

CS 331, Fall 2026  
Lecture 15 (10/19)

Today: — Continuous optimization  
— Statistical inference  
— Machine learning

## Continuous optimization (Part VI, Section 1)

This unit:

"decision variable"

↙

$$\min_{x \in X} f(x) \quad \text{or} \quad \max_{x \in X} f(x)$$

|                   |              |                           |
|-------------------|--------------|---------------------------|
| e.g.              | $f$          | $X$                       |
| Scheduling        | size         | non-overlapping intervals |
| MST               | total weight | spanning trees            |
| s-t shortest path | total weight | s-t paths                 |

Key difference: Now  $\underbrace{X}_{\text{continuous}} \subseteq \mathbb{R}^d$



- infinite sets
- typically cannot hope for exact only high-accuracy (some exceptions)

e.g.  $s$ - $t$  maxflow

$$\max_{x \in X} f(x)$$

$$f(x) = \sum_{e=(s,v) \in E} x_e$$

$$X = \left\{ x \in \mathbb{R}^E \mid \begin{array}{l} 0 \leq x_e \leq c_e \\ \sum_{e=(v,w)} x_e - \sum_{e=(u,v)} x_e = 0 \\ \forall v \in V \setminus \{s, t\} \end{array} \right\}$$

What are the rules of the game?

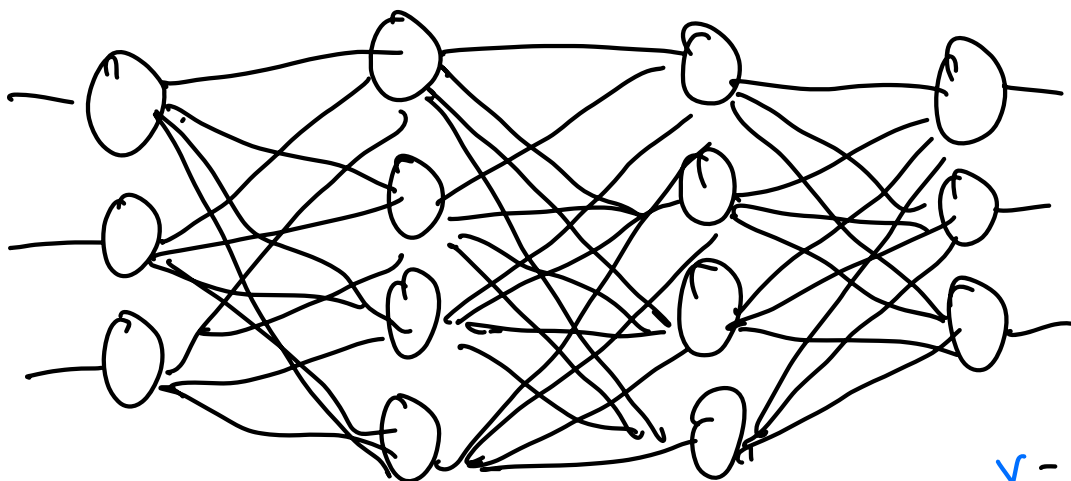
$$\min_{x \in X} f(x)$$

- Ability to evaluate  $f @ x$
- Ability to evaluate  $f' @ x$  reduces to

$$f'(x) \approx \frac{f(x+\delta) - f(x)}{\delta}$$

Sometimes, this is all that's reasonable.

$f(x)$  = average of neural network loss  
@ 1000000 pictures???



$x$  = neural net  
weights

However, this is not enough. Need structure

e.g. Can we compute  $\hat{x} \in \mathcal{X}$

$$\text{s.t. } f(\hat{x}) \leq \min_{x \in \mathcal{X}} f(x) + 0.99,$$

$$\mathcal{X} = [0,1], f: [0,1] \rightarrow [0,1]?$$

No! Let  $f(x) = \begin{cases} 0 & x = x^* \text{ (secret)} \\ 1 & \text{else} \end{cases}$

What structure helps? Next 3 lectures!

