CSCI 567: Machine Learning

Vatsal Sharan

Spring 2024

Lecture 1, Jan 12

USC University of Southern California

Logistics

Course website: https://vatsalsharan.github.io/spring24.html

- Logistics, slides, homework etc.

Ed Discussion: https://edstem.org/

- Main forum for communication

DEN: https://courses.uscden.net/d2l/home/27576

- 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: 4 homeworks (groups of 2), 2 late days per student (max 1 per HW)

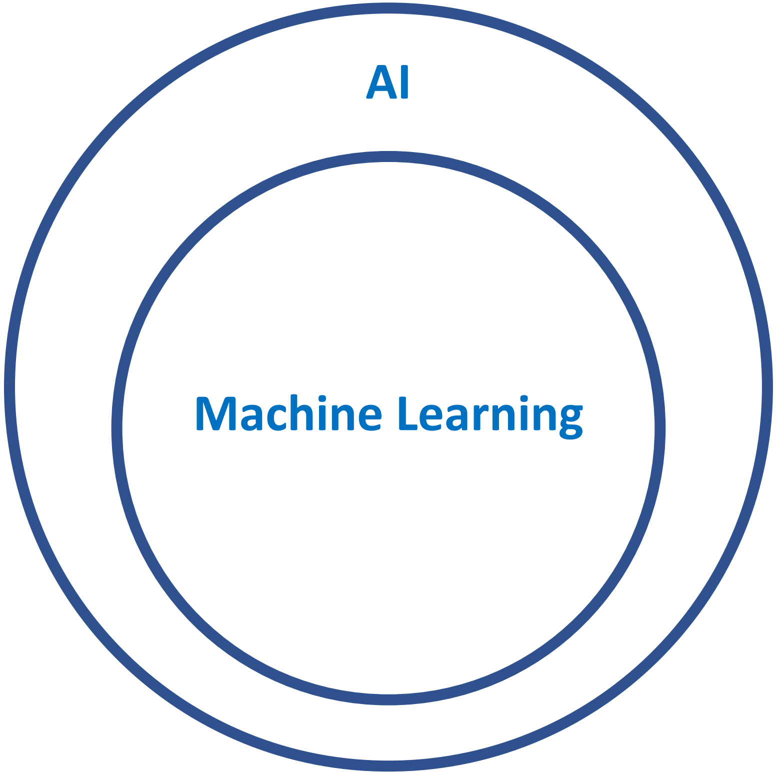
Exams: **3/1** and **4/26** during lecture hours (1pm-3:20pm)

Project: 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

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 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.

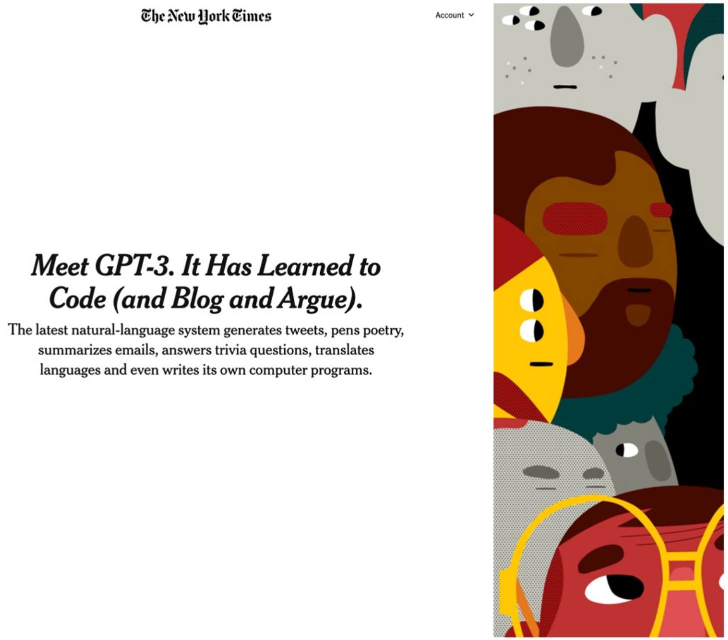
Give this article 608

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

New York Times, August 24, 2022

My slides from Fall 2022 motivating ML..

