CSCI 567: Machine Learning

Vatsal Sharan

Spring 2026

Lecture 1, Jan 16

USC University of Southern California

Logistics

Course website: https://vatsalsharan.github.io/spring26.html
- Logistics, slides, homework etc.

Ed Discussion: https://edstem.org/
- Main forum for communication

Brightspace: https://brightspace.usc.edu/d2l/home/261815
- Recordings

Gradescope: https://www.gradescope.com/
- Homework submission

Prerequisites

This is a mathematically advanced and intensive class
(that makes it more interesting!)

(1) Undergraduate level training or coursework on linear algebra, (multivariate) calculus, and probability and statistics;
(2) Programming with Python;
(3) Undergraduate level training in the analysis of algorithms (e.g. runtime analysis).

Overview of logistics, **go through course website** for details:

Homeworks (30%): 4 homeworks (groups of 2), 3 late days per group (max 1 per HW)

Exams (50%): **3/6** and **5/1** during lecture time (1pm)

Project (20%): You can choose your topic, groups of 4, more details later

Note: Plagiarism and other unacceptable violations

- Neither ethical nor in your self-interest
- Zero-tolerance
- Read collaboration policy on course website

AI usage in homeworks

Why do we have homeworks?

- This class has many new mathematical and conceptual elements
- Absorbing them takes time
- Homework problems and exercises are chosen to give you the opportunity to get comfortable with these new concepts

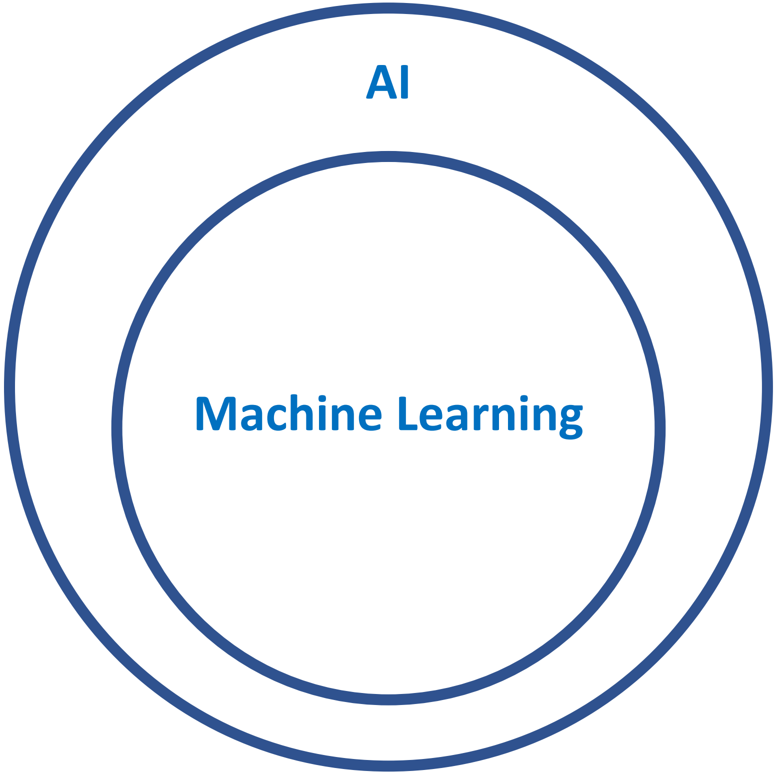
If you use AI to do your homework you are wasting the opportunity you have now to learn new concepts. Likely will not get such opportunities as easily in a job.

Therefore, our policy is to not allow AI usage for homeworks. (You can use it for the project, more on that later.)

If you need help:

- Come to Office Hours
- Post on Ed Discussion
- Discuss with your peers, reach out to the staff

Machine Learning



ML has been driving the recent advances in AI

What is ML?

"Humans appear to be able to learn new concepts without needing to be programmed explicitly in any conventional sense. In this paper we regard ***learning as the phenomenon of knowledge acquisition in the absence of explicit programming****."*

--- *A Theory of the Learnable*, 1984, Leslie Valiant

What is ML?

"Humans appear to be able to learn new concepts without needing to be programmed explicitly in any conventional sense. In this paper we regard ***learning as the phenomenon of knowledge acquisition in the absence of explicit programming****."*

--- *A Theory of the Learnable*, 1984, Leslie Valiant

"A computer program is said to ***learn*** *from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."*

--- *Machine Learning*, 1998, Tom Mitchell

My slides from Fall 2022 & Spring 24 motivating ML..

Enormous advances in recent years

The New York Times

THE SHIFT

We Need to Talk About How Good A.I. Is Getting

We're in a golden age of progress in artificial intelligence. It's time to start taking its potential and risks seriously.

Give this article 608

DALL-E 2's output when given input "infinite joy"

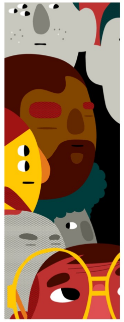
New York Times, August 24, 2022

Text generation: GPT-3

The New York Times

Meet GPT-3. It Has Learned to Code (and Blog and Argue).

The latest natural-language system generates tweets, pens poetry, summarizes emails, answers trivia questions, translates languages and even writes its own computer programs.



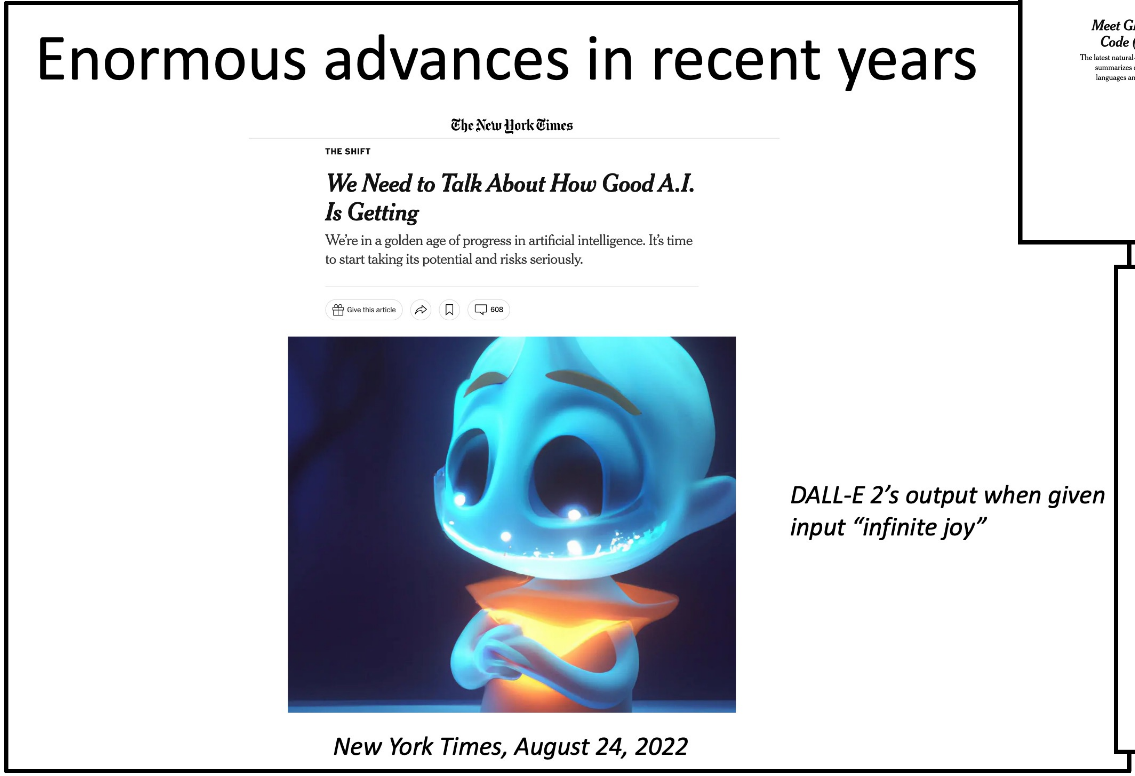
The New York Times

Meet DALL-E, the A.I. That Draws Anything at Your Command

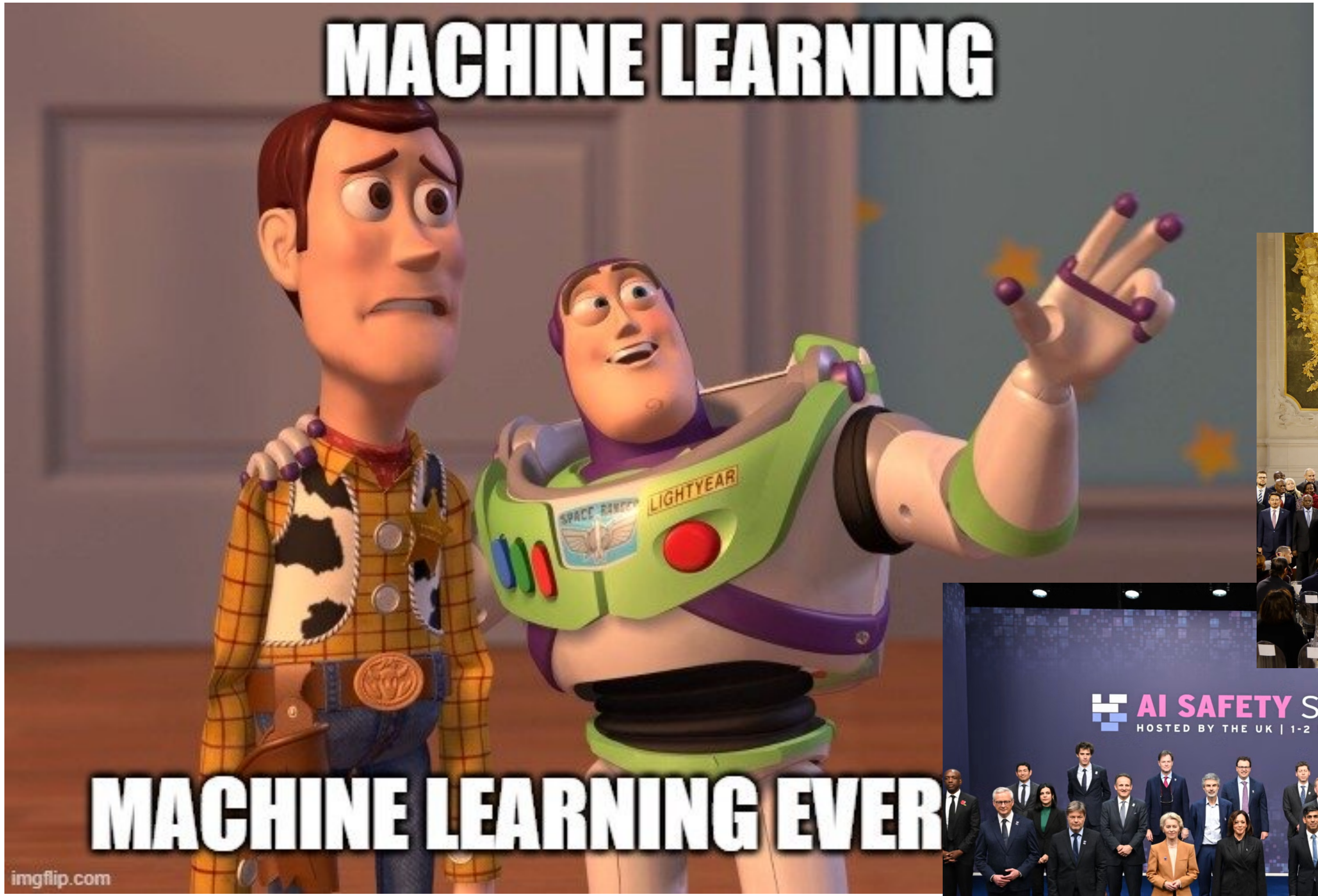
New technology that blends language and images could serve graphic artists — and speed disinformation campaigns.

Give this article 145

My slides from Fall 2022 & Spring 24 motivating ML..



