南方科技大学

MAT8034: Machine Learning

Supervised Learning: Linear Regression

Fang Kong

https://fangkongx.github.io/Teaching/MAT8034/Spring2025/index.html

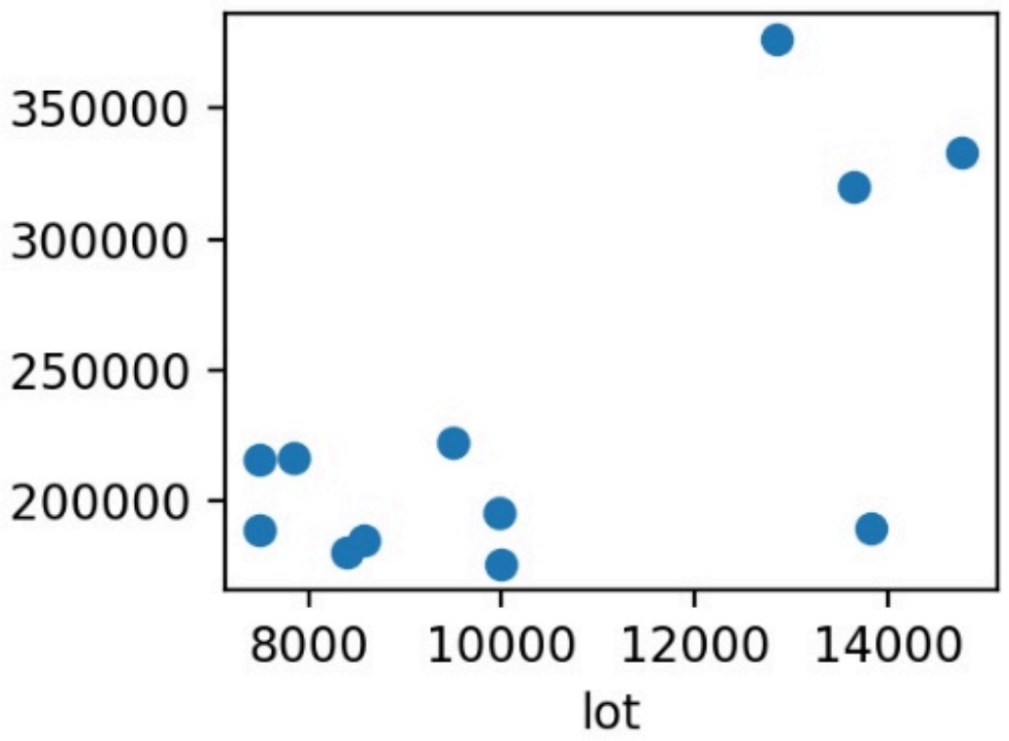
Part of slide credit: Stanford CS229

Outline

- Definitions
- Batch & Stochastic Gradient
- Normal Equations
- Probabilistic interpretation

Example: House pricing

	SalePrice	Lot.Area
4	189900	13830
5	195500	9978
9	189000	7500
10	175900	10000
12	180400	8402
22	216000	7500
36	376162	12858
47	320000	13650
55	216500	7851
56	185088	8577
58	222500	9505
59	333168	14774



Slightly Richer Example

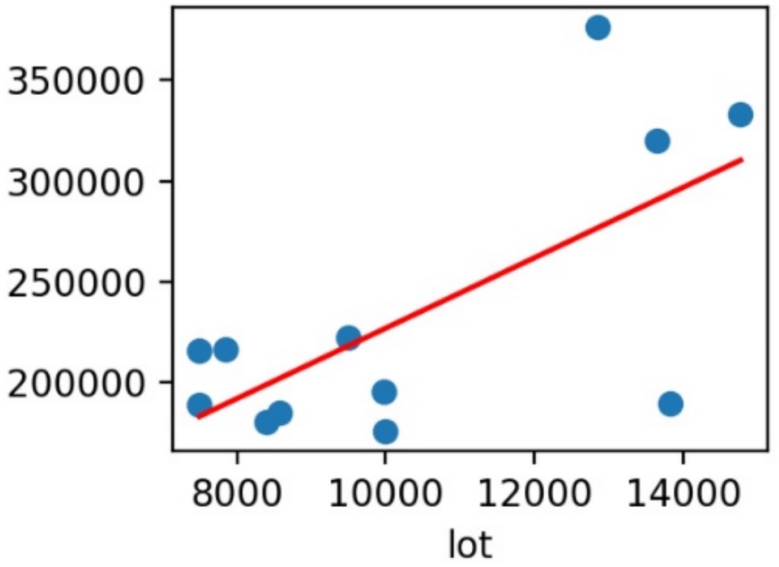
Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
$\vdots$	$\vdots$	$\vdots$