Text generation: GPT-3


The New York Times

Account

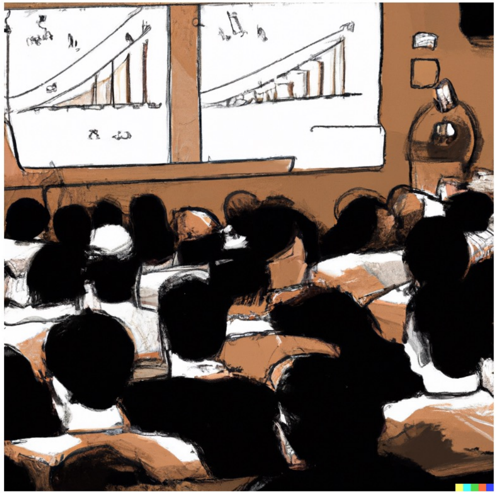
Meet GPT-3. It Has Learned to Code (and Blog and Argue).

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

Image generation: Dall-E 2

I gave the prompt:

A digital art image of a lecture on statistical machine learning. 200 students are sitting in a classroom, hearing about linear regression.



My slides from Fall 2022 motivating ML..

Text generation: GPT-3

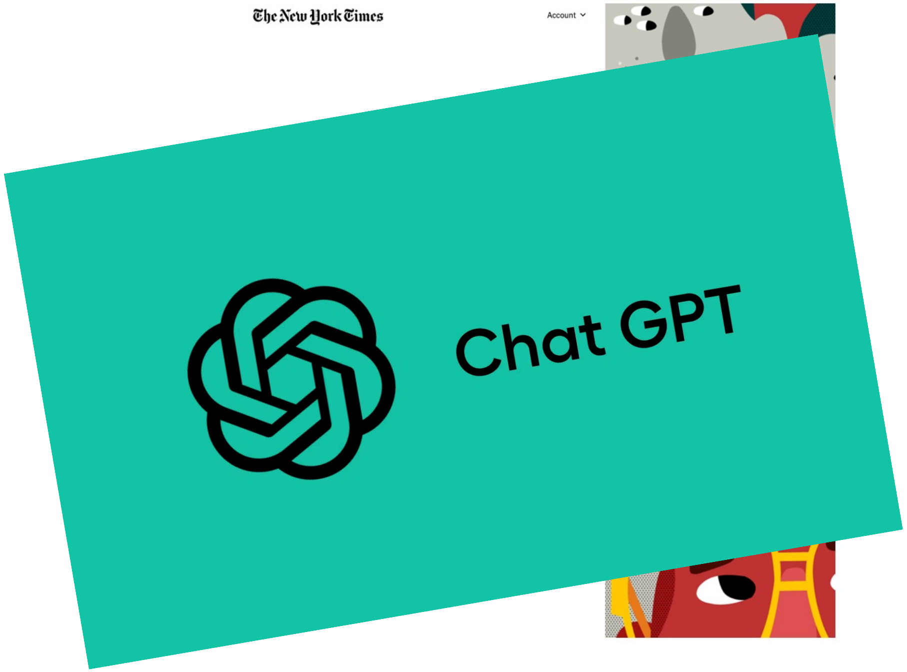
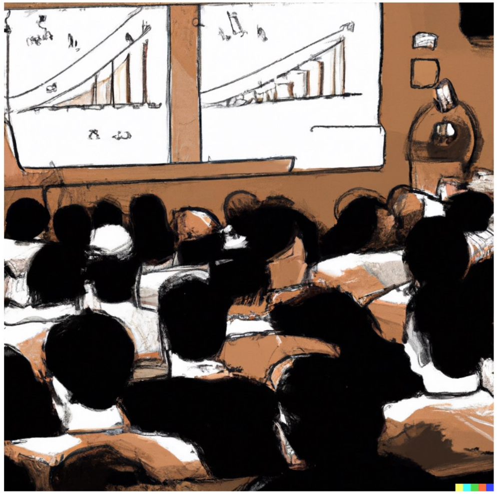


Image generation: Dall-E 2

I gave the prompt:

A digital art image of a lecture on statistical machine learning. 200 students are sitting in a classroom, hearing about linear regression.



My slides from Fall 2022 motivating ML..