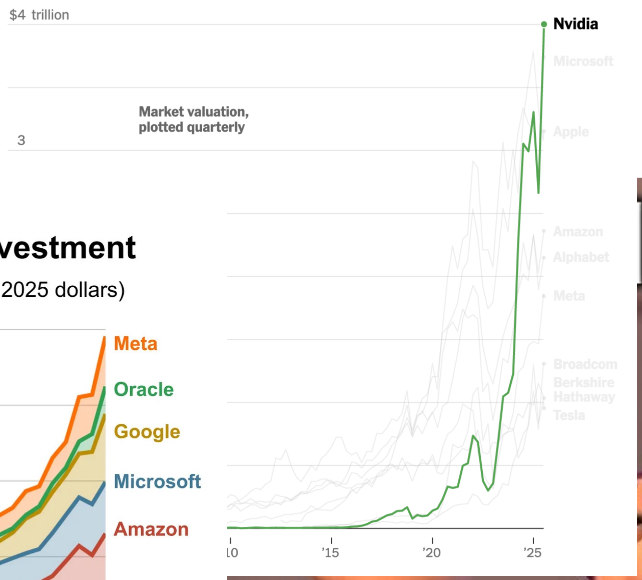
Now


MACHINE LEARNING
LIGHTYEAR
MACHINE LEARNING EVER
imgflip.com

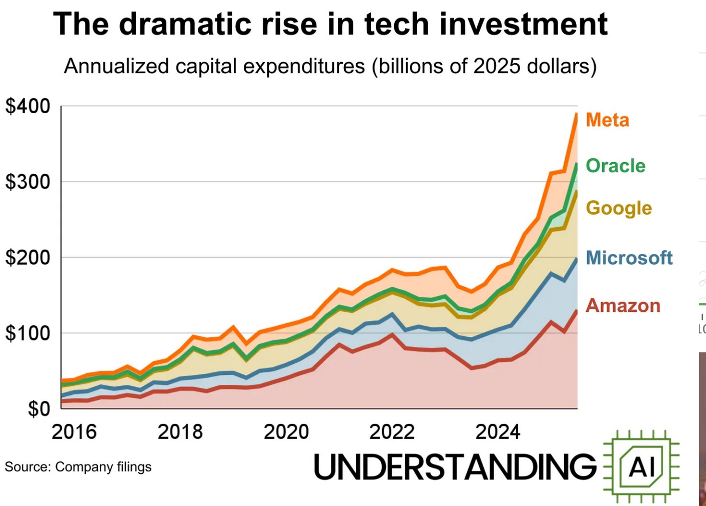

SOMMET POUR L'ACTION SUR L'IA


AI SAFETY SUMMIT
HOSTED BY THE UK | 1-2 NOVEMBER 2023

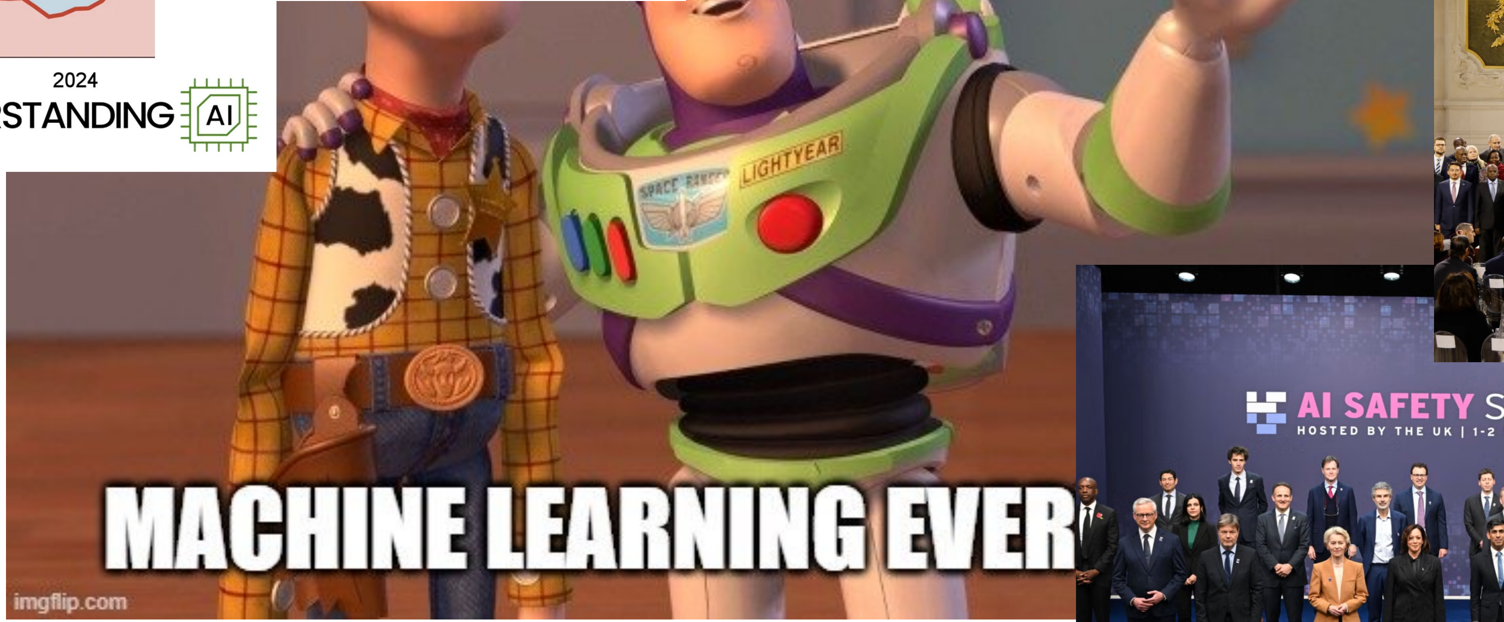
Now



$4 trillion
Nvidia
Microsoft
Market valuation,
plotted quarterly
3
Apple
Amazon
Alphabet
Meta
Broadcom
Berkshire
Hathaway
Tesla
'10
'15
'20
'25



The dramatic rise in tech investment
Annualized capital expenditures (billions of 2025 dollars)
$400
$300
$200
$100
$0
2016
2018
2020
2022
2024
Meta
Oracle
Google
Microsoft
Amazon
Source: Company filings
UNDERSTANDING AI

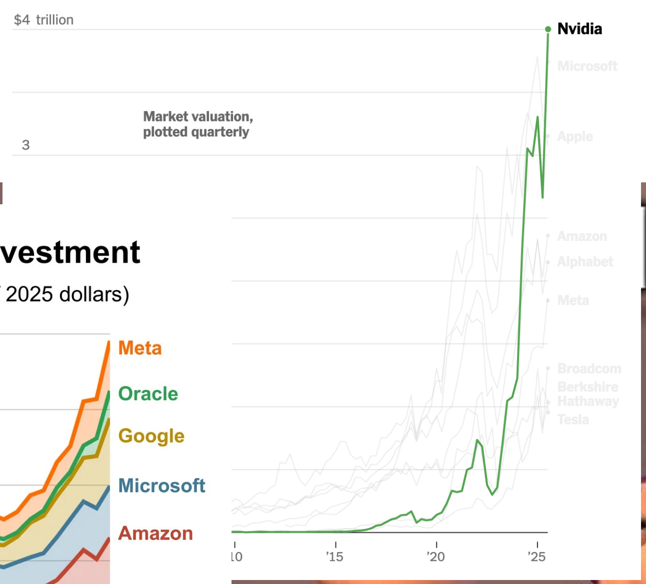


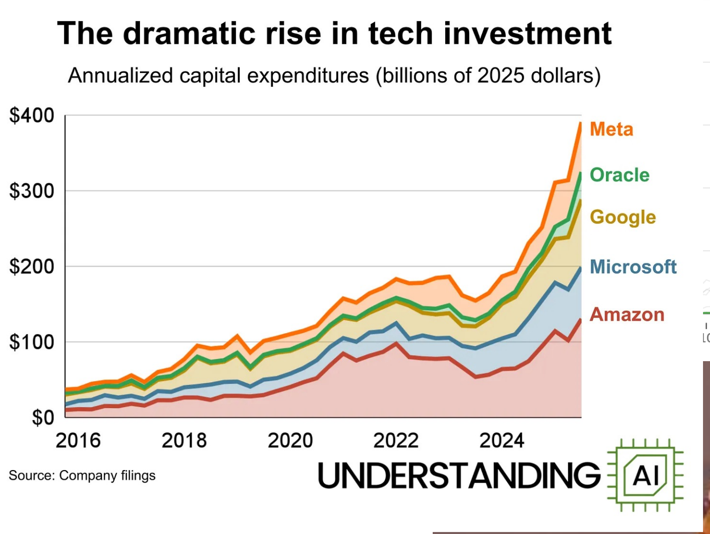
E LEARNING
LIGHTYEAR
MACHINE LEARNING EVER
imgflip.com

SOMMET
POUR L'ACTION
SUR L'IA

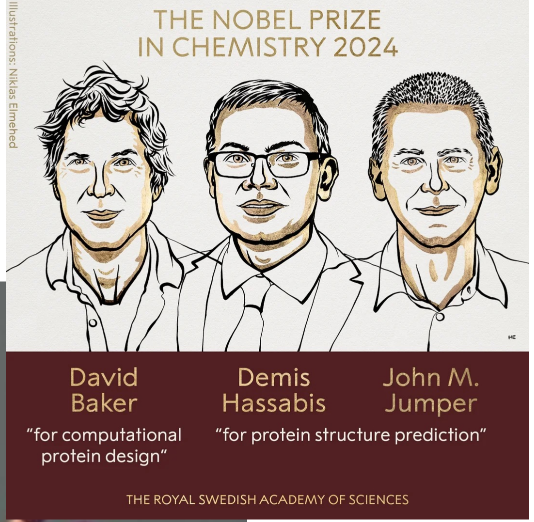
AI SAFETY SUMMIT
HOSTED BY THE UK | 1-2 NOVEMBER 2023

Now


$4 trillion
3
Market valuation, plotted quarterly
Nvidia
Microsoft
Apple
Amazon
Alphabet
Meta
Broadcom
Berkshire Hathaway
Tesla
'10
'15
'20
'25

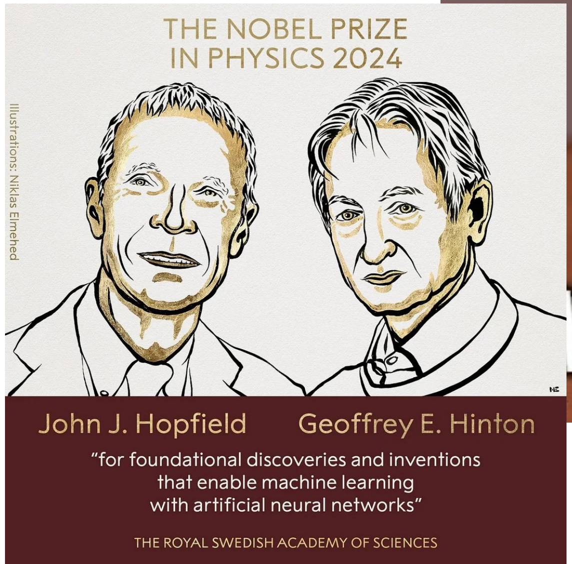

The dramatic rise in tech investment
Annualized capital expenditures (billions of 2025 dollars)
$400
$300
$200
$100
$0
2016
2018
2020
2022
2024
Meta
Oracle
Google
Microsoft
Amazon
Source: Company filings
UNDERSTANDING AI


E LEARNING
LIGHTYEAR
MACHINE LEARNING EVER


THE NOBEL PRIZE IN CHEMISTRY 2024
Illustrations: Niklas Elmehed
David Baker
"for computational protein design"
Demis Hassabis
John M. Jumper
"for protein structure prediction"
THE ROYAL SWEDISH ACADEMY OF SCIENCES


SOMMET POUR L'ACTION SUR L'IA


AI SAFETY SUMMIT
HOSTED BY THE UK | 1-2 NOVEMBER 2023

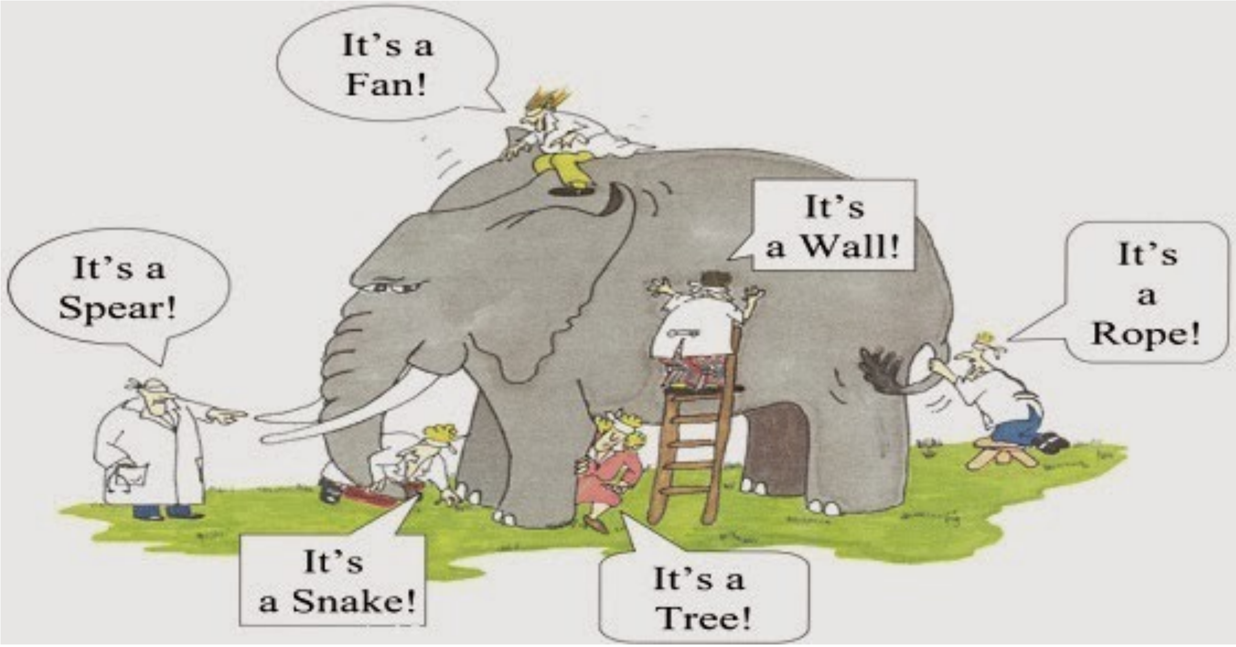

THE NOBEL PRIZE IN PHYSICS 2024
Illustrations: Niklas Elmehed
John J. Hopfield
Geoffrey E. Hinton
"for foundational discoveries and inventions that enable machine learning with artificial neural networks"
THE ROYAL SWEDISH ACADEMY OF SCIENCES

What do you find exciting (or not exciting?) about the advances?