- Structured 1-d optimization
- Structured high-d optimization
- Dimensionality reduction

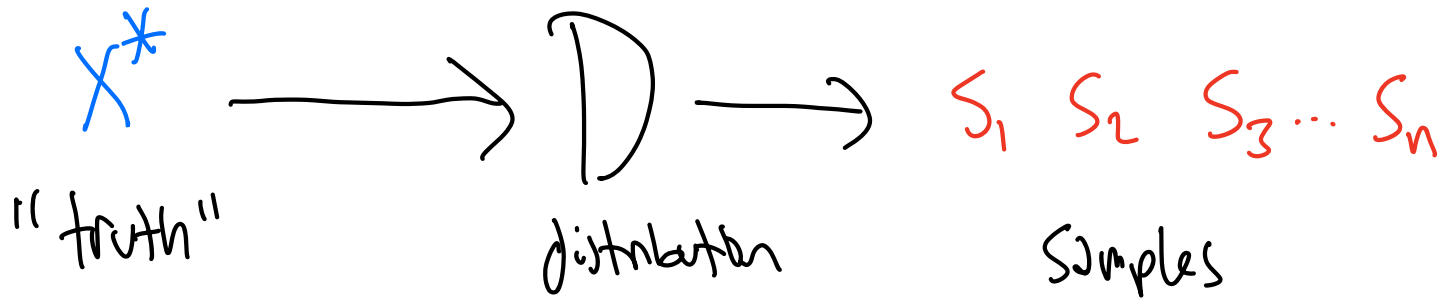
Today: Why continuous optimization?

Language of modern stats, ML

# Statistical inference (Part VI, Section 2.1)

---

Common situation:



Goal: infer  $x^*$  from  $s_1, s_2, s_3, \dots, s_n$

Something about data-generating  $D$       observed dataset

e.g.

treatment works?  
physical law  
weather  
election

Clinical trial  
expendent observations  
historical data  
polling outcomes

(LLMs) next word/sentence distribution

the internet

How to evaluate inference?

Loss function  $L$  maps guesses  $\rightarrow$  real values

Ex. We observe  $S_1 S_2 S_3 \dots S_n$

i.i.d. flips  $\sim \text{Bern}(p^*)$   $S = \begin{cases} 1 & \text{w.p. } p^* \\ 0 & \text{w.p. } 1-p^* \end{cases}$

How to evaluate guess  $\hat{p}$ ?

Reasonable first try:

$$L(\hat{p}) := |\hat{p} - p^*| \leftarrow$$

Problem: we can't measure this!

Depends on unknown  $p^*$

A useful loss can only depend on known quantities

$x^*$

perfect truth  
unknown  
not OK!

$S_1 S_2 \dots S_n$

noisy reflection  
known  
OK!

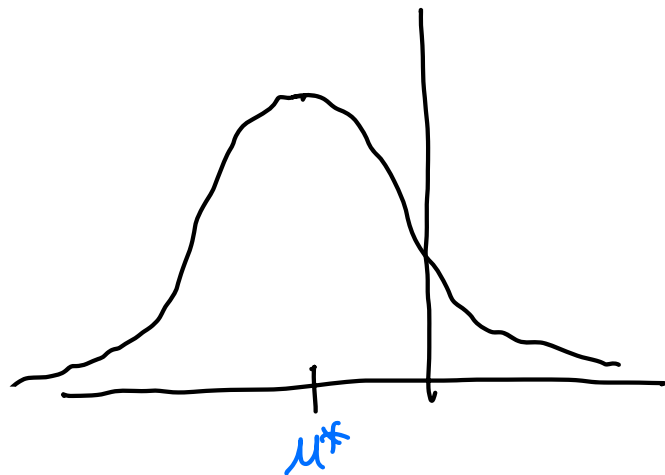
# Parametric statistics

Particularly clean case:  $\mathcal{D}$  completely parameterized  
by unknown  $\mu^* \in \mathbb{R}^d$ . (described fully)

Ex. 1-d Gaussian  $\mathcal{D} = \mathcal{N}(\underbrace{\mu^*}_{\text{mean}}, \underbrace{1}_{\text{variance}})$

density of  $S \sim \mathcal{D}$ :

$\propto \exp\left(-\frac{(S - \mu^*)^2}{2}\right)$   
"proportional to"