Definition

- Input $\mathcal{X}$ (house data), output $\mathcal{Y}$ (price)
- A hypothesis or a prediction function $h: \mathcal{X} \to \mathcal{Y}$
- A training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})\}$
  - $x^{(1)}$: $x_1^{(1)}$ represents the living area, $x_2^{(1)}$ represents the #bed room
- Given a training set, our goal is to produce a good function h
  - Will use h in the new data not in the training set

How to represent h?

- Simplest fit



- $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$
- Vector notation?

How to learn the parameter?

- Least-square cost function

$$J_\theta = \frac{1}{2}\sum_{i=1}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

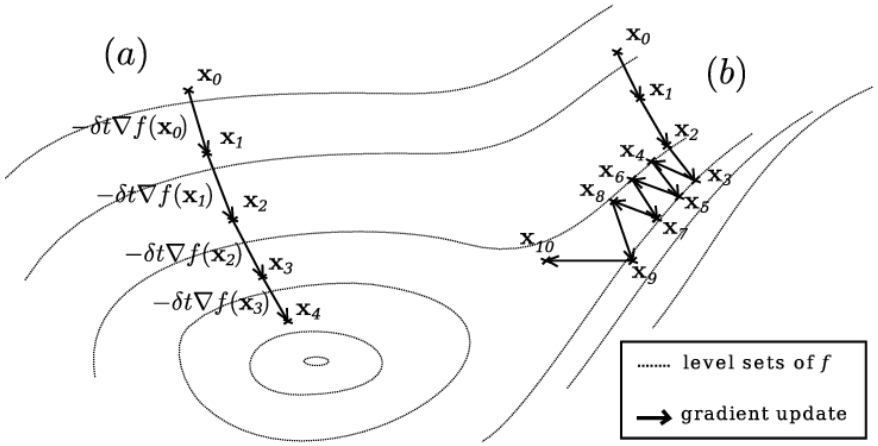
Least Mean Square Algorithm

- Start from an initial θ, then repeatedly change to make J_θ smaller

$$\theta^{(0)} = 0$$

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \alpha \frac{\partial}{\partial \theta_j} J(\theta^{(t)}) \qquad \text{for } j = 0, \ldots, d.$$

- Compute the derivatives…



https://ludovicarnold.com/teaching/optimization/gradient-descent/

Least Mean Square Algorithm

- Thus the update rule can be written as

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \alpha \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}.$$

We write this in *vector notation* for $j = 0, \ldots, d$ as:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)}.$$

Batch gradient descent

- Consider the update rule

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)}.$$

- Repeat until converge

Batch & stochastic gradient descent

- Consider the update rule $\theta^{(t+1)} = \theta^{(t)} - \alpha \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)}.$
- Repeat until converge

- A single update, we examine all data points
- In some modern applications, n may be in the billions or trillions!
  - E.g., we try to "predict" every word on the web
- Idea: Sample a few points (maybe even just one!) to approximate the gradient called Stochastic Gradient (SGD).
  - SGD is the workhorse of modern ML, e.g., pytorch & tensorflow

Stochastic minibatch

- We randomly select a **batch** of $B \subseteq \{1, \ldots, n\}$ where $|B| < n$.
- We approximate the gradient using just those B points as follows (vs. gradient descent)

$$\frac{1}{|B|} \sum_{j \in B} \left(h_\theta(x^{(j)}) - y^{(j)} \right) x^{(j)} \text{ v.s. } \frac{1}{n} \sum_{j=1}^{n} \left(h_\theta(x^{(j)}) - y^{(j)} \right) x^{(j)}.$$

- So our update rule for SGD is:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha_B \sum_{j \in B} \left(h_\theta(x^{(j)}) - y^{(j)} \right) x^{(j)}.$$

- NB: scaling of $|B|$ versus n is "hidden" inside choice of α_B.

