Lecture 2

The Simplest Supervised Learning: Linear Regression

Framework of Statistical Learning Theory

(supervised learning)

X: Instance Space (e.g. $\mathbb{R}^{16\times 16}$ for 16x16 greyscale images)

Y: Label Space (e.g. $\mathbb{R}$ for regression or $\{1,\ldots,k\}$ for multi-class classification)

$\mathscr{D}$: Probability Distribution over $X \times Y$ (*unknown*, *but can sample from*)

$\ell : Y \times Y \to \mathbb{R}_{\geq 0}$ Loss or Cost Function (e.g. $\ell(y,\hat{y}) = (y-\hat{y})^2$ for $Y = \mathbb{R}$)

Objective

Given a training set $S = \left\{ \left(x_i, y_i\right) \right\}_{i=1}^{m}$ drawn i.i.d. from $\mathscr{D}$, return hypothesis (predictor) $h : X \to Y$ that minimizes the population loss or expected risk:

$$\boxed{L_{\mathscr{D}}(h) := \mathbb{E}_{(x,y)\sim\mathscr{D}}[\ell(y, h(x))]}$$

Approximate Approach

Predetermine or assume a hypotheses space $\mathscr{H} \subset Y^X$, and return hypothesis $h \in \mathscr{H}$ that minimizes sample loss or empirical loss or empirical risk:

$$\boxed{L_S(h) := \frac{1}{m}\sum_{i=1}^{m} \ell\left(y_i, h\left(x_i\right)\right)}$$

Framework of Statistical Learning Theory

(supervised learning)

X: Instance Space (e.g. $\mathbb{R}^{16\times 16}$ for 16x16 greyscale images)

Y: Label Space (e.g. $\mathbb{R}$ for regression or $\{1,\ldots,k\}$ for multi-class classification)

$\mathscr{D}$: Probability Distribution over $X \times Y$ (*unknown*, *but can sample from*)

$\ell : Y \times Y \to \mathbb{R}_{\geq 0}$ Loss or Cost Function (e.g. $\ell(y,\hat{y}) = (y-\hat{y})^2$ for $Y = \mathbb{R}$)

Objective

Given a training set $S = \left\{ \left(x_i, y_i\right) \right\}_{i=1}^{m}$ drawn i.i.d. from $\mathscr{D}$, return hypothesis (predictor) $h : X \to Y$ that minimizes the population loss or expected risk:

$$\boxed{L_{\mathscr{D}}(h) := \mathbb{E}_{(x,y)\sim\mathscr{D}}[\ell(y, h(x))]}$$