Ex. High-d Gaussian  $\mathcal{D} = \mathcal{N}(\mu^* \in \mathbb{R}^d, I_d)$

density of  $S \sim \mathcal{D}$ : ( $S_j \sim \mathcal{N}(\mu_j^*, 1) \forall j$  indep.)

$$\propto \exp\left(-\frac{\|S - \mu^*\|_2^2}{2}\right)$$

$$\|v\|_2^2 = v_1^2 + v_2^2 + \dots + v_d^2$$

Euclidean distance

Given samples  $s_1 \dots s_n \sim \mathcal{D}$  (parameterized by  $\mu^*$ )

How to estimate  $\mu^*$ ?

Idea: Maximum likelihood estimation

Real life  $\mu^* \rightarrow \mathcal{D} \rightarrow s_1 s_2 \dots s_n$   
"likelihood"

Our guess  $\hat{\mu} \rightarrow \mathcal{D} \rightarrow s_1 s_2 \dots s_n$

Likelihood of producing observations  $s_1 s_2 \dots s_n$

if true distribution was  $\mathcal{N}(\hat{\mu}, 1)$ :

$$\propto \prod_{i \in [n]} \exp\left(-\frac{(s_i - \hat{\mu})^2}{2}\right)$$

$$= \exp\left(-\frac{1}{2} \sum_{i \in [n]} (s_i - \hat{\mu})^2\right)$$

the higher  
the better!

Often more convenient to work w/

$\exp(- \dots)$  NLL, function of  $\hat{x}$  (guess)  
 $s_1 \dots s_n$  (samples)

Max likelihood  $\approx$  minimizing negative log-likelihood (loss)

① Write density of  $s_1 \dots s_n$  as likelihood function of guessed parameter  $\hat{x}$

② Use NLL as loss, return  $\arg\min_{\hat{x}} \text{NLL}$   
how? continuous algo!

Ex. 1-d Gaussian

$$L(\hat{x} | s_1 \dots s_n) = \sum_{i \in [n]} (\hat{x} - s_i)^2$$

Theory: how many samples needed for ... error?

More examples:

Linear regression

$$S_i = \begin{matrix} \text{features} & \text{label} \\ \text{in } \mathbb{R}^d & \text{in } \mathbb{R} \end{matrix} \begin{pmatrix} x_i & y_i \end{pmatrix}$$

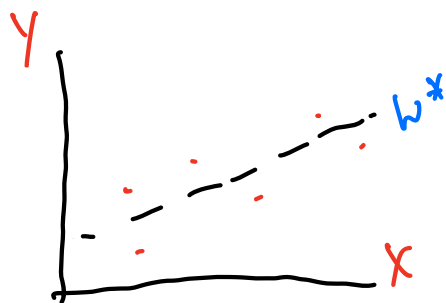
e.g.  $x_i = (\text{height, weight, salary, eye color, ...})$

$y_i = \text{Vertical jump}$

Statistical model:

$$\sum_{j \in (d)} w_j^* [x_i]_j$$

$w^* \rightarrow \mathcal{D} \rightarrow y_i = \langle w^*, x_i \rangle + \xi_i$



$\xi_i \sim \mathcal{N}(0, 1)$  (noise)

Single-sample

$$\propto \exp\left(-\frac{1}{2} (y_i - \langle w^*, x_i \rangle)^2\right)$$

MLE:

$$\underset{\hat{w}}{\text{argmin}} \frac{1}{2} \sum_{i \in (n)} (y_i - \langle \hat{w}, x_i \rangle)^2$$

Linear regression: Supervised (come w/ labels)

mean estimation: unsupervised

Another divide: Regression vs. Classification

with supervised

$y \in \mathbb{R}$

$y = 1, 2, 3, \dots$

Usually require link taking  $\mathbb{R} \rightarrow$  classes

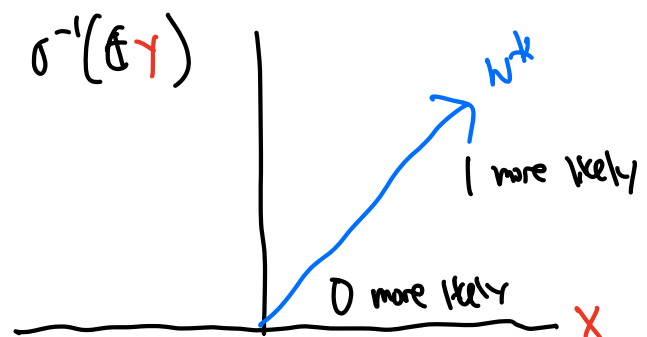
Ex. logistic regression ("generalized linear model")