Rapid progress, but a lot needs to be done..

- Require significant computational resources
- Lack of understanding
- Fairness
- Robustness
- Interpretability
- Privacy
- Alignment
- …

Machine learning can be *brittle*



The Blind Men and the Elephant

It was six men of Indostan
To learning much inclined,
Who went to see the Elephant
(Though all of them were blind),
That each by observation
Might satisfy his mind.

The First approached the Elephant,
And happening to fall
Against his broad and sturdy side,
At once began to bawl:
"God bless me! but the Elephant
Is very like a WALL!"

....

This class:

- Understand the fundamentals
- Understand when ML works, its limitations, think critically

This class:

- Understand the fundamentals
- Understand when ML works, its limitations, think critically

In particular,

- Study fundamental statistical ML methods (supervised learning, unsupervised learning, etc.)
- Solidify your knowledge with hands-on programming tasks
- Prepare you for studying advanced machine learning techniques

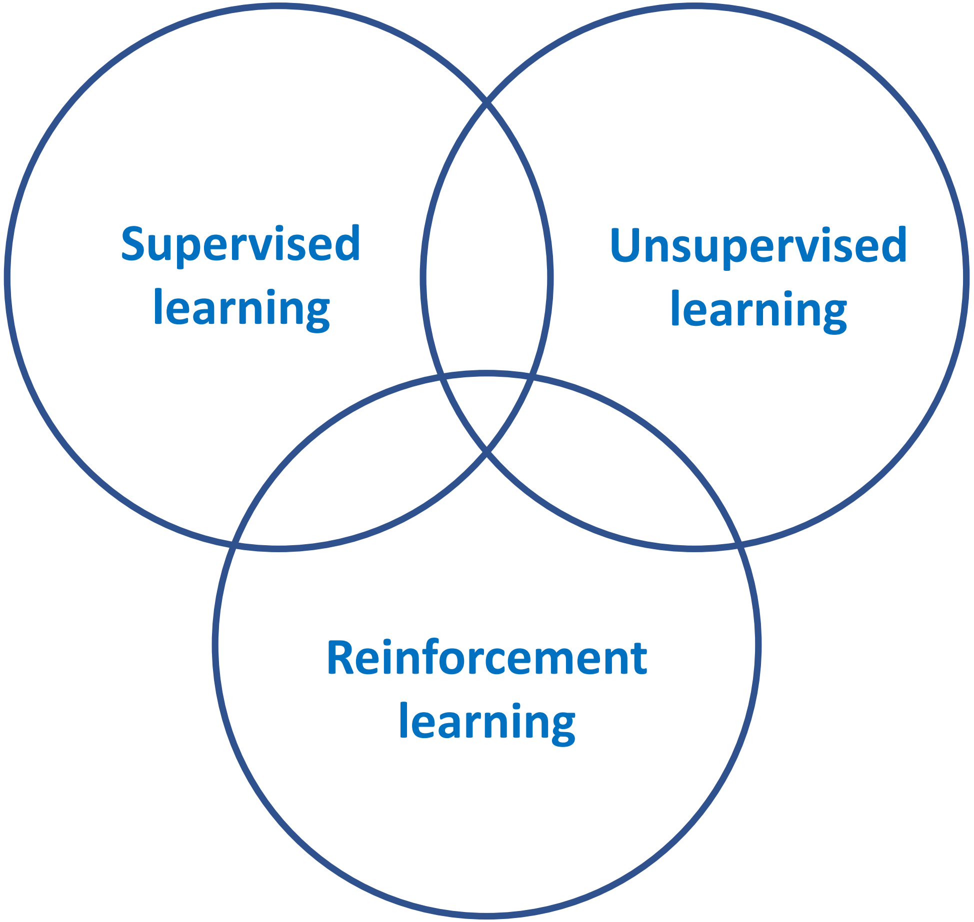
A simplistic taxonomy of ML

Supervised learning:
Aim to predict outputs of future datapoints

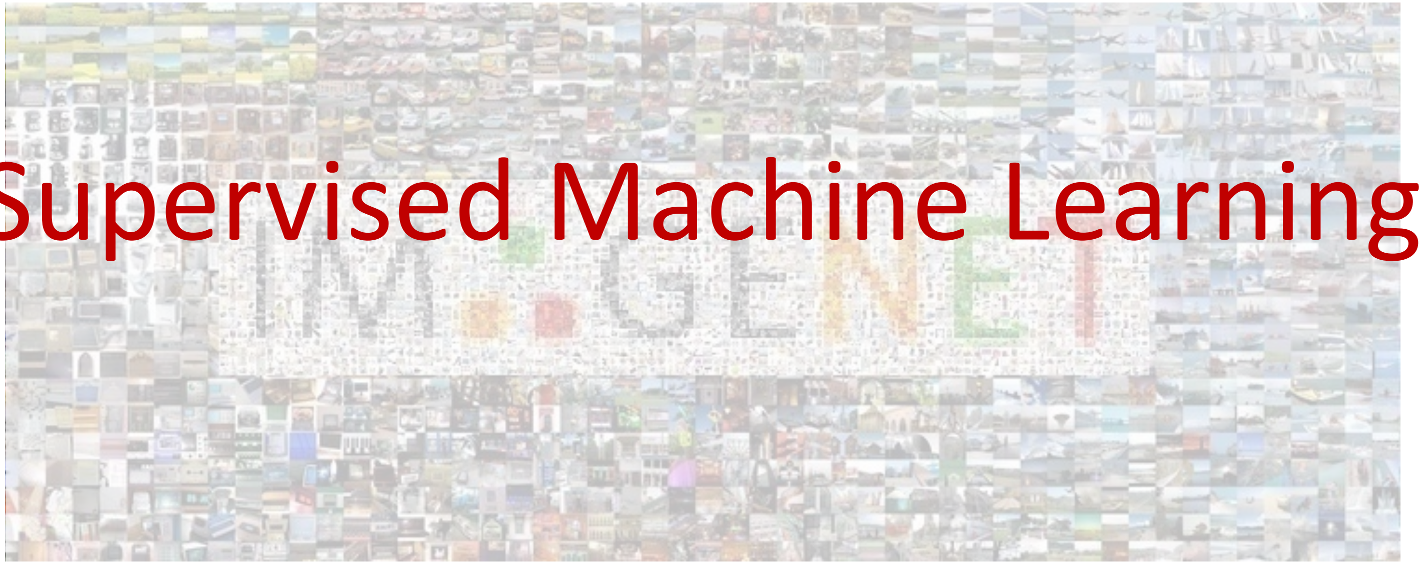
Unsupervised learning:
Aim to discover hidden patterns and explore data

Reinforcement learning:
Aim to make sequential decisions

A simplistic taxonomy of ML



Supervised Machine Learning



Supervised ML: Predict future outcomes using past outcomes

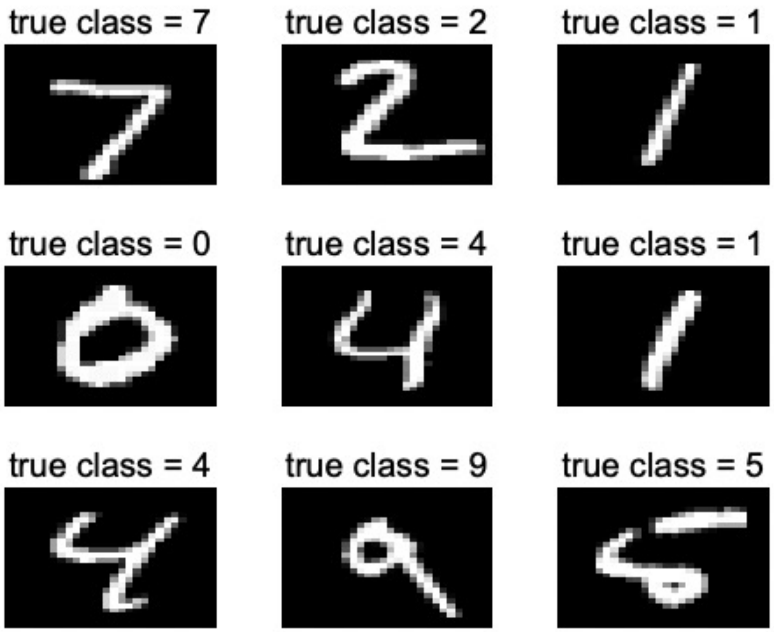
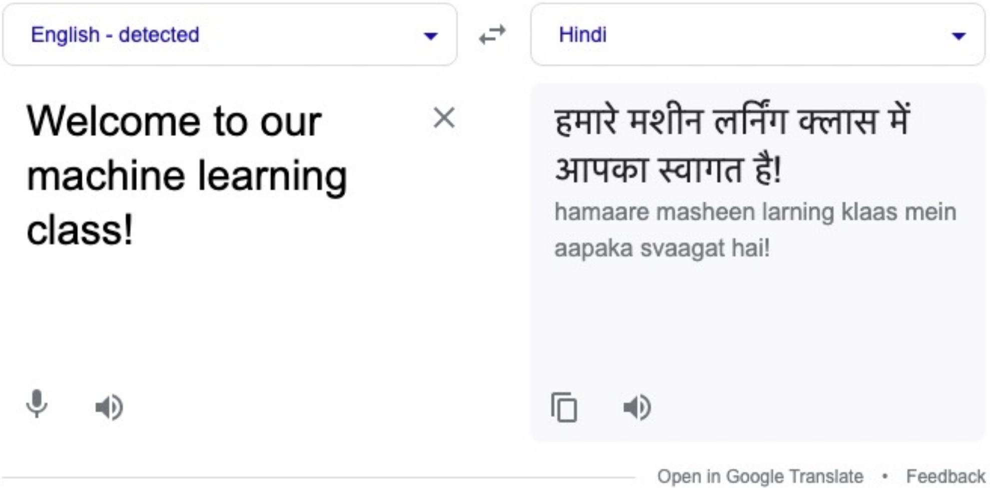
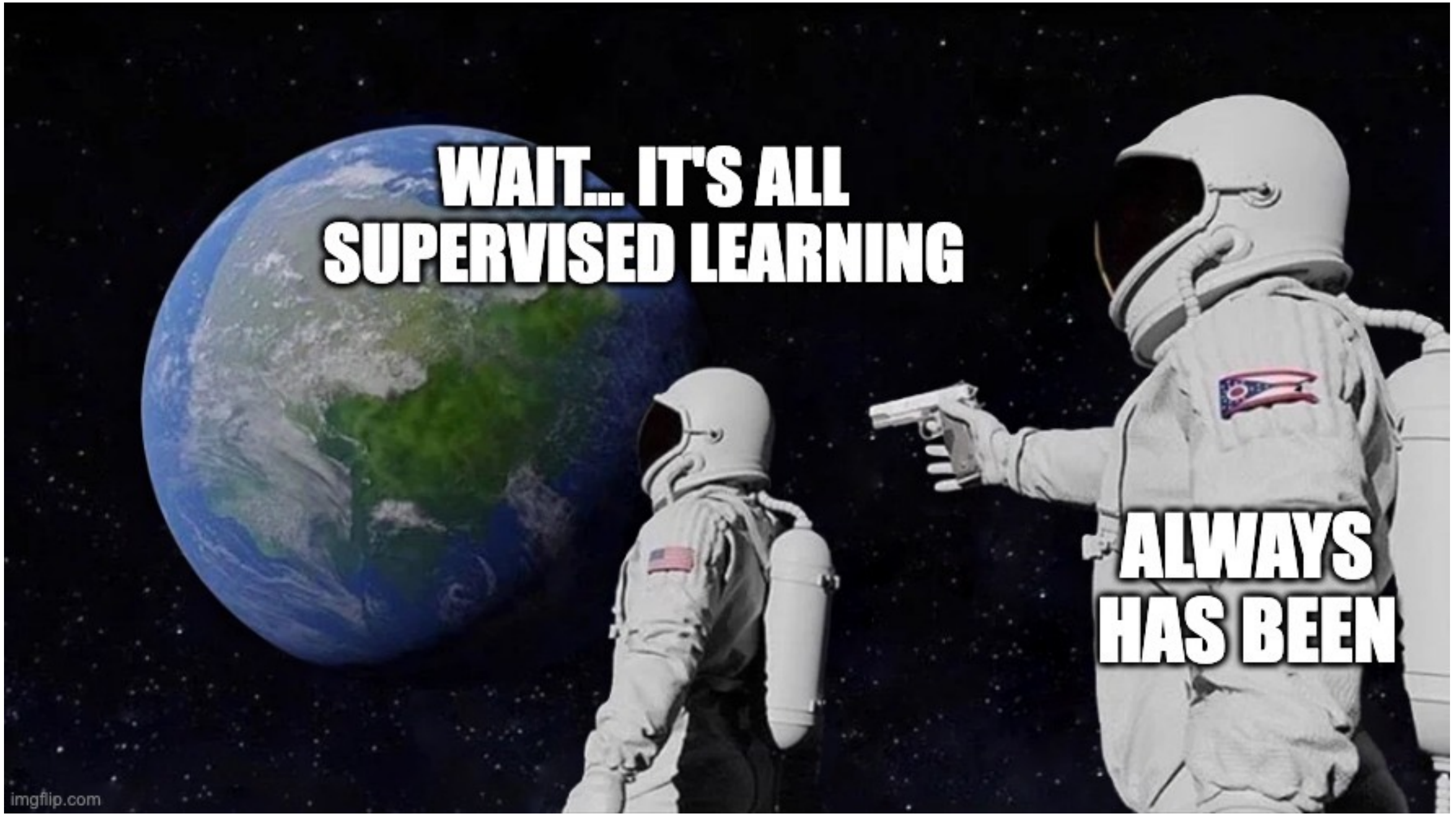


