Machine Learning: Generalization 1

Hao Tang

March 17, 2025

Machine learning is about programming with data

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

Correctness of classical programs

- A program is correct if it has the desired behavior on **all** input.
- Correctness is achieved through mathematical proofs and careful engineering.

Correctness of learned programs

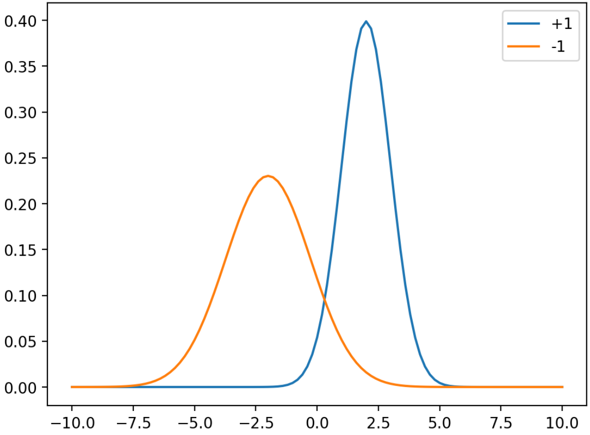
$$\boxed{\text{training set}} \xrightarrow{\text{training}} \text{classifier}$$

- Imagine we have trained a binary classifier.
- We know the loss on the training set.
- Even on the training set, the loss might not be 0.
- Can we say anything about the loss outside of the training set?
- Is it even possible? What assumptions do we need?

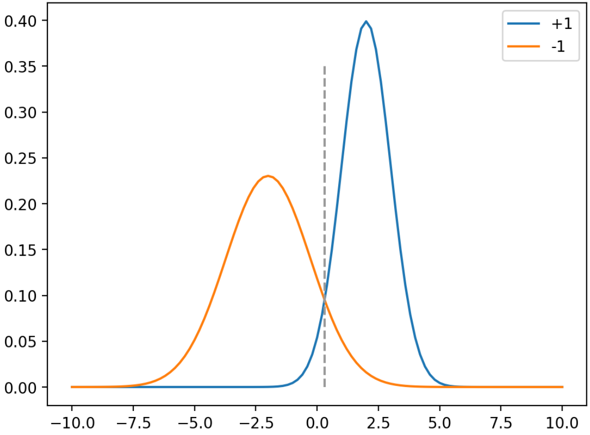
The big picture

- What is generalization? PAC learning and ERM
- When is generalization possible? Uniform convergence and VC dimension
- How to achieve generalization? Overfitting, underfitting, and regularization
- Generalization of neural networks. Universal approximation, overparameterization, and interpolation

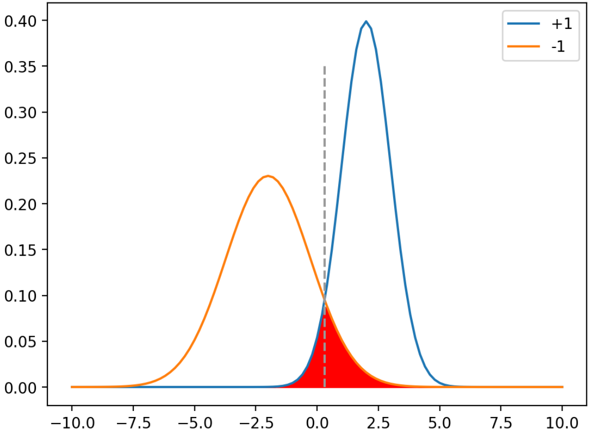
What happens if the data is Gaussian?



What happens if the data is Gaussian?



What happens if the data is Gaussian?



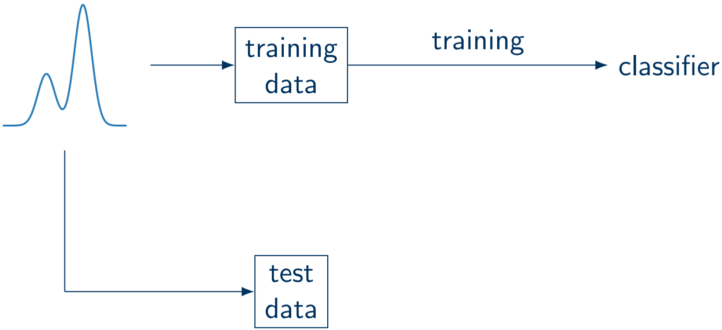
Gaussian data

- We need to know
  - the two distributions are Gaussian
  - their means
  - their variances.
- The decision boundary
  - can be found without training
  - is optimal (in the sense that no other boundary achieves a lower error).
- Next questions
  - What if we don't know the means?
  - What if we don't know the variances?
  - What if the two distributions are not Gaussian?
  - What if we don't know what the distributions are?

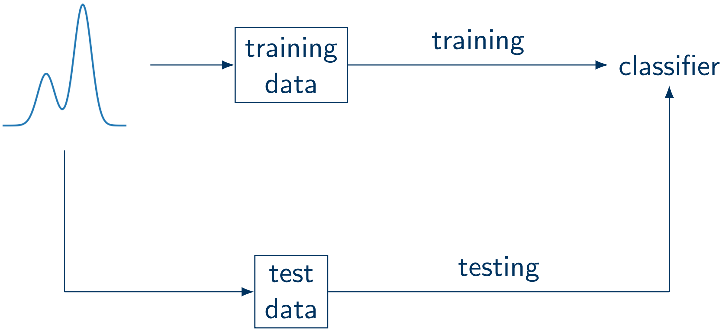
Generalization

Generalization

Generalization



Generalization



Generalization

- A data set is said to be **i.i.d. (independent and identically distributed)** if the data points come from the same distribution and are statistially independent from each other.
- A function (or program) is said to **generalize** to data within a distribution if the function achieves a low error on data drawn from that distribution *in expectation*.
- In particular, if a function generalizes then the function has to achieve a low error on both the training set and the test set.
- We do not know the distribution, and only have data drawn from the distribution.
- The only assumption is i.i.d. data.