Approximate Approach

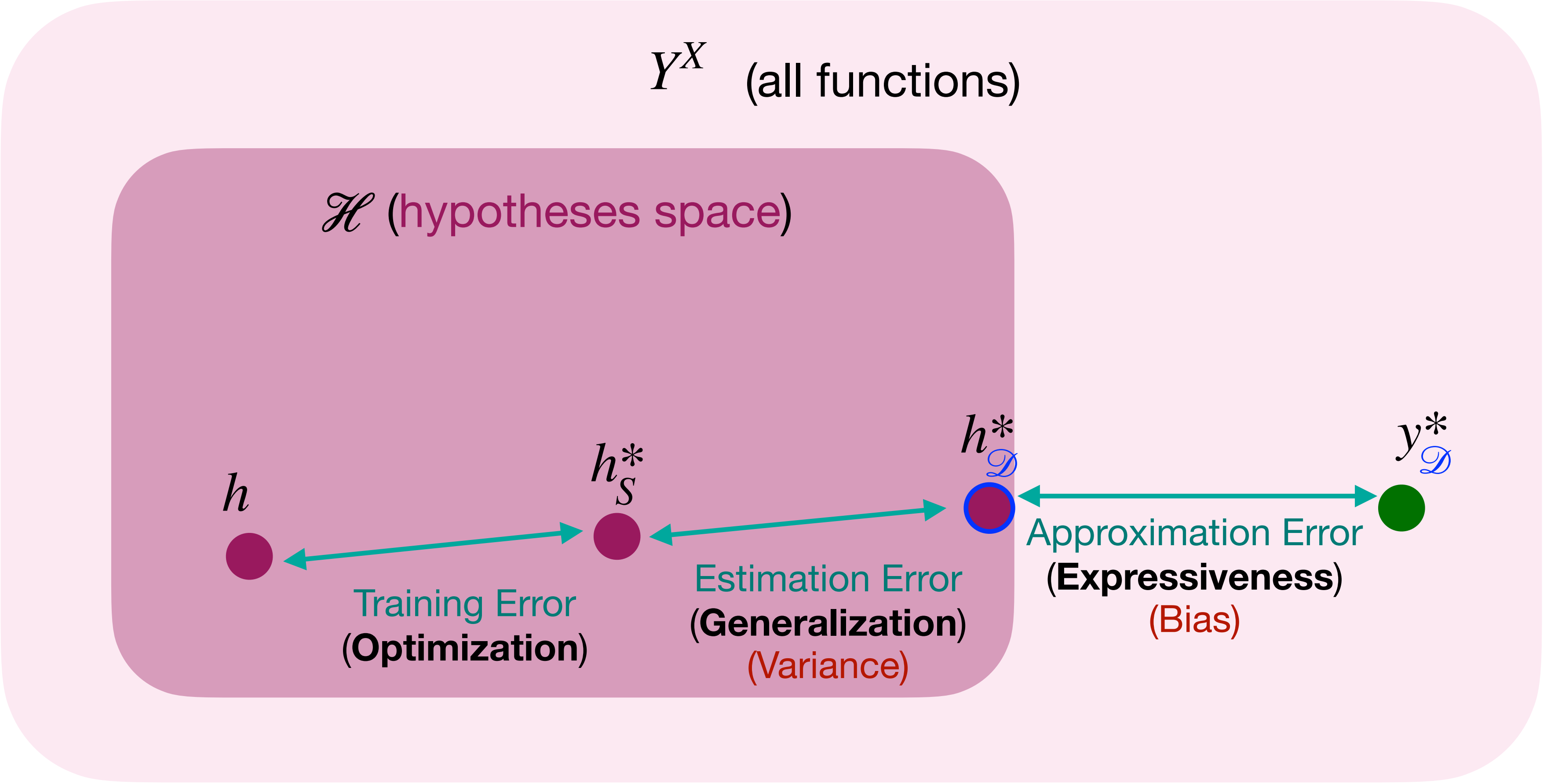
Predetermine or assume a hypotheses space $\mathscr{H} \subset Y^X$, and return hypothesis $h \in \mathscr{H}$ that minimizes sample loss or empirical loss or empirical risk:

$$\boxed{L_S(h) := \frac{1}{m}\sum_{i=1}^{m} \ell\left(y_i, h\left(x_i\right)\right)}$$

Note: An algorithm that searches for empirically optimal h_S^* is called a "learning algorithm."

Jargons in Statistical Learning Theory (SLT)

(**Expressiveness**, **Generalization**, **Optimization**)



$y^*_{\mathscr{D}}$: ground truth (minimizer of population loss over Y^X)

$h^*_{\mathscr{D}}$: optimal hypothesis (minimizer of population loss over $\mathscr{H}$ -infinite data sample)

h^*_S: empirically optimal hypothesis (minimizer of sample loss over $\mathscr{H}$)

h: returned hypothesis

Note: For sampling to give a good proxy, we must enforce the *consistency condition* in the infinite sample size limit. Namely, $\lim_{m\to\infty} h^*_S = h^*_{\mathscr{D}}$.

Linear Regression: the simplest example of supervised learning

Suppose there's a god-given linear relationship between an input $\boldsymbol{x} \in \mathbb{R}^d$ and the continuous output (label) $y \in \mathbb{R}$ according to

$$y = f(\boldsymbol{x}) + \eta_i = \boldsymbol{w}_{true}^T \boldsymbol{x} + \eta_i \ ,$$

where $\boldsymbol{w}_{true} \in \mathbb{R}^d$, and η_i is an i.i.d. drawn from $\mathcal{N}(0,\sigma^2)$.