$(x, y) \sim \mathcal{D}$  (parameter  $w^* \in \mathbb{R}^d$ )

$$y = \begin{cases} 1 & \text{w.p. } \sigma(\langle w^*, x \rangle) \\ 0 & \text{w.p. } 1 - \sigma(\langle w^*, x \rangle) \end{cases}$$

$$\sigma(t) = \frac{e^t}{1 + e^t} \in [0, 1]$$

$t \in \mathbb{R}$



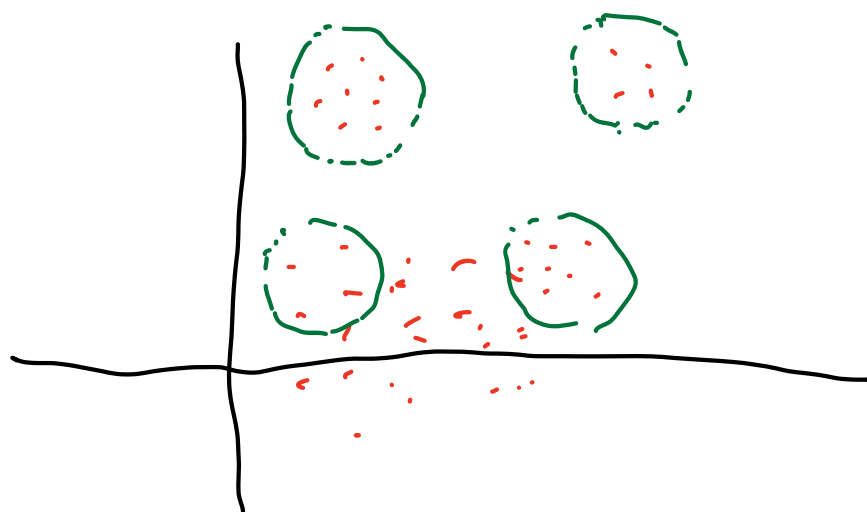
Related task: Clustering

$$X_i \sim D_j \quad j = 1, 2, \dots, k$$

"class"

Observe  $X_1, X_2, \dots, X_n$

Estimate  $j_1, j_2, \dots, j_n$  "clusters"



Inference:  
What "group" structure?

Despite being discrete, often helpful to parameterize  
data likelihood continuously (real life + easy to optimize)

Other inference tasks:

hypothesis testing, uncertainty quantification,

causal inference, experimental design, ...

# Machine Learning (Part VI, Section 2.2)

Cultural difference: supervised ML uses inference primarily as means to make future predictions

Evaluation: test-time loss

learning  $\neq$   
memorization

- receive  $\begin{matrix} (x_1, y_1) \\ (x_2, y_2) \\ \vdots \\ (x_n, y_n) \end{matrix} \sim \mathcal{D}$  (training data)

- evaluated on  $(x, ?) \sim \mathcal{D}$  (test part)

Optimization: agnostic settings

model as "good enough" to predict the truth

$x^* \rightarrow \mathcal{D} \rightarrow s_1 s_2 s_3 \dots s_n$  realizable

$x \approx \mathcal{D} \rightarrow s_1 s_2 s_3 \dots s_n$  agnostic

Nonetheless, many ideas / perspectives from  
statistical inference useful in ML!

