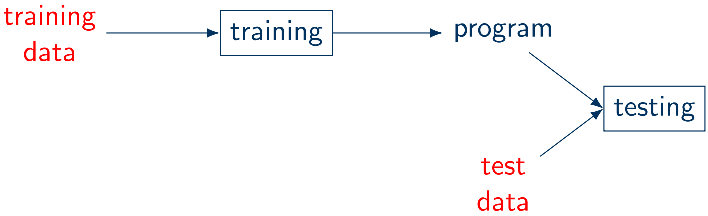
How to measure generalization

training data $\longrightarrow$ $\boxed{\text{training}}$ $\longrightarrow$ program

How to measure generalization



How to measure generalization



Generalization

- There exists a common (yet unknown) distribution $\mathcal{D}$ where both the training data and the test data are drawn from.

- The training set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ includes i.i.d. samples drawn from $\mathcal{D}$.

- The **training error** for a loss ℓ and a program h is defined as

$$L_S(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(x_i)). \tag{1}$$

- If we have a test set S', then $L_{S'}(h)$ is the error on the test set (or test error for short) for a program h.

Generalization

- The generalization error for a program h is defined as

$$L_{\mathcal{D}}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(y, h(x))]. \tag{2}$$

- The test error $L_{S'}(h)$ of a program h is an estimate of the generalization error $L_{\mathcal{D}}(h)$.

- The goal of learning is to find a program h with low generalization error $L_{\mathcal{D}}(h)$.

A learning algorithm

- $A : (\mathcal{X} \times \mathcal{Y})^m \to \mathcal{H}$
- In words, a learning algorithm is a function that takes a data set of size m and returns a function from the hypothesis class $\mathcal{H}$.
- A hypothesis class $\mathcal{H}$ is the set of possible programs of a particular form.
- For example, a linear classifier is $\mathcal{H} = \{w : x \mapsto w^\top \phi(x)\}$.

Probably approximately correct

A hypothesis class $\mathcal{H}$ is PAC-learnable with A if for any distribution $\mathcal{D}$, for all $\epsilon > 0$ and $0 \leq \delta \leq 1$ such that

$$\mathbb{P}_{S\sim\mathcal{D}^m}\left[L_{\mathcal{D}}(A(S)) - \min_{h'\in\mathcal{H}} L_{\mathcal{D}}(h') > \epsilon\right] < \delta. \tag{3}$$

Probably approximately correct

- The data set S is what is random.
- $L_{\mathcal{D}}(A(S))$ is also random.
- $\min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h')$ is the best a program of this form (in $\mathcal{H}$) can do.
- ϵ is the error tolerance, the approximately correct part.
- δ is the confidence probability, the probably part.

Probably approximately correct

- Suppose $\epsilon = 0.01$ and $\delta = 0.05$.
- We can say that our learning algorithm A can achieve at most 1% worse than the best of any other programs of this form 95% of the time.

No free lunch theorem

Suppose $|\mathcal{X}| = 2m$. For any learning algorithm A, there is a distribution $\mathcal{D}$ and $f : \mathcal{X} \to \{0,1\}$ such that $L_{\mathcal{D}}(f) = 0$, but

$$\mathbb{P}_{S \sim \mathcal{D}^m}\left[L_{\mathcal{D}}(A(S)) \geq \frac{1}{10}\right] \geq \frac{1}{10}. \tag{4}$$

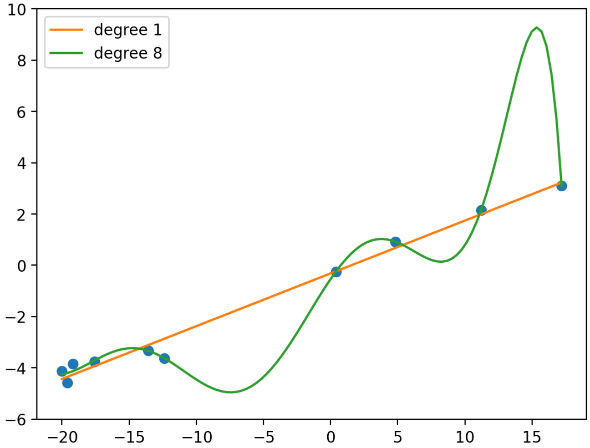
No free lunch theorem

- The 2 and 10 are arbitrary constants.
- In words, for any learning algorithm there exists a task that it will fail.
- What should we do?
- Don't compare to the best f in the universe.
- Compare to the best in the hypothesis space.

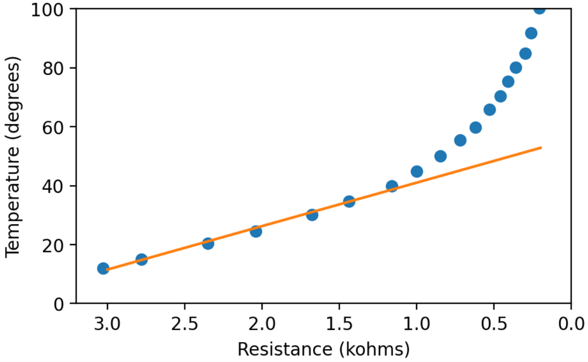
Tradeoff between model complexity and generalization

- When we say we only compare to the best in $\mathcal{H}$, we are comparing against $\min_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$.
- When $\mathcal{H}$ is large, $\min_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$ becomes lower.
- When $\mathcal{H}$ is the universe of all functions, we cannot learn.
- $\mathcal{H}$ needs to be about the right size.
- $\mathcal{H}$ can be a large, but the range of A needs to be about the right size.
- For example, we can only run a finite number of steps with stochastic gradient descent, so the range we can explore is limited by the algorithm.

Failure case 2



Failure case 1



Error decomposition

$$L_{\mathcal{D}}(h) = \underbrace{L_{\mathcal{D}}(h) - \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h')}_{\text{estimation error}} + \underbrace{\min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h')}_{\text{approximation error}} \tag{5}$$

- Approximation error is due to the choice of $\mathcal{H}$.
- Estimation error is due to not finding the best program in $\mathcal{H}$.

What about training?

- Since we only have a data set S, we can only minimize $L_S(h)$.
- Minimizing $L_S(h)$ is called empirical risk minimization (ERM).
- If $h_{\text{ERM}} = \operatorname{argmin}_{h \in \mathcal{H}} L_S(h)$, do we know anything about $L_{\mathcal{D}}(h_{\text{ERM}})$?

Uniform convergence

We say that $\mathcal{H}$ has uniform convergence property if for any distribution $\mathcal{D}$, for all $\epsilon > 0$ and $0 \leq \delta \leq 1$ such that for every $h \in \mathcal{H}$, we have

$$\mathbb{P}_{S \sim \mathcal{D}^m} \left[|L_S(h) - L_{\mathcal{D}}(h)| > \epsilon \right] < \delta. \tag{6}$$

Uniform convergence

- Uniform convergence assures that the training error and generalization error are not far from each other.

- This has to happen for all $h \in \mathcal{H}$, the uniform part (and a strong requirement).

Uniform convergence

- If we have uniform convergence,

$$L_{\mathcal{D}}(h_{\text{ERM}}) \leq L_S(h_{\text{ERM}}) + \epsilon \leq L_S(h) + \epsilon \leq L_{\mathcal{D}}(h) + \epsilon + \epsilon \tag{7}$$

  for any $h \in \mathcal{H}$.

- In particular,

$$L_{\mathcal{D}}(h_{\text{ERM}}) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + 2\epsilon. \tag{8}$$

- If $\mathcal{H}$ has uniform convergence property, then $\mathcal{H}$ is PAC-learnable with ERM.