Linear Regression: the simplest example of supervised learning

Suppose there's a god-given linear relationship between an input $\boldsymbol{x} \in \mathbb{R}^d$ and the continuous output (label) $y \in \mathbb{R}$ according to

$$y = f(\boldsymbol{x}) + \eta_i = \boldsymbol{w}_{true}^T \boldsymbol{x} + \eta_i \ ,$$

where $\boldsymbol{w}_{true} \in \mathbb{R}^d$, and η_i is an i.i.d. drawn from $\mathcal{N}(0,\sigma^2)$.

Suppose the *loss function* is the square error $\ell(y, \hat{y}) = (y - \hat{y})^2$, remember that the objective of SLT is to *search* for the hypothesis function $h(\boldsymbol{x})$ that minimizes the empirical risk

$$L_S(h) := \frac{1}{m} \sum_{i=1}^{m} \ell\left(y_i, h\left(\boldsymbol{x}^{(i)}\right)\right),$$

given a training set $S = \left\{ \left(\boldsymbol{x}^{(i)}, y_i\right) \right\}_{i=1}^{m}$.

Linear Regression: the simplest example of supervised learning

Suppose there's a god-given linear relationship between an input $\boldsymbol{x} \in \mathbb{R}^d$ and the continuous output (label) $y \in \mathbb{R}$ according to

$$y = f(\boldsymbol{x}) + \eta_i = \boldsymbol{w}_{true}^T \boldsymbol{x} + \eta_i \,,$$

where $\boldsymbol{w}_{true} \in \mathbb{R}^d$, and η_i is an i.i.d. drawn from $\mathcal{N}(0,\sigma^2)$.

Suppose the *loss function* is the square error $\ell(y, \hat{y}) = (y - \hat{y})^2$, remember that the objective of SLT is to *search* for the hypothesis function $h(\boldsymbol{x})$ that minimizes the empirical risk

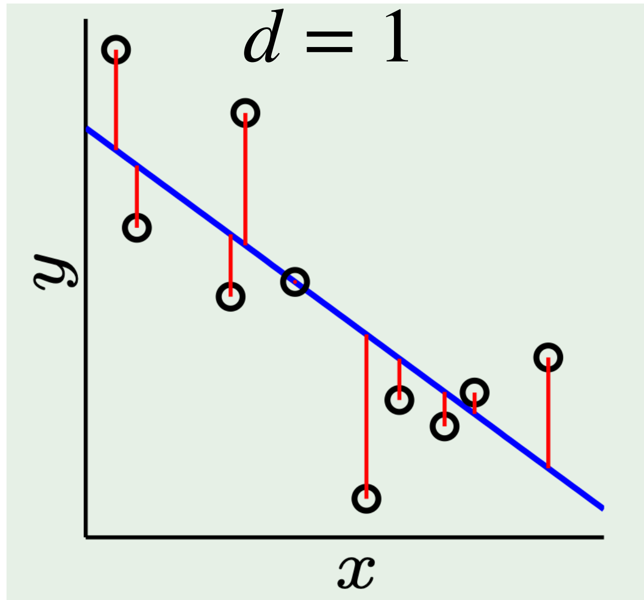
$$L_S(h) := \frac{1}{m} \sum_{i=1}^{m} \ell\left(y_i, h\left(\boldsymbol{x}^{(i)}\right)\right),$$

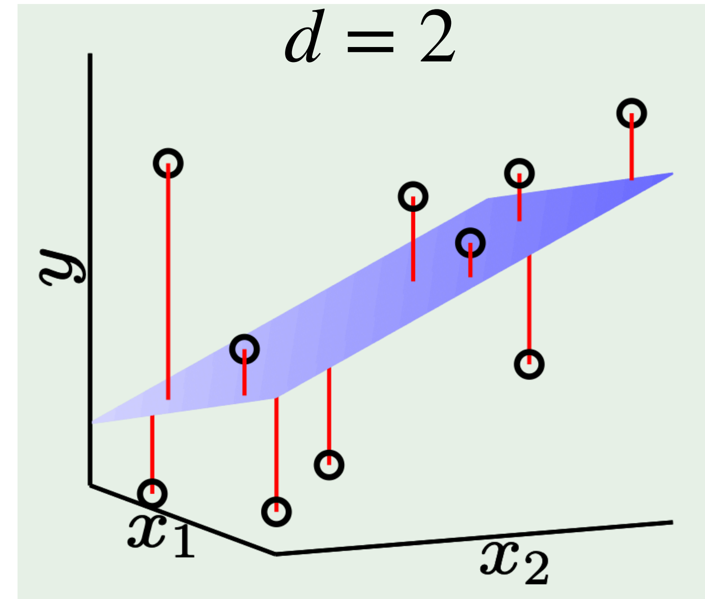
given a training set $S = \left\{ \left(\boldsymbol{x}^{(i)}, y_i\right) \right\}_{i=1}^{m}$.