## The recipe

$S_1 S_2 \dots S_T$   
training data

$S_{T+1} \dots S_n$   
holdout "eval" data

$S$   
future test

① Choose expressive parametric model

$p(\underbrace{x}_{\text{features}}; \underbrace{w}_{\text{learned parameters}}) \rightarrow \underbrace{\hat{y}}_{\text{predicted label}}$

② Choose loss  $L(\underbrace{p}_{\text{predicted}}, \underbrace{y}_{\text{target}})$  to evaluate predictions

e.g.  $1 \quad |p - y| > \frac{1}{2}$   
 $0 \quad |p - y| \leq \frac{1}{2}$   
"zero-one loss"

$(1 - y) \log \frac{1}{1 - p} + y \log \frac{1}{p}$   
"cross entropy"  
Smooth surrogate to zero-one

③ Train your model

$$\hat{w} \leftarrow \arg\min \sum_{i \in G^T} L(p(x_i; \hat{w}), y_i)$$

(How? Continuous optimization!)

④ Evaluate on holdout

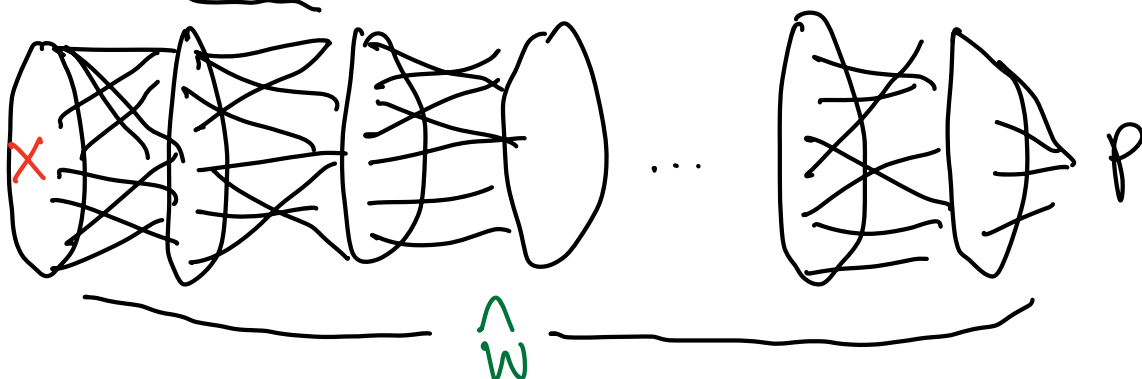
$$\frac{1}{n-t} \sum_{i \in \binom{[n]}{t}} L(p(x_i; \hat{w}), y_i) \approx \mathbb{E} \left[ L(p(x; \hat{w}), y) \right]$$

$(x, y)$  = fresh sample

Repeat & repeat until happy! (Careful: adaptivity)

Modern upgrades

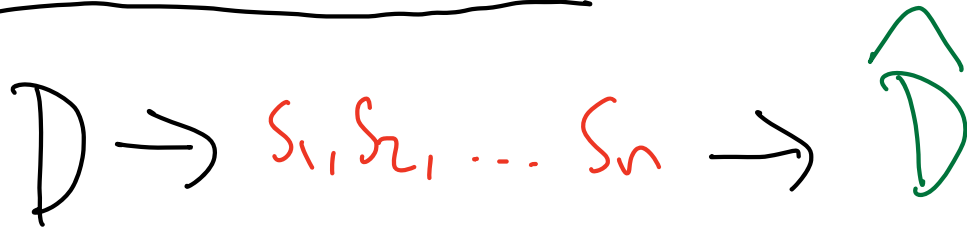
① = deep neural nets



③ = specialized optimizers (Adam, AdamW, Mean, ...)

Turns out recipe works in other paradigms...

## Generative models



e.g. cat pictures

"learned  $D$ "

produce new samples

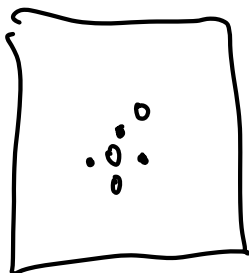
e.g. LLMs (autoregressive:  $p(\dots | \hat{w}, s_1, s_2, \dots, s_{n-1})$ )

diffusion (neural net  $p(\dots | \hat{w})$  used to learn trajectory for diff eq)

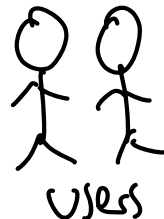
## Reinforcement learning



e.g.



game playing



RLHF

users