# Lecture 1



---

image generation, text generation, game playing, protein folding.

Taxonomy: supervised, unsupervised, reinforcement.

- Supervised learning: Aim to predict outputs of future datapoints.
- Unsupervised learning: Aim to discover hidden patterns and explore data.
- Reinforcement learning: Aim to make sequential decisions.

Supervised ML: predict future outcomes using past outcomes.

> image classification, machine translation

house price prediction: training data, extract features, correlation analysis...

Sale price $\approx$ price per sqft (slope) $\times$ square footage $+$ fixed expense (intercept)

## Concept

### General framework for supervised learning

An input space: $\mathcal{X} \in \mathbb{R}^d$

> Data points in $d$ dimensions
>
> In previous eg. $d: 1$ (eg. footage)

Anf output space: $\mathcal{Y}$

> $y \in \mathbb{R}$ for sale price prediction
>
> $y \in \{\pm 1\}$ label classification

Goal: yearn a prediction $f(x) : \mathcal{X} \to y$

**Loss function:** $l(f(x), y)$ , depends on the task.

> squared loss: $for\ y \in \mathcal{R},\ l(f(x), y) = (f(x) - y)^2$

**Minimize loss over some distribution $D$ over instances** $(x, y)$

Def: Risk of prediction $f(x)$ is:

$$\begin{aligned} R(f) &= E_{(x,y)\sim D}[l(f(x), y)] \\ &= \sum_{x', y'} Prob_D(x = x', y = y') \cdot l(f(x'), y') \end{aligned}$$

This is the actual, or the ideal parameter of the model, often hard to compute.

Challenge: Don't know distribution of data $D$.

**i.i.d assumption:** have a set of labelled instances distributed independently & identically from $D$. If $E_1, E_2$ are independent, then $\Pr(E_1 \cap E_2) = \Pr(E_1) \Pr(E_2)$ .

Theoretical abstraction, often useful. Pay attention to whether this is valid! (need "stationarity")

Def: Given a set of labelled datapoeints:

$$S = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$$

the empirical risk of any $f : \mathcal{X} \to y$

$$\hat{R}_s(f) = \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i)$$

which denotes the average loss over the data points.

**Function class:** a collection of functions: $\mathcal{X} \to y$

$$X \in \mathbb{R}, y \in \mathbb{R}, F = f : y = wx + c$$

**ERM: Empirical Risk Minimizer**

Because the $R(f)$ is hard to compute, so we compute the approximation drawn from data with iid assumption and minimize the $\hat{R}_s(f)$ .

Def: function class $\mathcal{F} = \{f : \mathcal{X} \to y\}$ $l$ set of labelled datapoints $S$, ERM corresponds to

$$\min_{f \in \mathcal{F}} \hat{R}_s(f) = \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i)$$

**Generalization**

$$R(f) = \hat{R}_s(f) + (R(f) - \hat{R}_s(f))$$

To minimize $R(f)$ (test for new data)

First try to minimize $\hat{R}_s(f)$ (train by old data)

What's left is $\hat{R}(f) - \hat{R}_s(f)$ , this is known as generalization gap.

Generalization: How well does predictions "generalize" to new samples?

**Measuring Generalization:** Training/Test paradigm

In theory: Generalization bounds (based on "complexity" of the model)

In practice: empirical evaluation

Divide data into

> training set - a subset of data to train model
>
> testing set - a subset of data to test model

Ideally: only use testing set once (a few times)

**Supervised learning summary**

**Loss function:** what's the right loss function for the task? depends on the problems that one is trying to solve, and on the rest.

**Representation:** what class of functions should we use?

Also known as the inductive bias?

No free lunch theorem: no model can do well on every task.

"All models are wrong, but some are useful", George Box.