Image classification

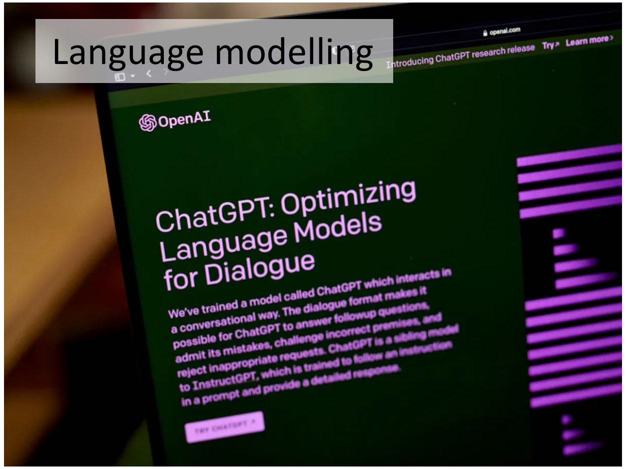


Machine translation

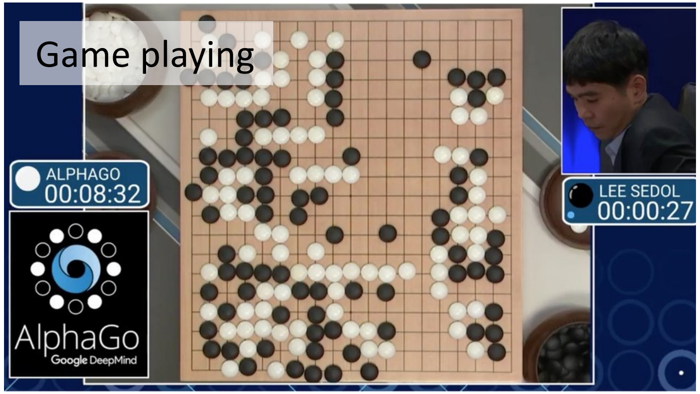
Supervised ML is at the heart of many AI advances



Supervised ML is at the heart of many AI advances



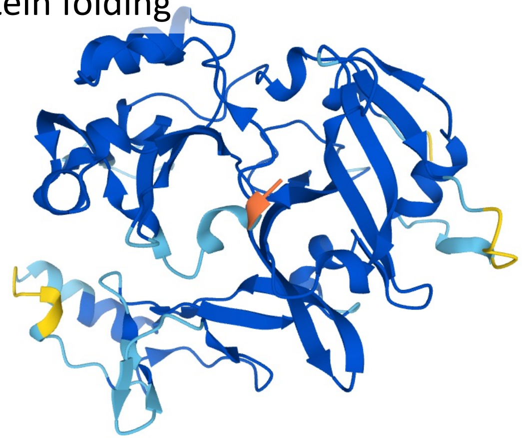
Language modelling

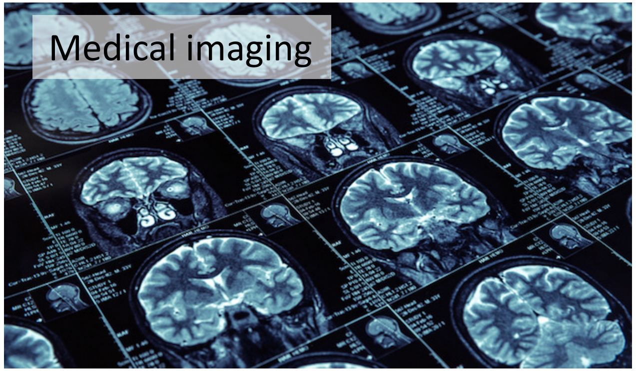


Game playing

Generative modelling

Protein folding



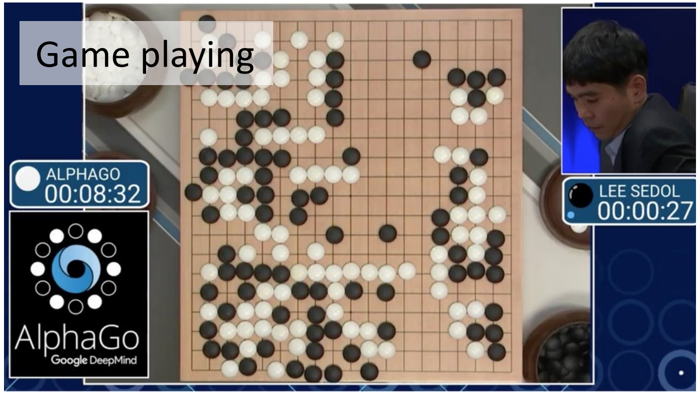


Medical imaging

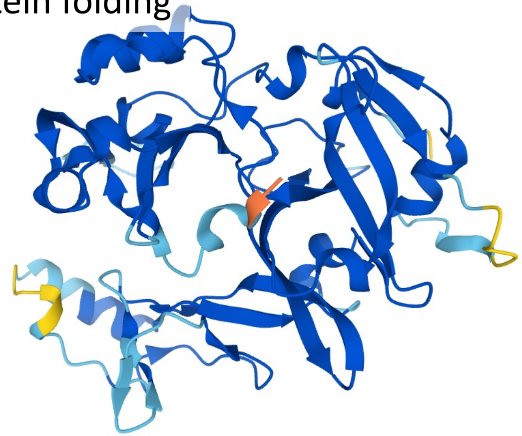
Supervised ML is at the heart of many AI advances

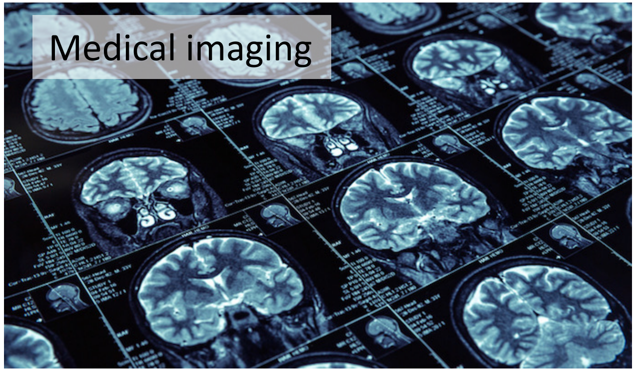
Language modelling

Given previous words ->
Predict next word



Protein folding





Supervised ML is at the heart of many AI advances

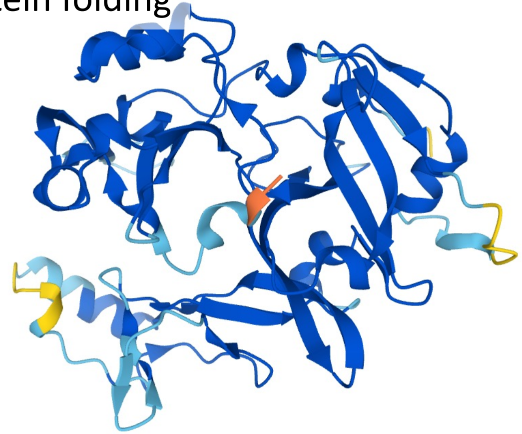
Language modelling

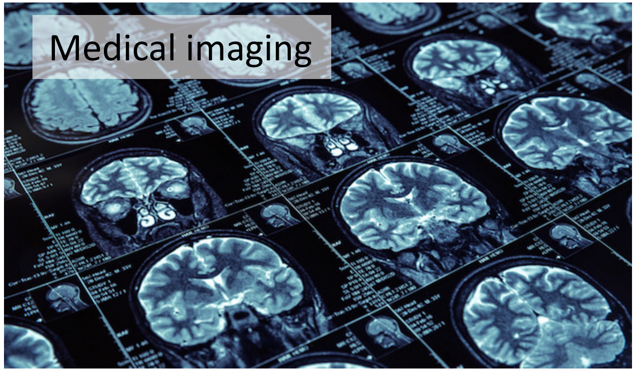
Given previous words ->
Predict next word

Game playing

Given current board state ->
Predict probability of winning

Protein folding





Supervised ML is at the heart of many AI advances

Language modelling

Given previous words ->
Predict next word

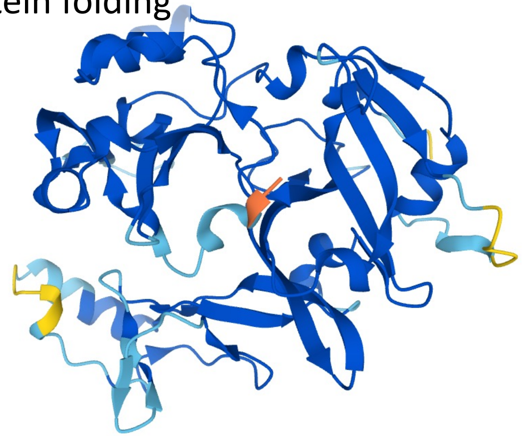
Game playing

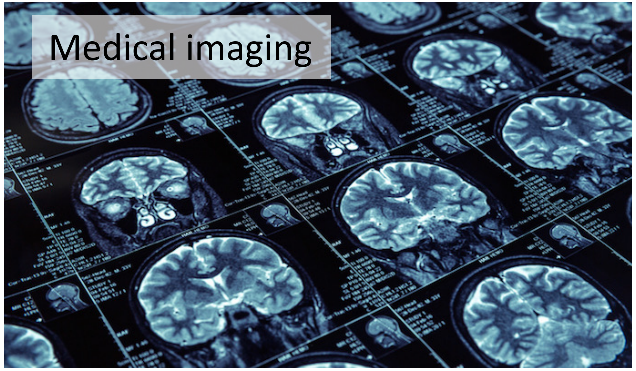
Given current board state ->
Predict probability of winning

Generative modelling

Given noisy image ->
Predict denoised image

Protein folding





Supervised ML is at the heart of many AI advances

Language modelling

Given previous words ->
Predict next word

Game playing

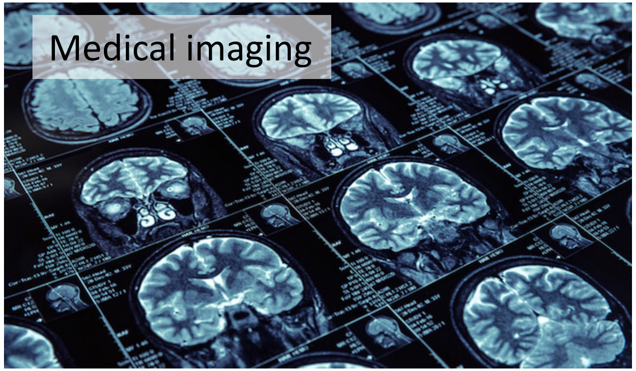
Given current board state ->
Predict probability of winning