For simplicity, suppose we restrict our hypothesis class to be the simplest class possible, i.e. a linear continuous real-valued function (The SAME class as the god-given function, can't be simpler than that!)

$$h(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x},$$

can we find a learning algorithm that spits out h_S^*?





In other words, how do we find a d-dimensional hyperplane with a minimum square error from training data.

For brevity, let's rewrite the empirical risk as an L^2 norm

$$L_S(h) = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y_i\right)^2 = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2,$$

$$\boldsymbol{X} = \begin{bmatrix} -\boldsymbol{x}^{(1)\top}- \\ -\boldsymbol{x}^{(2)\top}- \\ \vdots \\ -\boldsymbol{x}^{(d)\top}- \end{bmatrix}$$

where for $\boldsymbol{x} = \left(x_1, \ldots, x_d\right) \in \mathbb{R}^d$ the L^p norm of $\boldsymbol{x}$ is defined as

$$\|\boldsymbol{x}\|_p = \left(\left|x_1\right|^p + \cdots + \left|x_d\right|^p\right)^{\frac{1}{p}}.$$

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{bmatrix}$$

The learning algorithm should be able to spit out the weight $\boldsymbol{w}$ that minimizes the Empirical Risk Minimisation (ERM) problem, which we'll call an in-sample error

$$E_{in}(\boldsymbol{w}^*) \equiv L_S(h_S^*) = \frac{1}{m} \min_{\boldsymbol{w} \in \mathbb{R}^d} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2$$

The learning algorithm should be able to spit out the weight $\boldsymbol{w}$ that minimizes the Empirical Risk Minimisation (ERM) problem, which we'll call an in-sample error

$$E_{in}(\boldsymbol{w}^*) \equiv L_S(h_S^*) = \frac{1}{m} \min_{\boldsymbol{w} \in \mathbb{R}^d} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2$$

The necessary condition for $\boldsymbol{w}$ to be a critical point is

$$\nabla E_{in}(\boldsymbol{w}^*) = \frac{2}{N} \boldsymbol{X}^\top (\boldsymbol{X}\boldsymbol{w}^* - \boldsymbol{y}) = \boldsymbol{0}$$

The learning algorithm should be able to spit out the weight $\boldsymbol{w}$ that minimizes the Empirical Risk Minimisation (ERM) problem, which we'll call an in-sample error

$$E_{in}(\boldsymbol{w}^*) \equiv L_S(h_S^*) = \frac{1}{m} \min_{w \in \mathbb{R}^d} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2$$

The necessary condition for $\boldsymbol{w}$ to be a critical point is

$$\nabla E_{in}(\boldsymbol{w}^*) = \frac{2}{N} \boldsymbol{X}^\top (\boldsymbol{X}\boldsymbol{w}^* - \boldsymbol{y}) = \boldsymbol{0}$$

In this case, the critical point turns out to be the minimizer (convex optimization)

$$\boxed{\boldsymbol{w}^* = \boldsymbol{X}^\dagger \boldsymbol{y},}$$

where we denote the Moore-Penrose pseudo-inverse of $\boldsymbol{X}$ as

$$\boldsymbol{X}^\dagger = \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top$$

The learning algorithm should be able to spit out the weight $\boldsymbol{w}$ that minimizes the Empirical Risk Minimisation (ERM) problem, which we'll call an in-sample error

$$E_{in}(\boldsymbol{w}^*) \equiv L_S(h_S^*) = \frac{1}{m} \min_{w \in \mathbb{R}^d} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2$$

The necessary condition for $\boldsymbol{w}$ to be a critical point is

$$\nabla E_{in}(\boldsymbol{w}^*) = \frac{2}{N}\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{w}^* - \boldsymbol{y}) = \boldsymbol{0}$$

In this case, the critical point turns out to be the minimizer (convex optimization)

$$\boxed{\boldsymbol{w}^* = \boldsymbol{X}^\dagger \boldsymbol{y},}$$

where we denote the Moore-Penrose pseudo-inverse of $\boldsymbol{X}$ as

$$\boldsymbol{X}^\dagger = \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top$$

So we have the first learning algorithm to find the unique minimizer of the least-square linear regression problem in one-shot!