Stochastic minibatch v.s. Gradient descent

- Recall our rule B points as follows:
- $$\theta^{(t+1)} = \theta^{(t)} - \alpha_B \sum_{j \in B} \left(h_\theta(x^{(j)}) - y^{(j)} \right) x^{(j)}.$$
- If $|B| = \{1, \ldots, n\}$ (the whole set), then they coincide.
- Smaller B implies a lower quality approximation of the gradient (higher variance).
- Nevertheless, it may actually converge faster! (Case where the dataset has many copies of the same point–extreme, but lots of redundancy)

Section summary

- Our goal was to optimize a loss function to find a good predictor
- We learned about gradient descent and the workhorse algorithm for ML, Stochastic Gradient Descent (SGD)
- We touched on the tradeoffs of choosing the right batch size

Normal Equations

Motivation

- Solve the least square exactly!

The matrix form

$$X = \begin{bmatrix} — & (x^{(1)})^T & — \\ — & (x^{(2)})^T & — \\ & \vdots & \\ — & (x^{(n)})^T & — \end{bmatrix} \qquad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

$$\begin{aligned} X\theta - \vec{y} &= \begin{bmatrix} (x^{(1)})^T\theta \\ \vdots \\ (x^{(n)})^T\theta \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix} \\ &= \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ \vdots \\ h_\theta(x^{(n)}) - y^{(n)} \end{bmatrix}. \end{aligned}$$

$$\begin{aligned} \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) &= \frac{1}{2}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2 \\ &= J(\theta) \end{aligned}$$

Normal equation

- Hope to minimize $J(\theta)$, find θ such that $\nabla J(\theta) = 0$

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \frac{1}{2}(X\theta - \vec{y})^T (X\theta - \vec{y}) \\
&= \frac{1}{2}\nabla_\theta \left((X\theta)^T X\theta - (X\theta)^T \vec{y} - \vec{y}^T (X\theta) + \vec{y}^T \vec{y}\right) \\
&= \frac{1}{2}\nabla_\theta \left(\theta^T (X^T X)\theta - \vec{y}^T (X\theta) - \vec{y}^T (X\theta)\right) \\
&= \frac{1}{2}\nabla_\theta \left(\theta^T (X^T X)\theta - 2(X^T \vec{y})^T \theta\right) \\
&= \frac{1}{2}\left(2X^T X\theta - 2X^T \vec{y}\right) \\
&= X^T X\theta - X^T \vec{y}
\end{aligned}
$$

Some useful facts:

$$a^T b = b^T a$$

$$\nabla_x b^T x = b$$

$$\nabla_x x^T A x = 2Ax$$

$$\theta = (X^T X)^{-1} X^T \vec{y}.$$

Probabilistic interpretation

A Justification for Least Squares?

- **Given** a training set $\{(x^{(i)}, y^{(i)})$ for $i = 1, \ldots, n\}$ in which $x^{(i)} \in \mathbb{R}^{d+1}$ and $y^{(i)} \in \mathbb{R}$.
- **Do** find $\theta \in \mathbb{R}^{d+1}$ s.t. $\theta = \text{argmin}_\theta \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)})^2$ in which $h_\theta(x) = \theta^T x$.

Where did this model come from?

One way to view is via a *probabilistic interpretation (helpful throughout the course)*.

A Justification for Least Squares?

We make an assumption (common in statistics) that the data are *generated* according to some model (that may contain random choices). That is,

$$y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}.$$

Here, $\varepsilon^{(i)}$ is a random variable that captures "noise" that is, unmodeled effects, measurement errors, etc.

Please keep in mind: this is just a model! As they say, all models are wrong but some models are *useful.* This model has been *shockingly* useful.

What do we expect of the noise?