Generative modelling

Given noisy image ->
Predict denoised image

Protein folding

Given protein chain ->
Predict 3D structure



Supervised ML is at the heart of many AI advances

Language modelling

Given previous words ->
Predict next word

Game playing

Given current board state ->
Predict probability of winning

Generative modelling

Given noisy image ->
Predict denoised image

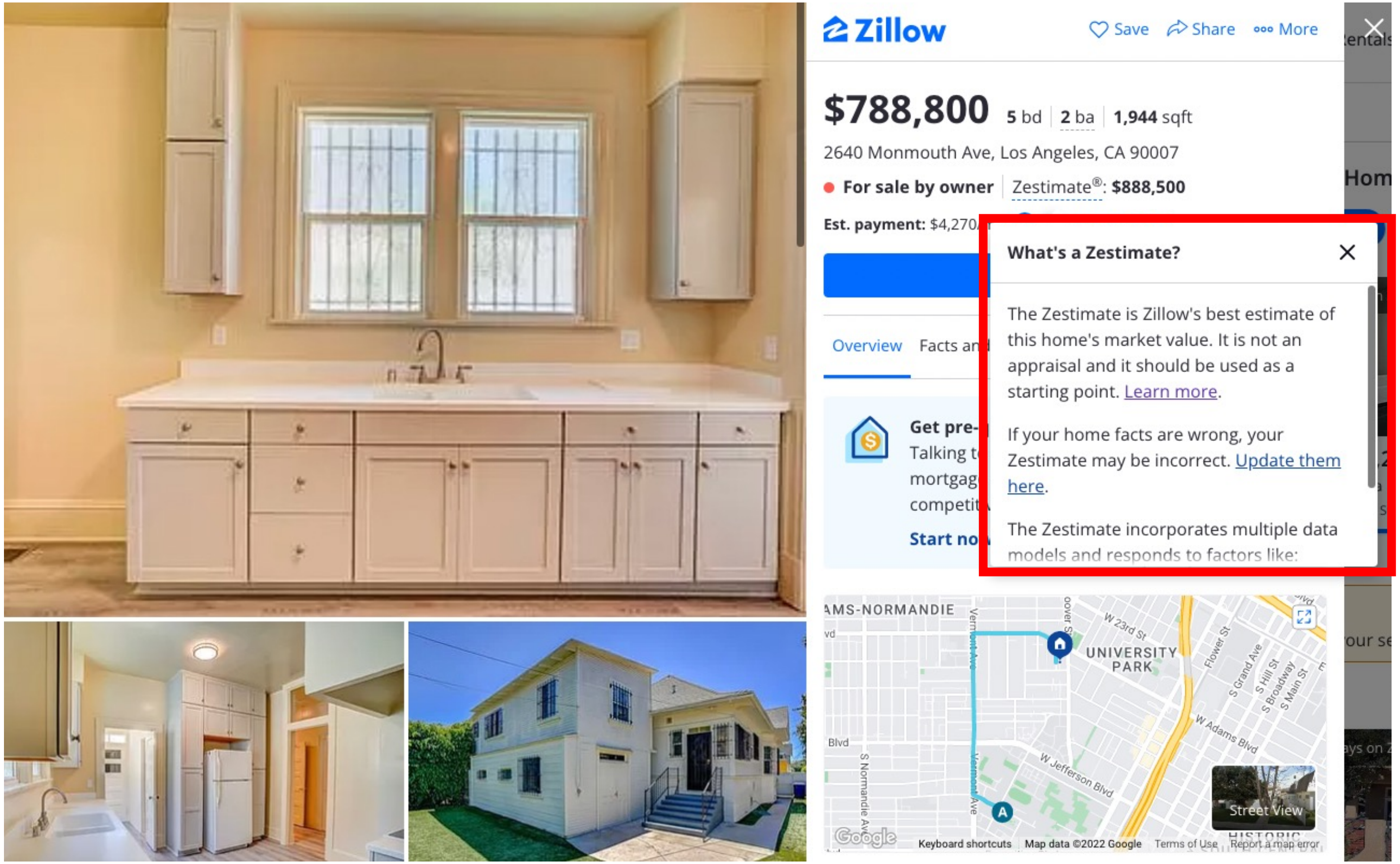
Protein folding

Given protein chain ->
Predict 3D structure

Medical imaging

Given image ->
Predict if there is tumor etc.

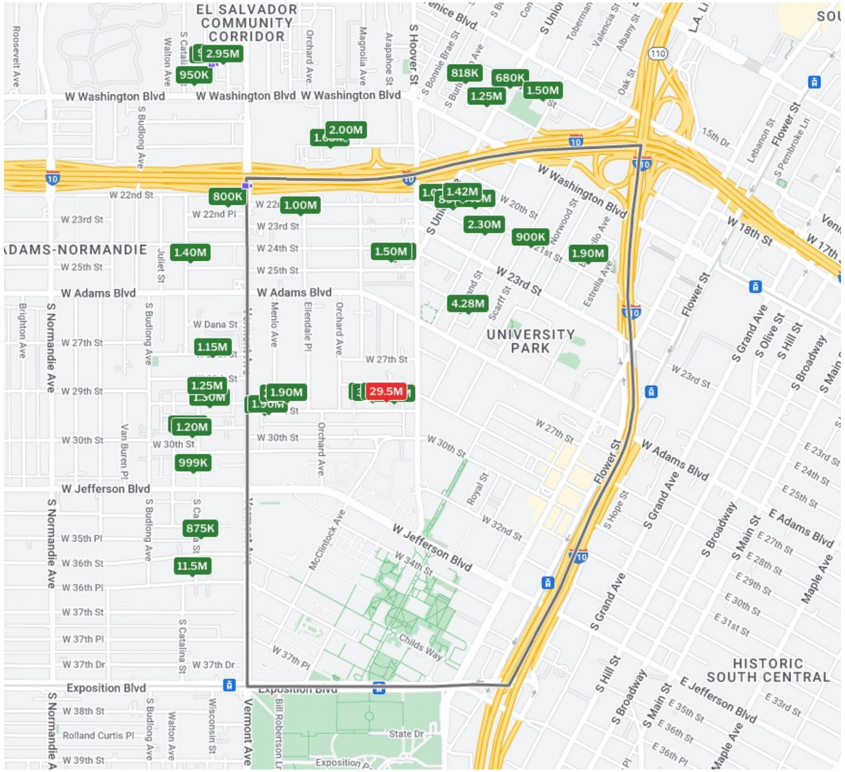
Supervised ML: Predict future outcomes using past outcomes



Predicting sale price of a house

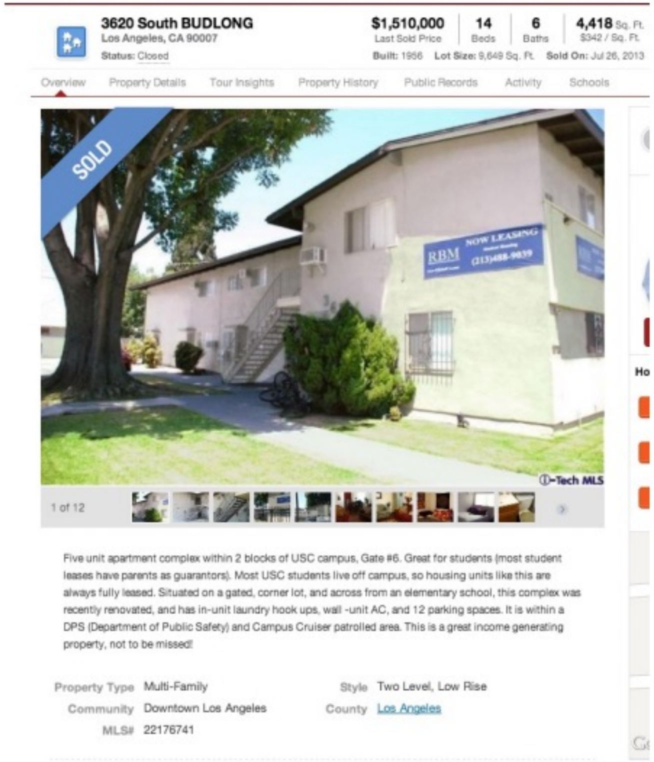
Simplistic version: Predicting sale price of a house

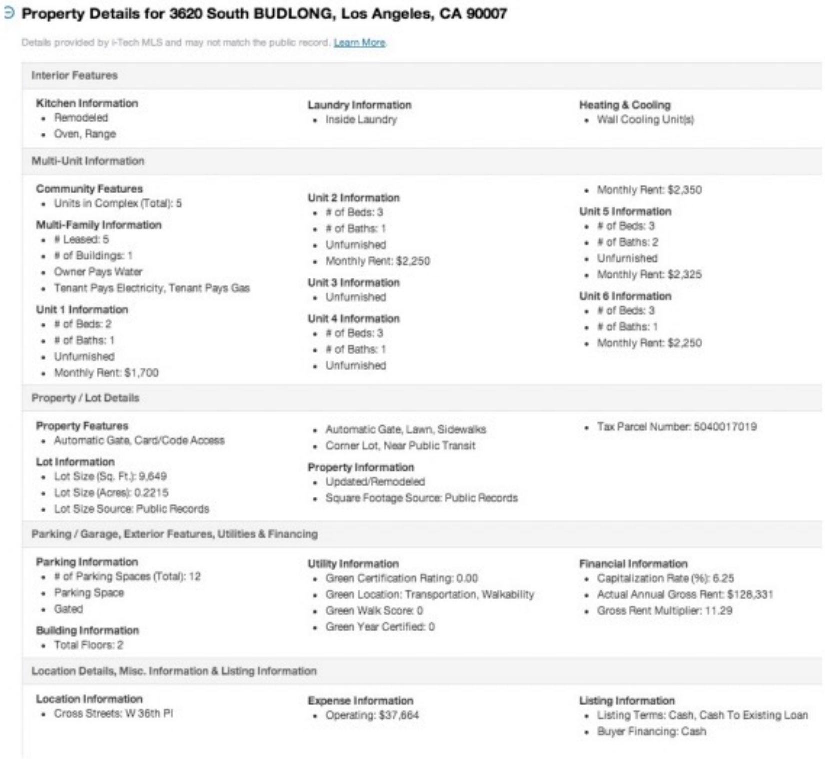
Retrieve historical sales records (training data):



Simplistic version: Predicting sale price of a house

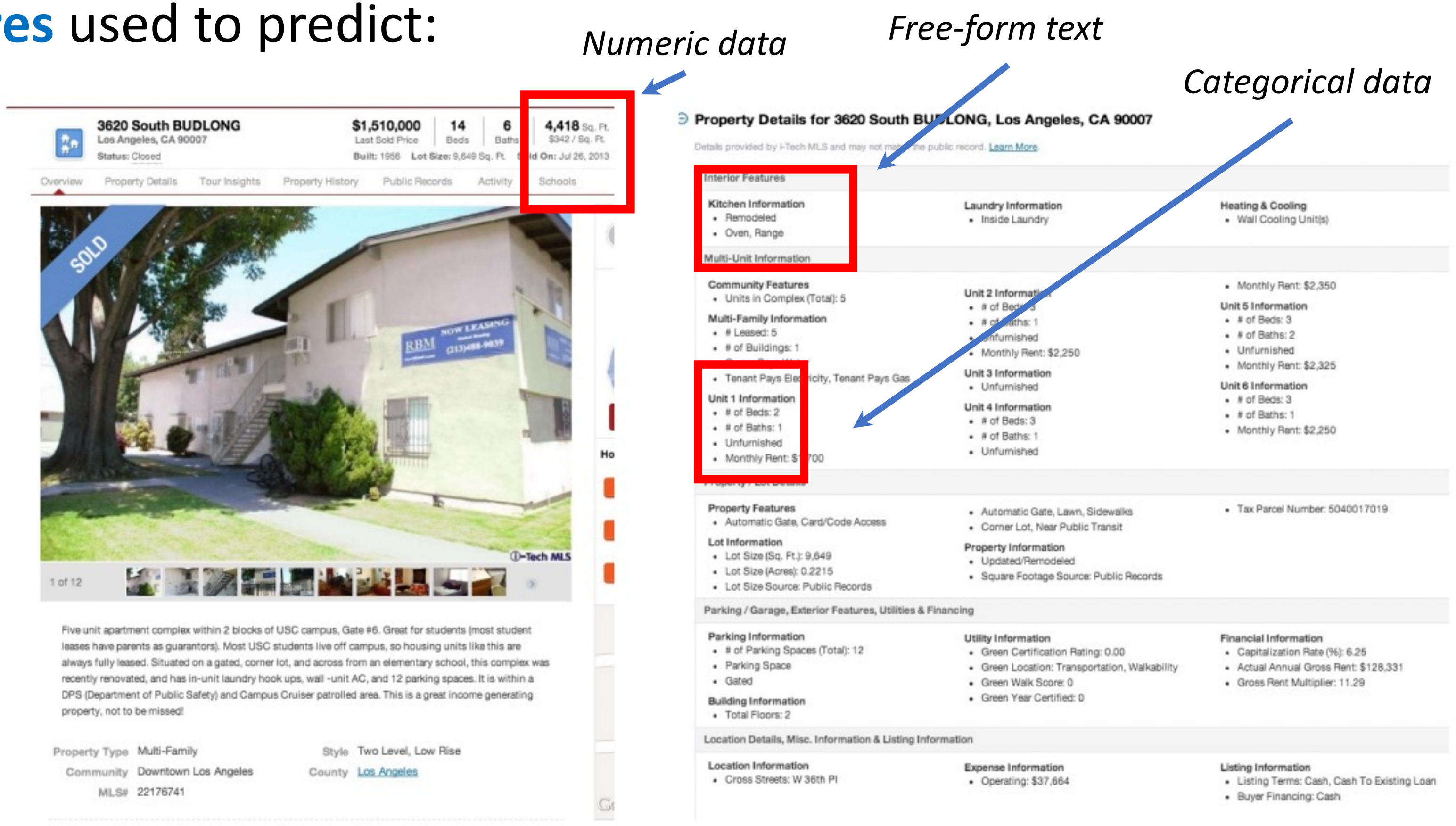
Features used to predict:





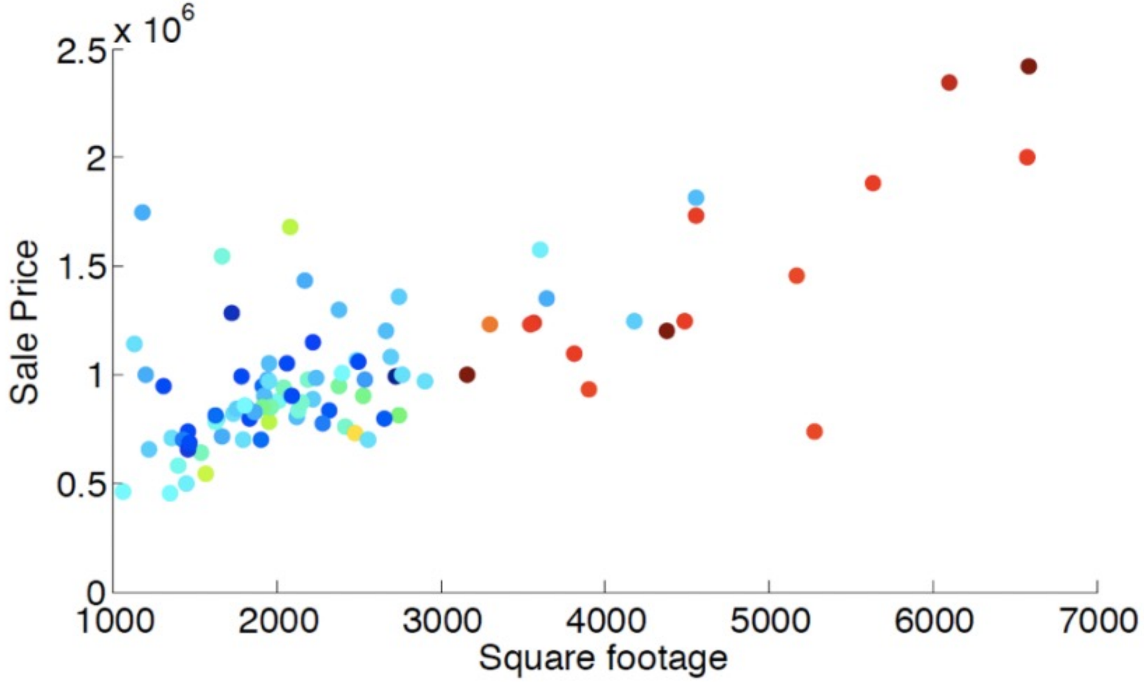
Simplistic version: Predicting sale price of a house

Features used to predict:



Simplistic version: Predicting sale price of a house

Correlation between square footage and sale price:

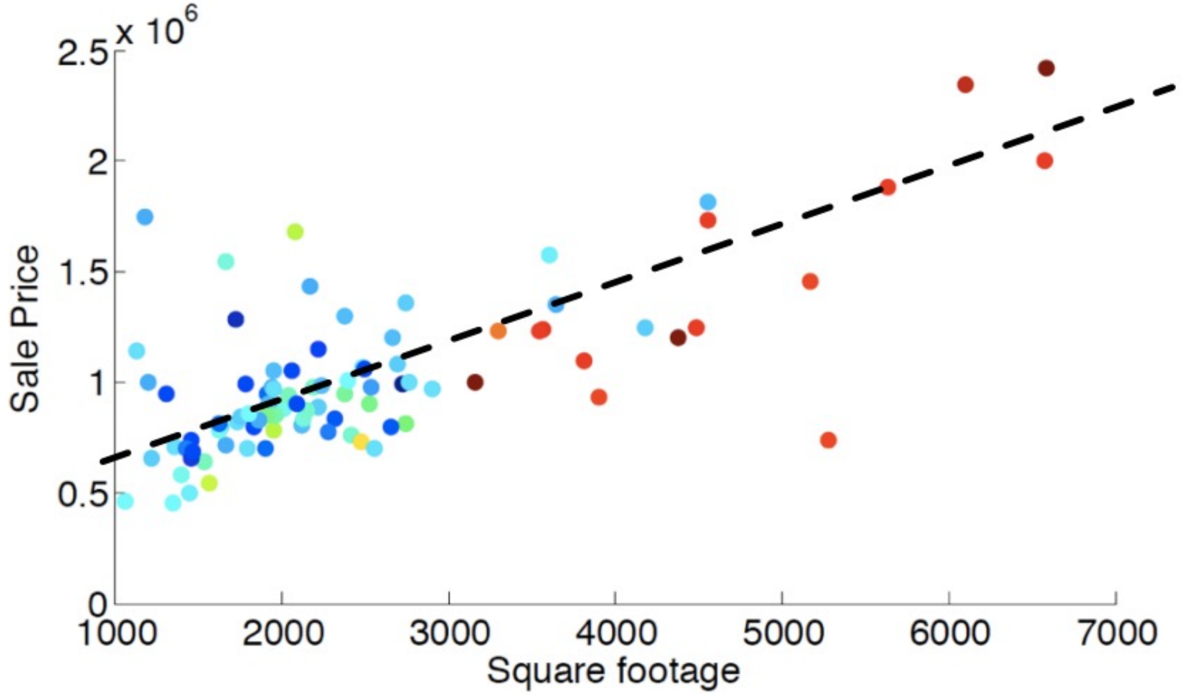


Simplistic version: Predicting sale price of a house

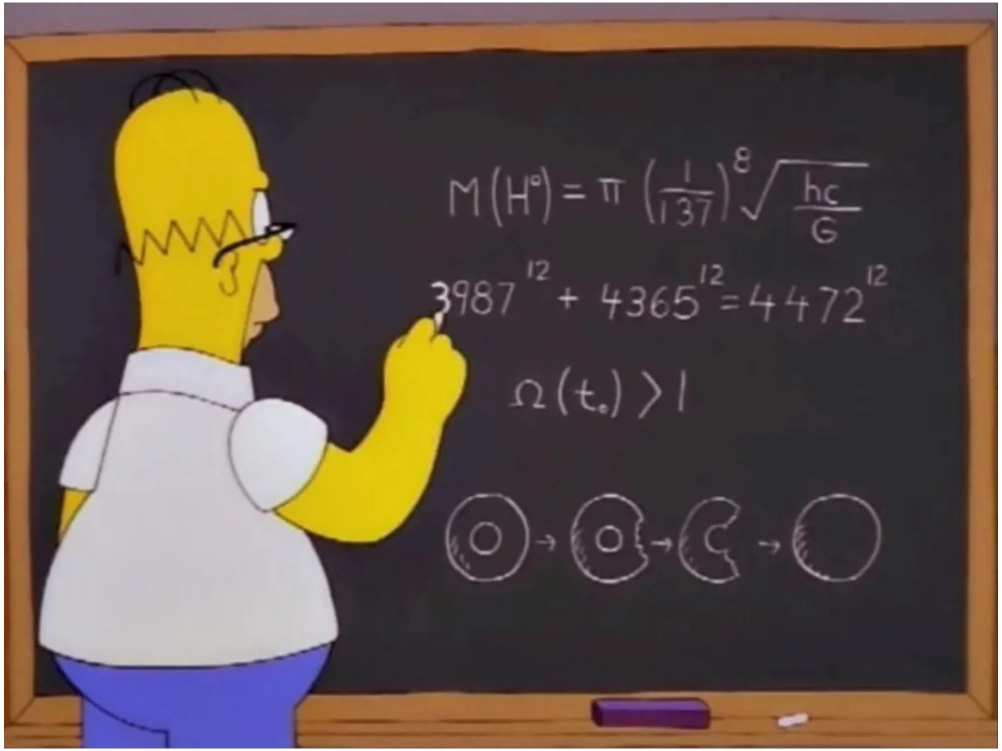
Possibly linear relationship:

Sale price ≈ **price per sqft** × square footage + **fixed expense**

(slope) *(intercept)*



General framework for supervised learning



Time for some math!

General framework for supervised learning

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

* Datapoints in d dimensions

* In previous example, d=1

} Feature engineering

→ An output space : $\mathcal{Y}$

* $y \in \mathbb{R}$ for sale price prediction

Goal : Learn a predictor $f(x) : \mathcal{X} \to \mathcal{Y}$

which predicts output of x

Loss function : $\ell(f(x), y)$

e.g. squared loss for $y \in \mathbb{R}$: $\ell(f(x), y) = (f(x) - y)^2$

What to minimize loss over?

Def: Given a set of labelled datapoints $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, the **training error (empirical risk)** for predictor $f: \mathcal{X} \to \mathcal{Y}$ wrt set S is

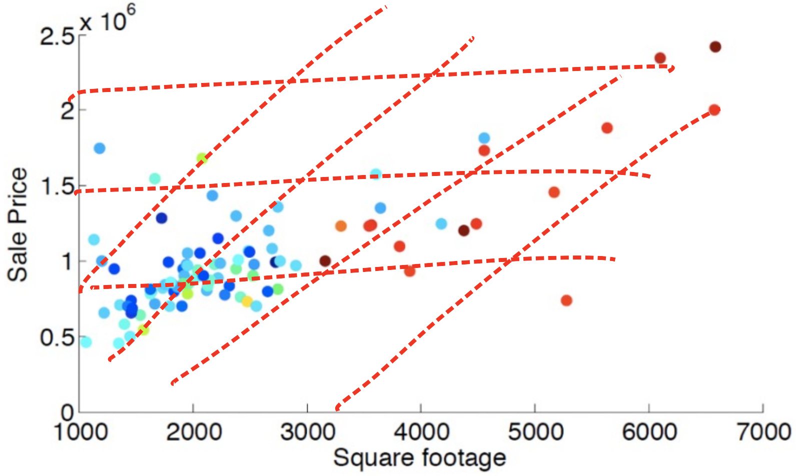
$$\hat{R}_S(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)$$

Function class

Def: A function class (hypothesis class) is a collection of functions $f: \mathcal{X} \to \mathcal{Y}$.

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

- Each of these is a linear function.
- The class of all linear functions is a function class.



Empirical risk minimizer (ERM)

Def: Given a function class $\mathcal{F} = \{f : \mathcal{X} \to \mathcal{Y}\}$, empirical risk minimization over a set of labelled datapoints S corresponds to,

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

Optimization

Generalization

* We want predictors which generalize to unseen datapoints.

Def (Test error): The test error of a predictor f is the average loss on a "new" set S' of m points

$S' = \{ (x_i', y_i'), i \in m \}$

$$\frac{1}{m} \sum_{i=1}^{m} \ell(f(x_i'), y_i')$$

Training / Test paradigm: Assume training set S & test set S' are drawn from same distribution

Measuring generalization: Training/Test paradigm

Randomly divide data into

Training set : subset of data to train model

Test set : subset of data used to test model

Generalization gap : Difference b/w test & training errors

Generalization: More formally

Minimize loss over distribution (D) of instances

Def: Risk of predictor f

$$R(f) = \mathbb{E}_{(x,y)\sim D}\left[\ell(f(x), y)\right]$$

$$= \sum_{x',y'} \text{Prob}_D(x = x', y = y')\, \ell(f(x'), y')$$

How to empirically evaluate this?

The average loss on "test set" S'. $S' = \{(x_i', y_i'), i \in m\}$

$((x_i', y_i') \sim D)$

$$R(f) \approx \frac{1}{m} \sum_{i=1}^{m} \ell(f(x_i'), y_i')$$

A tautology

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

To minimize $R(f)$

→ First try to minimize $\hat{R}_S(f)$

→ What's left is $R(f) - \hat{R}_S(f)$. This is the generalization gap.

Supervised learning in one slide

Loss function: What is the right loss function for the task?

Depends on the problem that one is trying to solve, and on the rest…

Supervised learning in one slide

Loss function: What is the right loss function for the task?

Representation: What class of functions should we use?

Also known as the "inductive bias".
No-free lunch theorem from learning theory tells us that
no model can do well on every task
"All models are wrong, but some are useful", George Box

Supervised learning in one slide

Loss function: What is the right loss function for the task?

Representation: What class of functions should we use?