Generalization

- There exists a 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 $L_{\mathcal{D}}(h)$.

$$\mathbb{E}_{S\sim\mathcal{D}^n}[L_S(h)] = \mathbb{E}_{S'\sim\mathcal{D}^{n'}}[L_{S'}(h)] = L_{\mathcal{D}}(h) \tag{3}$$

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

Learning algorithms and hypothesis classes

- A learning algorithm A 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 $\mathcal{H} = \{x \mapsto w^\top x : w \in \mathbb{R}^d\}$.

Probably approximately correct

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

$$\mathbb{P}_{S \sim \mathcal{D}^n}\left[L_{\mathcal{D}}(A(S)) - \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') > \epsilon\right] < \delta \tag{4}$$

for any $n \geq N$.

Probably approximately correct

- The data set S is is a random variable.
- $A(S)$ is a program returned by A after training on S.
- $L_{\mathcal{D}}(A(S))$ is also a random variable.
- $\min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h')$ is the best error we can achieve among all programs in $\mathcal{H}$.
- ϵ is the error tolerance, the approximately correct part.
- δ is the confidence probability, the probably part.

Probably approximately correct

- Imagine we do the following experiment many many times.
  1. Draw a training set S and obtain a trained program $A(S)$.
  2. Evaluate $L_{\mathcal{D}}(A(S)) - \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h')$.
  3. Repeat

- On average, the chance of seeing $L_{\mathcal{D}}(A(S)) - \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') > \epsilon$ is δ.

- Think of ϵ and δ as something small, close to 0.

- With high probability $1 - \delta$, the two terms $L_{\mathcal{D}}(A(S))$ and $\min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h')$ only differ by a small amount ϵ.

- With high probability, the program learned by A achieves a similar error to the best program in $\mathcal{H}$.

PAC learning

- PAC learnability is merely a definition.
- The minimum number of samples required, N, also known as **sample complexity**, is a function of $\mathcal{H}$, ϵ, and δ.
- We can now ask, "is the set of linear classifiers PAC learnable if we minimize the zero-one loss on a training set?"

Empirical risk minimization

- Minimizing the loss on a training set is also known as **empirical risk minimization (ERM)**.

$$A_{\mathrm{ERM},\mathcal{H}}(S) = h_{\mathrm{ERM}} = \operatorname*{argmin}_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(x_i)) \tag{5}$$

- The set of linear classifiers is $\mathcal{H}_{\mathrm{lin}} = \{w \mapsto w^\top x : w \in \mathbb{R}^d\}$.
- We can now formally ask, "is the set of linear classifiers $\mathcal{H}_{\mathrm{lin}}$ PAC learnable with ERM?"
- And if so, how does the sample complexity N depends on $\mathcal{H}_{\mathrm{lin}}$, ϵ, and δ?

The universe of all programs

- Instead of choosing the set linear classifiers $\mathcal{H}_{\text{lin}} = \{x \mapsto w^\top x : w \in \mathbb{R}^d\}$, can we choose $\mathcal{H}_{\text{universe}} = \{\text{any function in the universe}\}$?

- We can now ask, "is $\mathcal{H}_{\text{universe}}$ PAC learnable with ERM or with any other learning algorithms?"

Probably approximately correct

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

$$\mathbb{P}_{S\sim\mathcal{D}^n}\left[L_{\mathcal{D}}(A(S)) - \min_{h'\in\mathcal{H}} L_{\mathcal{D}}(h') > \epsilon\right] < \delta \tag{6}$$

for any $n \geq N$.

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

$$\mathbb{P}_{S \sim \mathcal{D}^m} \left[L_{\mathcal{D}}(A(S)) \geq \frac{1}{10} \right] \geq \frac{1}{10}. \tag{7}$$

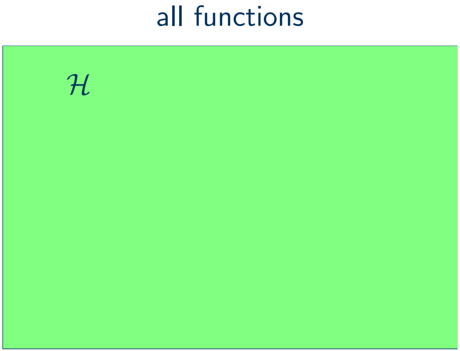
No free lunch theorem

- The 2 and 10 are arbitrary constants.
- In words, for any learning algorithm, there exists a distribution and a perfect function, but the learning algorithm has a sufficiently large error with sufficiently high probability.
- What is the problem?
- A cheater is an algorithm that knows $\mathcal{D}$.
- The universe always includes whatever the cheater can find.
- Don't compare to the cheater that knows $\mathcal{D}$. Instead, compare to the best in the hypothesis space.

No free lunch theorem

all functions

No free lunch theorem



No free lunch theorem

all functions