- Step 1: From training data, construct the *design matrix* $\boldsymbol{X}$ and the vector $\boldsymbol{y}$.
- Step 2: Compute the Pseudo-inverse of the design matrix $\boldsymbol{X}^\dagger = \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top$
- Step 3: Perform the matrix vector multiplication $\boldsymbol{w}^* = \boldsymbol{X}^\dagger \boldsymbol{y}$ to obtain the empirically optimal hypothesis $h_S^*(\boldsymbol{x}) = \boldsymbol{w}^{*T}\boldsymbol{x}$.

Recall from Linear Algebra that invertibility of the matrix is always tricky, as one needs to check if the matrix has full-rank (typically the case when $m \gg d$)…

Recall from Linear Algebra that invertibility of the matrix is always tricky, as one needs to check if the matrix has full-rank (typically the case when $m \gg d$)…

However, assuming invertibility, here's an important result you'll prove in homework 1

$$\bar{E}_{in} \equiv \mathbb{E}_D\left[E_{in}(\boldsymbol{w}_D^*)\right] = \sigma^2\left(1 - \frac{d}{m}\right)$$

(in-sample error)

$$\bar{E}_{out} \equiv \mathbb{E}_D\left[E_{out}(\boldsymbol{w}_D^*)\right] = \sigma^2\left(1 + \frac{d}{m}\right)$$

(out-of-sample error)

Recall from Linear Algebra that invertibility of the matrix is always tricky, as one needs to check if the matrix has full-rank (typically the case when $m \gg d$)…

However, assuming invertibility, here's an important result you'll prove in homework 1

$$\boxed{\begin{aligned} \bar{E}_{in} &\equiv \mathbb{E}_D \left[E_{in}(\boldsymbol{w}_D^*) \right] = \sigma^2 \left(1 - \frac{d}{m} \right) \\ \bar{E}_{out} &\equiv \mathbb{E}_D \left[E_{out}(\boldsymbol{w}_D^*) \right] = \sigma^2 \left(1 + \frac{d}{m} \right) \end{aligned}}$$

(in-sample error)

(out-of-sample error)

This gives us the *generalization error*

$$\boxed{|\bar{E}_{out} - \bar{E}_{in}| = 2\sigma^2 \left(\frac{d}{m} \right)}$$

Recall from Linear Algebra that invertibility of the matrix is always tricky, as one needs to check if the matrix has full-rank (typically the case when $m \gg d$)…

However, assuming invertibility, here's an important result you'll prove in homework 1

$$\boxed{\begin{aligned} \bar{E}_{in} &\equiv \mathbb{E}_D \left[E_{in}(\boldsymbol{w}_D^*) \right] = \sigma^2 \left(1 - \frac{d}{m} \right) \\ \bar{E}_{out} &\equiv \mathbb{E}_D \left[E_{out}(\boldsymbol{w}_D^*) \right] = \sigma^2 \left(1 + \frac{d}{m} \right) \end{aligned}}$$

(in-sample error)

(out-of-sample error)

This gives us the *generalization error*

$$\boxed{| \bar{E}_{out} - \bar{E}_{in} | = 2\sigma^2 \left(\frac{d}{m} \right)}$$

- Large generalization error if $d \gg m$. Make sense since you'll likely fit a noisy subspace of the actual high-dimensional hyperplane.
- Even when $d \approx m$, noise suppresses learning the actual high-dimensional hyperplane.
- We'll learn how to "*regularise*" learning to not be too sensitive to noise (lower the variance of the learned models).

Schematic of how in-sample error and out-of-sample error are related in linear least square