Optimization: How can we efficiently solve the empirical risk minimization problem?

Depends on all the above and also…

Supervised learning in one slide

Loss function: What is the right loss function for the task?

Representation: What class of functions should we use?

Optimization: How can we efficiently solve the empirical risk minimization problem?

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

Supervised learning in one slide

Loss function: What is the right loss function for the task?

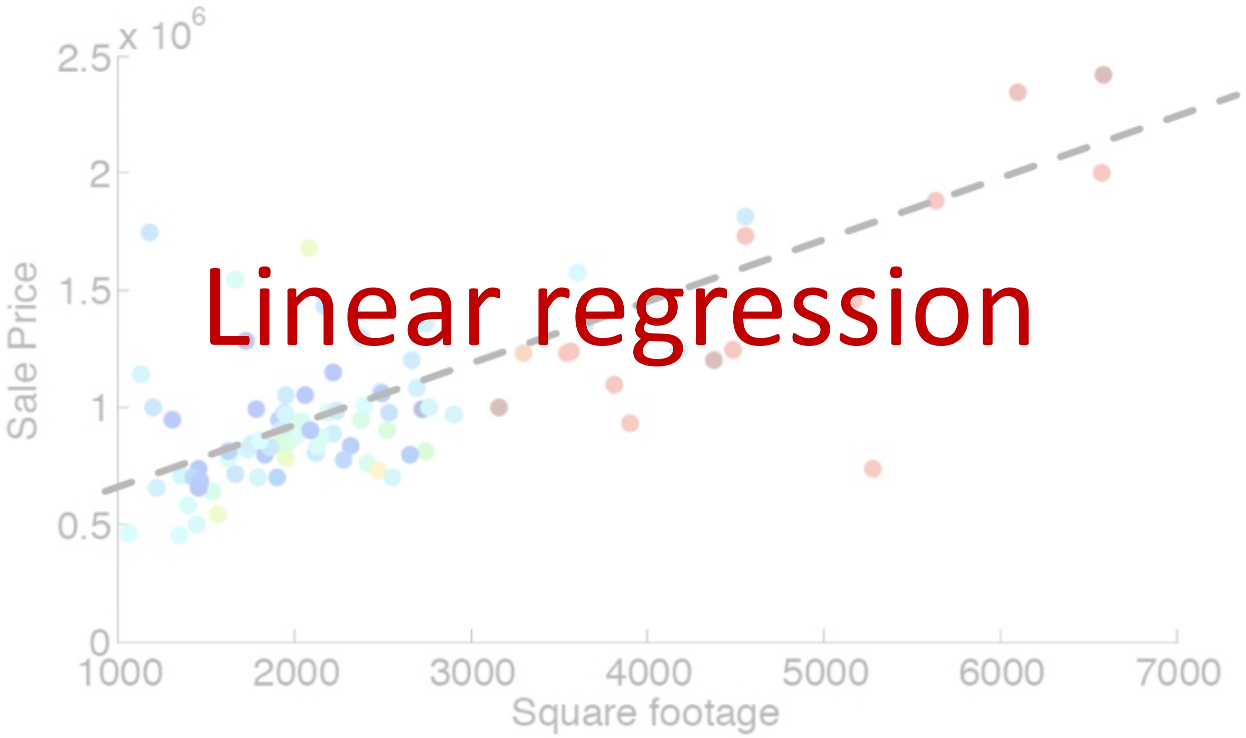
Representation: What class of functions should we use?

Optimization: How can we efficiently solve the empirical risk minimization problem?

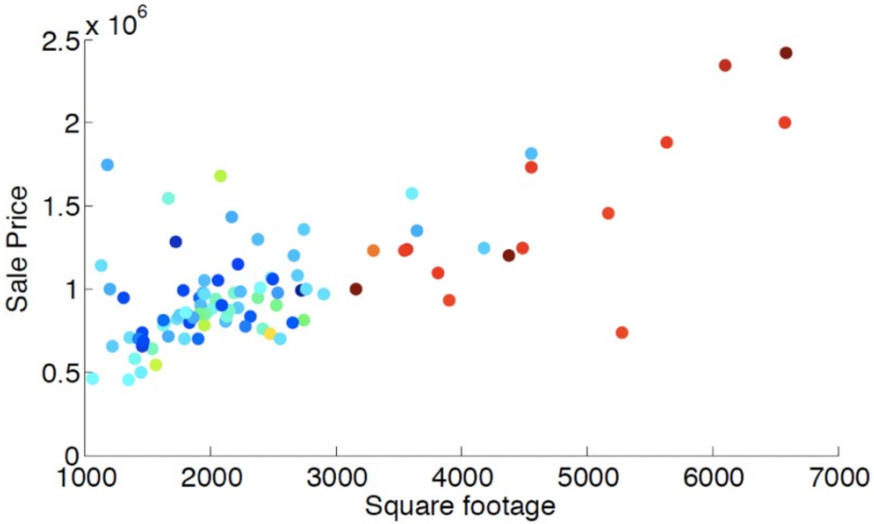
Generalization: Will the predictions of our model transfer gracefully to unseen examples?

All related! And the fuel which powers everything is data.

Linear regression



House price prediction: the loss function



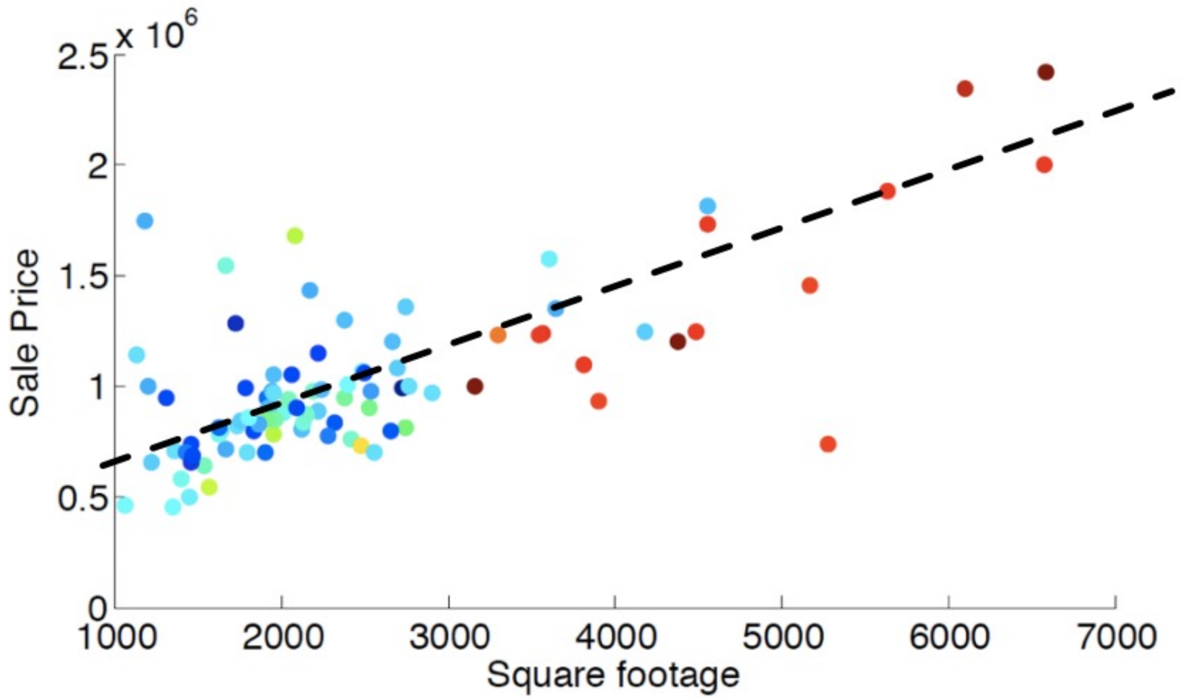
We're looking at real-valued outputs. Some popular loss functions:

- Squared loss (most common): $(\text{prediction} - \text{sale price})^2$.
- Absolute value loss: $|\text{prediction} - \text{sale price}|$.

House price prediction: **the function class**

Possibly linear relationship:

Sale price ≈ **price per sqft** × square footage + **fixed expense**



Linear regression

Predicted sale price = **price_per_sqft** × square footage + **fixed_expense**

one model: price_per_sqft = 0.3K, fixed_expense = 210K

sqft	sale price (K)	prediction (K)	squared error
2000	810	810	0
2100	907	840	67^2
1100	312	540	228^2
5500	2,600	1,860	740^2
...	...	...	...
Total			$0 + 67^2 + 228^2 + 740^2 + \cdots$

Adjust price_per_sqft and fixed_expense such that the total squared error is minimized.

Putting things together: Linear regression

- Input: $\boldsymbol{x} \in \mathbb{R}^d$, Output: $y \in \mathbb{R}$.
- Loss for predictor $f : \mathbb{R}^d \to \mathbb{R}$ on $(\boldsymbol{x}, y)$: $(f(\boldsymbol{x}) - y)^2$.
- Training data $S = \{(\boldsymbol{x}_i, y_i), i = 1, \ldots, n\}$.
- Linear model $\{f : f(x) = w_0 + \sum_{j=1}^{d} w_j x_j = w_0 + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}, \boldsymbol{w} \in \mathbb{R}^d\}$.
    - $\boldsymbol{w} = [w_1, \ldots, w_d]^\top$ are the weights.
    - w_0 is bias.

Note: For notational convenience

Append 1 to each $\boldsymbol{x}$ as first feature: $\widetilde{\boldsymbol{x}}$ = [1 x_1 x_2 … x_d $]^T$

Let $\widetilde{\boldsymbol{w}}$ = [$w_0, w_1, w_2,$ … , w_d $]^T$ represent all $d + 1$ parameters

Model becomes $f(\boldsymbol{x}) = \widetilde{\boldsymbol{w}}^T \widetilde{\boldsymbol{x}}$

Sometimes, we'll use $\boldsymbol{w}, \boldsymbol{x}, d$ for $\widetilde{\boldsymbol{w}}, \widetilde{\boldsymbol{x}}, d + 1$

Goal

- Goal is to minimize total error (empirical risk minimization):

$$\hat{R}_S(\tilde{\boldsymbol{w}}) = \frac{1}{n}\sum_{i=1}^{n}(f(\boldsymbol{x}_i) - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(\tilde{\boldsymbol{x}}_i{}^\top \tilde{\boldsymbol{w}} - y_i)^2.$$

- Define Residual Sum of Squares:

$$\mathrm{RSS}(\tilde{\boldsymbol{w}}) = n\hat{R}_S(\tilde{\boldsymbol{w}}) = \sum_{i=1}^{n}(\tilde{\boldsymbol{x}}_i{}^\top \tilde{\boldsymbol{w}} - y_i)^2.$$

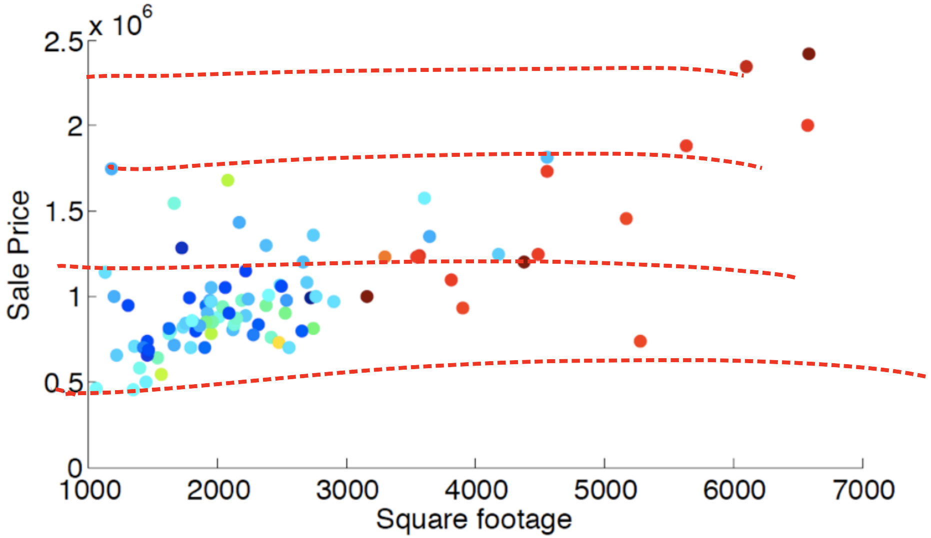
- Goal of empirical risk minimization:

$$\tilde{\boldsymbol{w}}^* = \operatorname*{argmin}_{\tilde{\boldsymbol{w}} \in \mathbb{R}^{d+1}} \mathrm{RSS}(\tilde{\boldsymbol{w}})$$

This is known as the least squares solution.

Warmup: $d = 0$

Only one parameter w_0: constant prediction $f(x) = w_0$



f is a horizontal line, where should it be?