**Optimization:** how to solve the ERM problems?

**Generalization:** will the predictions of our model transfer gracefully to unseen examples?

## Linear Regression

$n$ 阶矩阵 $A$ 可逆，则 $|A| \neq 0$ ，为非奇异矩阵；$r(A) = n$ 是满秩；列向量 $a_1, \cdots a_n$ 线性无关；对应的齐次方程组 $AX = 0$ 只有 $0$ 解；$A$ 的 $n$ 个特征值非 $0$ 。

$n$ 阶矩阵 $A$ 不可逆，则 $|A| = 0$ ，为奇异(singular)矩阵；$r(A) < n$ 不是满秩；列向量 $a_1, \cdots a_n$ 线性相关；对应的齐次方程组有非 $0$ 解；$A$ 的 $n$ 个特征值存在 $0$ 值。

House price prediction: the function loss

Squared error: $(y - f(x))^2$

Absolute error: $|y - f(x)|$

predicted price $=$ price_per_sqft $\times$ square footage $+$ fixed_expense

**Formal Setup**

Input: $\mathcal{X} \in \mathbb{R}^d$

Output: $y \in \mathbb{R}^d$

Training data: $S = \{(x_i, y_i), i = 1, \cdots, n\}$

Linear model: $f : R^l \rightarrow R$ with

$$\begin{aligned} f(x) &= w_0 + \sum_{i=1}^{d} w_i x_i \\ &= w_0 + w^T x \\ w &= [w_1, \cdots, w_d]^T \quad (weights, weight\ vectors) \\ bias &= w_0 \end{aligned}$$

For notational convenience:

$\tilde{x}$ apppend $1$ to each $x$ as first feature $\tilde{x} = [1\ x_1\ \cdots\ x_d]^T$

let $\tilde{w} = [w_0\ w_1\ \cdots w_d]^T$ represent all $d+1$

Model: $f(x) = \tilde{w}^T \tilde{x}$

**Goal**

Minimize total squared error:

$$\hat{R}_s(\tilde{w}) = \frac{1}{n} \sum_i (f(x_i) - y_i)^2 = \frac{1}{n} \sum_i (\tilde{x}_i^T \tilde{w} - y_i)^2$$

Def: Residual sum of squares:

$$R_{ss}(\tilde{w}) = n\hat{R}_s(\tilde{w}) = \sum_i (\tilde{x}_i^T \tilde{w} - y_i)^2$$

ERM: find $\tilde{w}^* = \arg\min R_{ss}(\tilde{w}), \tilde{w} \in \mathbb{R}^{d+1}$ known as least squares equation.

**Warmup**

1.Warmup: $d = 0$

$$\begin{aligned} R_{ss}(w_0) &= \sum_i (w_0 - y_i)^2 \\ &= n w_0^2 - 2(\sum_i y_i) w_0 + const \\ &= n(w_0 - \frac{1}{n} \sum y_i)^2 + const \end{aligned}$$

Completion of squares.

$$w_0^* = \frac{1}{n}\sum_i y_i \ (the\ average)$$

2.Warmup: $d = 1$

$$R_{ss}(\tilde{w}) = \sum_i (w_0 + w_1 x_i - y_i)^2$$

General approach? find stationary point (point with 0 gradient)

$$\frac{\partial R_{ss}(\tilde{w}_0)}{\partial \tilde{w}_0} = 0 \Rightarrow \sum_i (w_0 + w_1 x_i - y_i) = 0$$
$$\frac{\partial R_{ss}(\tilde{w}_1)}{\partial \tilde{w}_1} = 0 \Rightarrow \sum_i (w_0 + w_1 x_i - y_i) x_i = 0$$

a linear system:

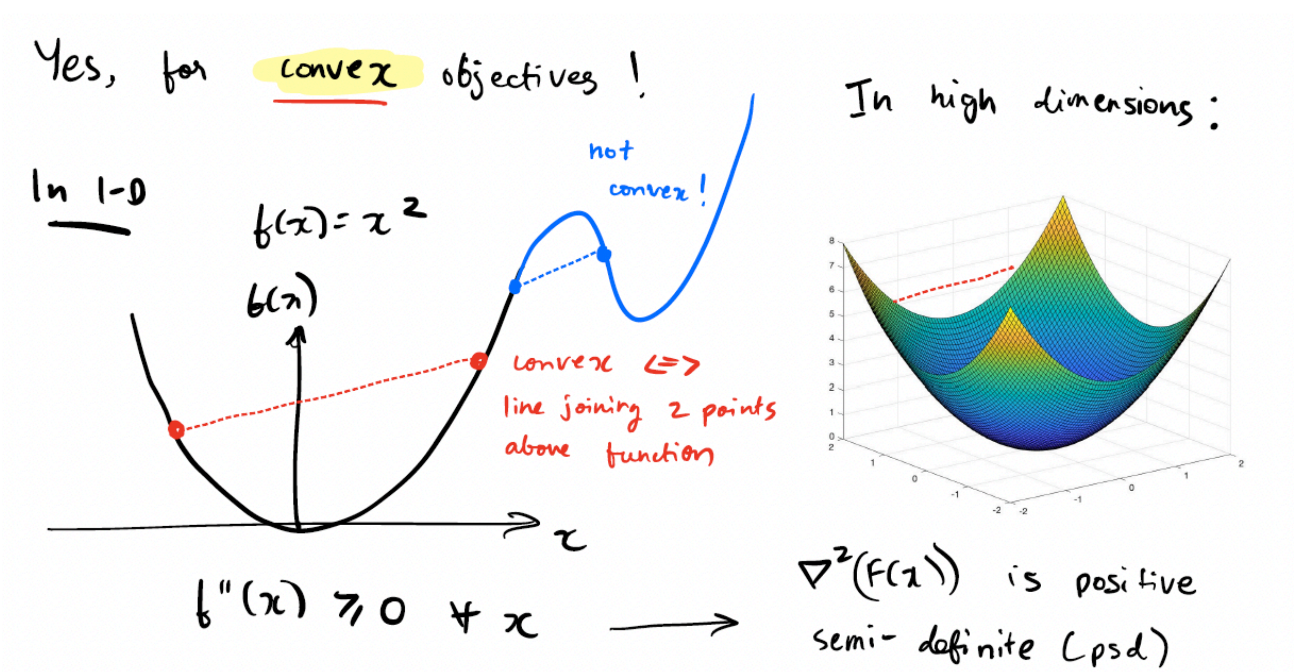
$$\begin{pmatrix} n & \sum x_i \\ \sum_i x_i & \sum x_i^2 \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} = \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{pmatrix}$$

solve:

$$\begin{pmatrix} w_0^* \\ w_i^* \end{pmatrix} = \begin{pmatrix} n & \sum x_i \\ \sum_i x_i & \sum_i x_i^2 \end{pmatrix}^{-1} \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{pmatrix}$$

assuming $\begin{pmatrix} n & \sum x_i \\ \sum_i x_i & \sum_i x_i^2 \end{pmatrix}$ is invertible.

**Attention:** for convex objectives, stationary points are minimizers.



If $f''(x) \geqslant 0$ , or we say $\nabla^2(F(x))$ is positive semi-definite (psd, $x^TAx \geqslant 0$), $f(x)$ is convex function.

**General least square solution**

$$\begin{aligned} R_{ss}(\tilde{w}) &= \sum_i (\tilde{x}_i^T \tilde{w} - y_i)^2 \\ Set\ \nabla R_{ss}(\tilde{w}) &= 0 \\ \nabla R_{ss}(\tilde{w}) &= 2\sum_i \tilde{x}_i(\tilde{x}_i^T \tilde{w} - y_i) \\ &\propto (\sum_i \tilde{x}_i \tilde{x}_i^T)\tilde{w} - \sum_i \tilde{x}_i y_i \\ &= (\tilde{X}^T\tilde{X})\tilde{w} - \tilde{X}^T y = 0 \end{aligned}$$

$$\begin{aligned} \tilde{x}_i &\in \mathbb{R}^{(d+1)\times 1} \\ \tilde{X} &= \begin{pmatrix} --x_1^T-- \\ \vdots \\ --x_n^T-- \end{pmatrix} \in \mathbb{R}^{n\times(d+1)} \\ y &= \begin{pmatrix} y_1^T \\ \vdots \\ y_n^T \end{pmatrix} \in \mathbb{R}^n \end{aligned}$$