What properties should we expect from $\varepsilon^{(i)}$

$$y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}.$$

Again, it's a model and $\varepsilon^{(i)}$ is a random variable:

- $\mathbb{E}[\varepsilon^{(i)}] = 0$ – the noise is unbiased.
- The errors for different points are *independent* and *identically distributed* (called, **iid**)

$$\mathbb{E}[\varepsilon^{(i)}\varepsilon^{(j)}] = \mathbb{E}[\varepsilon^{(i)}]\mathbb{E}[\varepsilon^{(j)}] \text{ for } i \neq j.$$

and

$$\mathbb{E}\left[\left(\varepsilon^{(i)}\right)^2\right] = \sigma^2$$

Here σ^2 is some measure of *how noisy* the data are. Turns out, this effectively defines the *Gaussian or Normal distribution.*

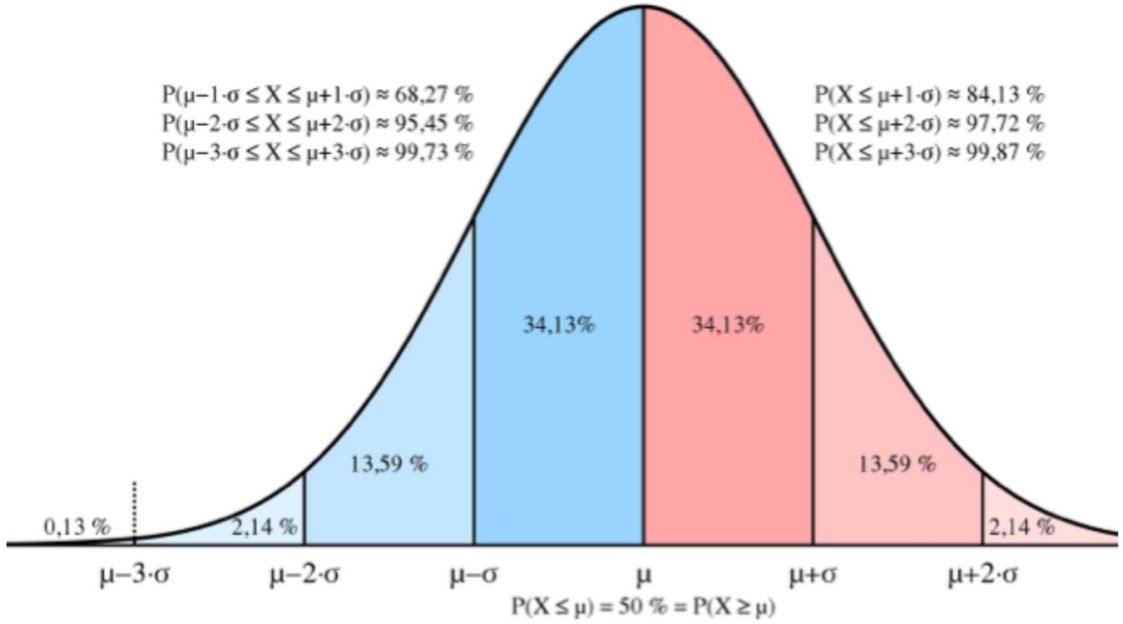
Gaussian distribution

We write $z \sim \mathcal{N}(\mu, \sigma^2)$ and read these symbols as

> *z is distributed as a normal with mean μ and standard deviation σ^2.*

or equivalently

$$P(z) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(z-\mu)^2}{2\sigma^2}\right\}.$$



Distribution of y

Recall in our model,

$$y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)} \text{ in which } \varepsilon^{(i)} \sim \mathcal{N}(0, \sigma^2).$$

or more compactly notation:

$$y^{(i)} \mid x^{(i)}; \theta \sim \mathcal{N}(\theta^T x, \sigma^2).$$

equivalently,

$$P\left(y^{(i)} \mid x^{(i)}; \theta\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(y^{(i)} - x^T\theta)^2}{2\sigma^2}\right\}$$

Likelihoods!

- Intuition: among many distributions, pick the one that agrees with the data the most (is most "likely")

$$L(\theta) = p(y|X;\theta) = \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)};\theta) \qquad \text{iid assumption}$$

$$= \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{ -\frac{(x^{(i)}\theta - y^{(i)})^2}{2\sigma^2} \right\}$$

Log Likelihoods!

- For convenience, use the Log Likelihood

$$\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \log \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2.
\end{aligned}$$

- Finding a θ that maximizes the log likelihood
  - What happens?
  - Equivalent to minimizing $\frac{1}{2} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2$

Takeaway

- "Under the hood," solving least squares is solving a maximum likelihood problem for a particular probabilistic model.
- Justify LMS as a very natural method (but not the only procedure)

- Worth noting: the choice does not depend on σ

Summary

- The regression problem for house pricing
- LMS
  - Gradient descent
  - Normal equation
- Justification for LMS
  - Log likelihood