Warmup: $d = 0$

$$RSS(w_0) = \sum_{i=1}^{n} (w_0 - y_i)^2$$

$$= n w_0^2 - 2 \left(\sum_{i=1}^{n} y_i \right) w_0 + \sum y_i^2$$

$$= n \left(w_0 - \frac{1}{n} \sum_i y_i \right)^2 + \underbrace{\text{const term}}_{\text{not dependent on } w_0}$$

$$w_0^* = \frac{1}{n} \sum_{i=1}^{n} y_i \quad \text{(the average)}$$

Warmup: $d = 1$

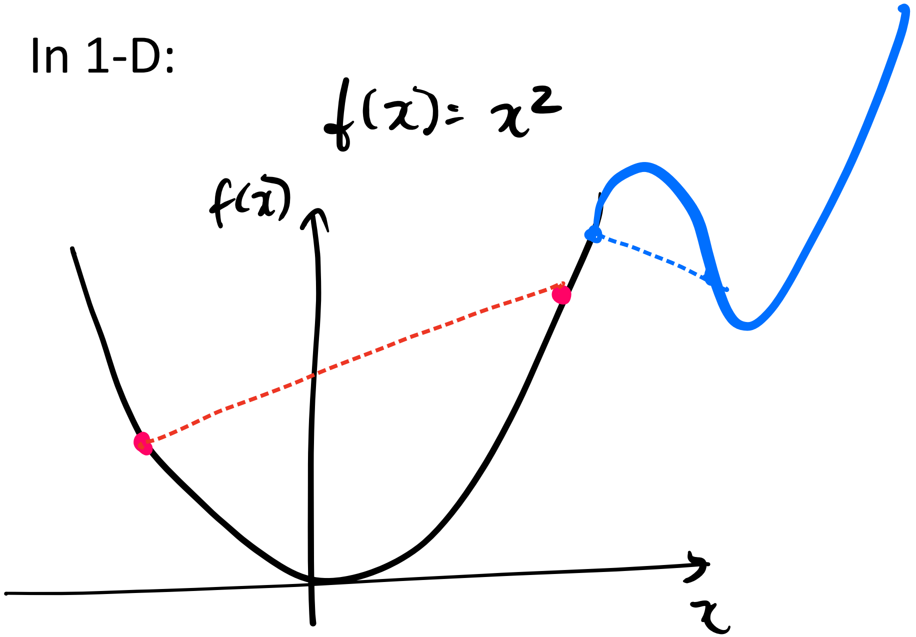
$$RSS(\tilde{w}) = \sum_i (w_0 + w_1 x_i - y_i)^2$$

General approach: find stationary point i.e. points with zero gradient

Are stationary points minimizers?

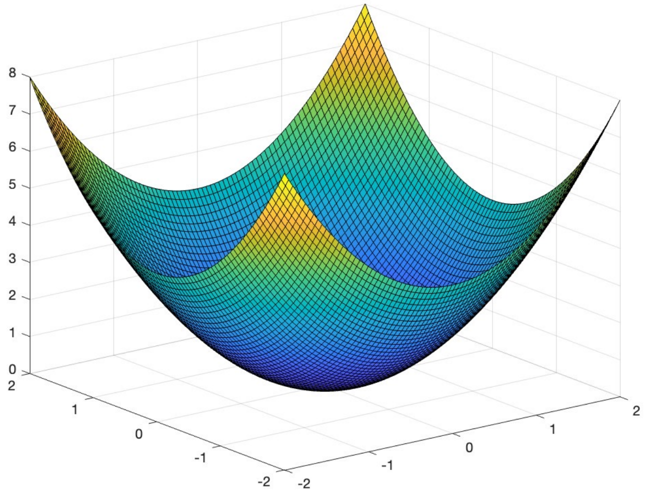
Yes, for ***convex*** objectives!

In 1-D:



$f''(x) \geq 0$

In high dimensions, this looks like:



$\nabla^2 F$ is positive semi-
algebra (psd)

Warmup: $d = 1$

$$\mathrm{RSS}(\tilde{\mathbf{w}}) = \sum_i (w_0 + w_1 x_i - y_i)^2$$

General approach: find stationary points, i.e., points with zero gradient.

$$\frac{\partial \, \mathrm{RSS}(\tilde{w})}{\partial w_0} = 0 \qquad 2\sum_{i=1}^{n} (w_0 + w_1 x_i - y_i) = 0$$

$$\Rightarrow \quad n w_0 + w_1 \sum_i x_i = \sum_i y_i$$

$$\frac{\partial \, \mathrm{RSS}(\tilde{w})}{\partial w_1} = 0 \qquad 2\sum_i (w_0 + w_1 x_i - y_i)\, x_i = 0$$

$$\Rightarrow \quad w_0 \sum_i x_i + w_1 \sum_i x_i^2 = \sum_i x_i y_i$$

General least square solution

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

$$\begin{pmatrix} w_0^* \\ w_1^* \end{pmatrix} = \begin{pmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{pmatrix}^{-1} \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{pmatrix}$$

General least square solution

$$RSS(\tilde{w}) = \sum_i (\tilde{x}_i^T \tilde{w} - y_i)^2$$

Set $\nabla RSS(\tilde{w}) = 0$

What is $\nabla_w F(w)$ where $F(w) = (v^T w - y)^2$?

$$F(w) = \left(\sum_j v_j w_j - y\right)^2$$

$$\frac{\partial F}{\partial w_i} = 2\left(\sum_j v_j w_j - y\right) v_i$$

$$\nabla_w F = \left[\, 2\left(\sum_j (v_j w_j - y)\right) v_1,\; 2\left(\sum_j (v_j w_j - y)\right) v_2,\; \ldots \right]$$

$$= 2\,(v^T w - y)\, v$$

$$\nabla_{\tilde{w}} RSS(\tilde{w}) = 2 \sum_{i=1}^{n} (\tilde{x}_i^T \tilde{w} - y_i) \tilde{x}_i$$

$$= 2 \left(\sum_i \tilde{x}_i \tilde{x}_i^T \right) \tilde{w} - 2 \sum_i \tilde{x}_i y_i$$

$$\tilde{X} = \begin{pmatrix} \tilde{x}_1^T \\ \tilde{x}_2^T \\ \vdots \\ \tilde{x}_n^T \end{pmatrix} \in \mathbb{R}^{n \times (d+1)} \qquad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^n$$

$$\nabla RSS(\tilde{w}) = 2 \left((\tilde{X}^T \tilde{X}) \tilde{w} - \tilde{X}^T Y \right) = 0$$

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

(assuming $\tilde{X}^T \tilde{X}$ is invertible)

Covariance matrix and understanding LS

$$\tilde{X}^T \tilde{X} = \begin{pmatrix} | & | & & | \\ \tilde{x}_1 & \tilde{x}_2 & \cdots & \tilde{x}_n \\ | & | & & | \end{pmatrix} \begin{pmatrix} — & \tilde{x}_1^T & — \\ — & \tilde{x}_2^T & — \\ & \vdots & \\ — & \tilde{x}_n^T & — \end{pmatrix}$$

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

each weight w_j is just the covariance of the jth feature with the label.

Another approach

RSS is a **quadratic**, so let's complete the square:

For any v, $\|v\|_2^2 = v^T v$

$$\mathrm{RSS}(\tilde{\boldsymbol{w}}) = \sum_i (\tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{x}}_i - y_i)^2 = \|\tilde{\boldsymbol{X}}\tilde{\boldsymbol{w}} - \boldsymbol{y}\|_2^2$$

$$= \left(\tilde{\boldsymbol{X}}\tilde{\boldsymbol{w}} - \boldsymbol{y}\right)^{\mathrm{T}} \left(\tilde{\boldsymbol{X}}\tilde{\boldsymbol{w}} - \boldsymbol{y}\right)$$

$$= \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \boldsymbol{y}^{\mathrm{T}} \tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y} + \text{cnt.}$$

$$= \left(\tilde{\boldsymbol{w}} - (\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}})^{-1}\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}\right)^{\mathrm{T}} \left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\right) \left(\tilde{\boldsymbol{w}} - (\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}})^{-1}\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}\right) + \text{cnt.}$$

Note: $\boldsymbol{u}^{\mathrm{T}} \left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\right) \boldsymbol{u} = \left(\tilde{\boldsymbol{X}}\boldsymbol{u}\right)^{\mathrm{T}} \tilde{\boldsymbol{X}}\boldsymbol{u} = \|\tilde{\boldsymbol{X}}\boldsymbol{u}\|_2^2 \geq 0$ and is 0 if $\boldsymbol{u} = 0$.

So $\tilde{\boldsymbol{w}}^* = (\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}})^{-1}\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$ is the minimizer.

Computational complexity

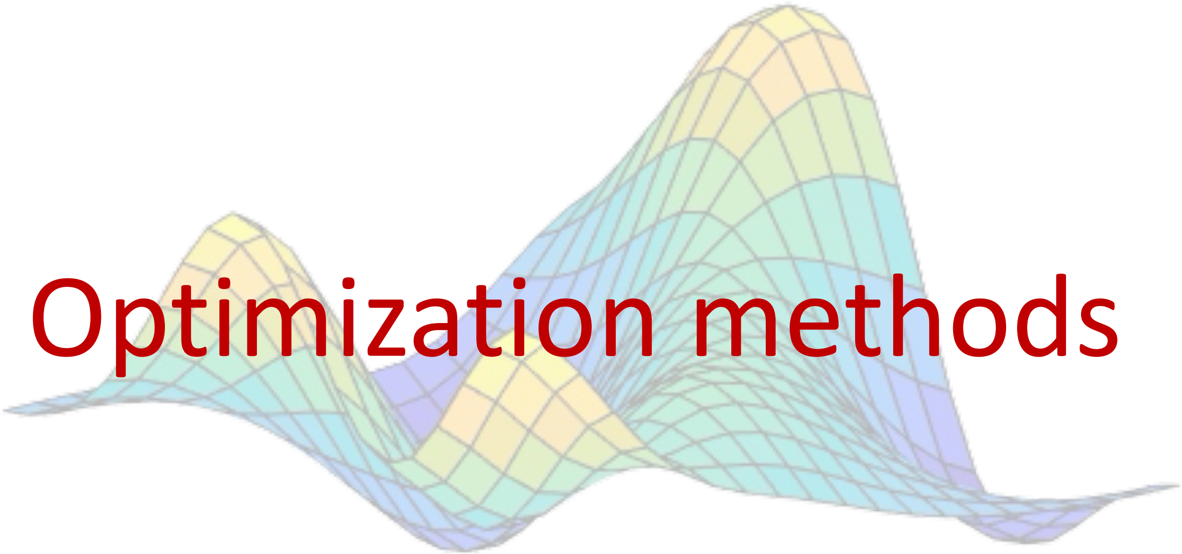
Bottleneck of computing

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

is to invert the matrix $\tilde{\boldsymbol{X}}^T \tilde{\boldsymbol{X}} \in \mathbb{R}^{(d+1)} \times \mathbb{R}^{(d+1)}$.

Takes time $O(d^3)$

Optimization methods



Problem setup
Given: a function $\mathrm{F}(\boldsymbol{w})$
Goal: minimize $\mathrm{F}(\boldsymbol{w})$ (approximately)

Two simple yet extremely popular methods
Gradient Descent (GD): simple and fundamental
Stochastic Gradient Descent (SGD): faster, effective for large-scale problems

Gradient is the *first-order information* of a function.
Therefore, these methods are called *first-order methods*.

Gradient descent

GD: keep moving in the *negative gradient direction*

Start from some w_0. For $t = 0, 1, \ldots$

$$w_{t+1} = w_t - \eta \nabla_{w=w_t} F(w)$$

where $\eta > 0$ is called the step size or learning rate

- in theory η 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

An example

Consider squared loss on one datapoint (x, y) where $x = (x^{(1)}, x^{(2)})$ for $\boldsymbol{w} = (w^{(1)}, w^{(2)})$.

$$F(\boldsymbol{w}) = (w^{(1)}x^{(1)} + w^{(2)}x^{(2)} - y)^2.$$

Gradient is

$$\frac{\partial F}{\partial w^{(1)}} = 2(w^{(1)}x^{(1)} + w^{(2)}x^{(2)} - y) \cdot x^{(1)} \qquad \frac{\partial F}{\partial w^{(2)}} = 2(w^{(1)}x^{(1)} + w^{(2)}x^{(2)} - y) \cdot x^{(2)}$$

GD:

- Initialize $w_0^{(1)}$ and $w_0^{(2)}$ (to be 0 or *randomly*), $t = 0$
- do

$$w_{t+1}^{(1)} \leftarrow w_t^{(1)} - \eta \left[2(w^{(1)}x^{(1)} + w^{(2)}x^{(2)} - y) \cdot x^{(1)}\right]$$
$$w_{t+1}^{(2)} \leftarrow w_t^{(2)} - \eta \left[2(w^{(1)}x^{(1)} + w^{(2)}x^{(2)} - y) \cdot x^{(2)}\right]$$
$$t \leftarrow t + 1$$

- until $F(\boldsymbol{w}_t)$ **does not change much** or t **reaches a fixed number**