$$\begin{aligned} (\tilde{X}^T\tilde{X})\tilde{w} &= \tilde{X}^T y \\ \therefore \tilde{w}^* &= (\tilde{X}^T\tilde{X})^{-1}X^T y \in \mathbb{R}^{(d+1)\times 1} \end{aligned}$$

$$\tilde{X}^T\tilde{X} = \begin{pmatrix} | & & | \\ \tilde{x}_1 & \cdots & \tilde{x}_n \\ | & & | \end{pmatrix} \begin{pmatrix} --x_1^T-- \\ \vdots \\ --x_n^T-- \end{pmatrix} = \sum_i \tilde{x}_i\tilde{x}_i^T \in \mathbb{R}^{(d+1)\times(d+1)}$$

suppose each feature is $0$-mean. covariance matrix: $\tilde{X}^T\tilde{X}_{(d+1)\times(d+1)}$

suppose $\tilde{X}^T\tilde{X} = I, then\ \tilde{w}^* = \tilde{X}^T y$

each weight is the covariance of the feature with label $y$ .

$$w^* = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T y$$

to invert $\tilde{X}^T\tilde{X} \in \mathbb{R}^{(d+1)} \times \mathbb{R}^{(d+1)}$ . ( Takes time $O(d^3)$, sometimes hard to invert with complex data)

$$\begin{aligned} y_i &= x_i^T w = w^T x_i \\ y_{n\times 1} &= \tilde{X}_{n\times(d+1)} w_{(d+1)\times 1} = \begin{pmatrix} \tilde{x}_1^T w \\ \vdots \\ \tilde{x}_n^T w \end{pmatrix} \end{aligned}$$

Some useful induction:

$$||A||_2^2 = A^T A$$
$$(AB)^T = B^T A^T$$
$$a^T b = b^T a$$
$$\frac{\partial\, x^T A x}{\partial\, x} = (A + A^T)x$$
$$\frac{\partial\, w^T x}{\partial\, x} = w$$

So we have another approach:

$$\begin{aligned} R_{ss} &= \sum_i (\tilde{x}_i^T \tilde{w} - y_i)^2 = ||\tilde{X}\tilde{w} - y||_2^2 \\ &= (\tilde{X}\tilde{w} - y)^T (\tilde{X}\tilde{w} - y) \\ &= \tilde{w}^T \tilde{X}^T \tilde{X}\tilde{w} - y^T \tilde{X}\tilde{w} - \tilde{w}^T \tilde{X}^T y + cnt \end{aligned}$$
$$\nabla_w R_{ss} = (\tilde{X}^T \tilde{X} + (\tilde{X}^T \tilde{X})^T)\tilde{w} - \tilde{X}^T y - \tilde{X}^T y = 0$$
$$\therefore w^* = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T y$$

## Optimization methods

GD(Gradient Descent): simple and fundamental

SGD(Stochastic Gradient Descent): faster, effective for large-scale problems

GD: first-order methods

Keep moving in the negative gradient direction

$$\begin{aligned} & start\ with\ some\ w^{(0)} \\ & For\ t = 0\ to\ T: \\ & \quad w^{(t+1)} = w^{(t)} - \eta \nabla F(w^{(t)}) \\ & \quad t = t + 1 \end{aligned}$$

$\eta > 0$ is called step size (learning rate)

- in theory % should be set in terms of some parameters of $f$ .
- in practice we just try several small values.
- might need to be changing over iterations (think $f(w) = |w|$)
- adaptive and automatic step size tuning is an active research area.

Why GD?

Intuition: first-order taylor approximation

$$F(w) \approx F(w^{(t)}) + \nabla F(w^{(t)})(w - w^{(t)})$$
$$F(w^{(t+1)}) \approx F(w^{(t)}) - \eta |\nabla F(w^{(t)}|_2^2 \leqslant F(w^{(t)})$$

this is only an approximation, and can be invalid if step size is too large.

Convergence guarantees for GD

$$F(w^{(l)} - F(w^*)) \leqslant \varepsilon$$

for nonconvex objectives, guarantees exist:

How close is $w^{(t)}$ as an approximate stationary point

$$||\nabla F(w^{(t)})|| \leqslant \varepsilon$$

if it is convex, optimization is unique. That means stationary point = global minimizer