Text generation: GPT-3

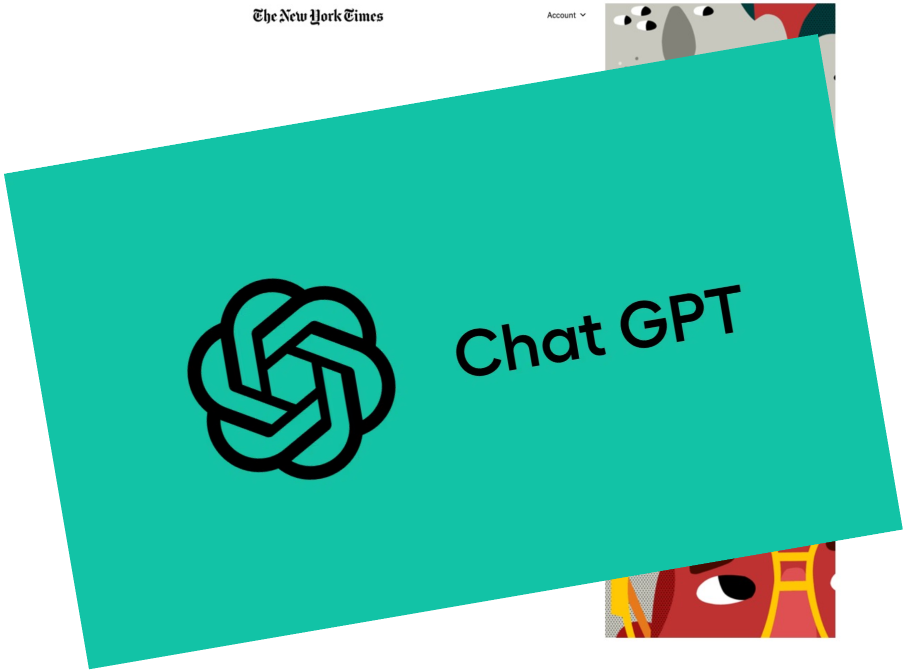


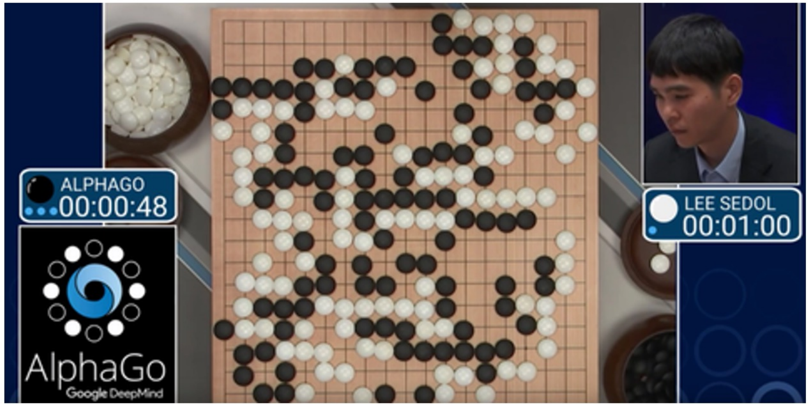
Image generation: Dall-E 2

I gave the prompt:
A digital art image of a lecture on statistical machine learning. 200 students are sitting in a classroom, hearing about linear regression.

What do you think are the most important advances?

Some other flashy highlights..

Game playing: AlphaGo



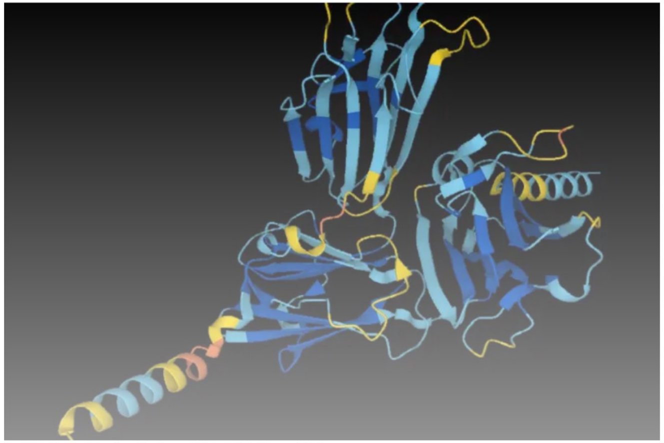
Protein folding: AlphaFold

DeepMind's protein-folding AI cracks biology's biggest problem

Artificial intelligence firm DeepMind has transformed biology by predicting the structure of nearly all proteins known to science in just 18 months, a breakthrough that will speed drug development and revolutionise basic science

TECHNOLOGY 28 July 2022

By Matthew Sparkes



Predicting the structure of proteins is one of the grand challenges of biology
DeepMind

Exciting time, but a lot needs to be done..

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

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

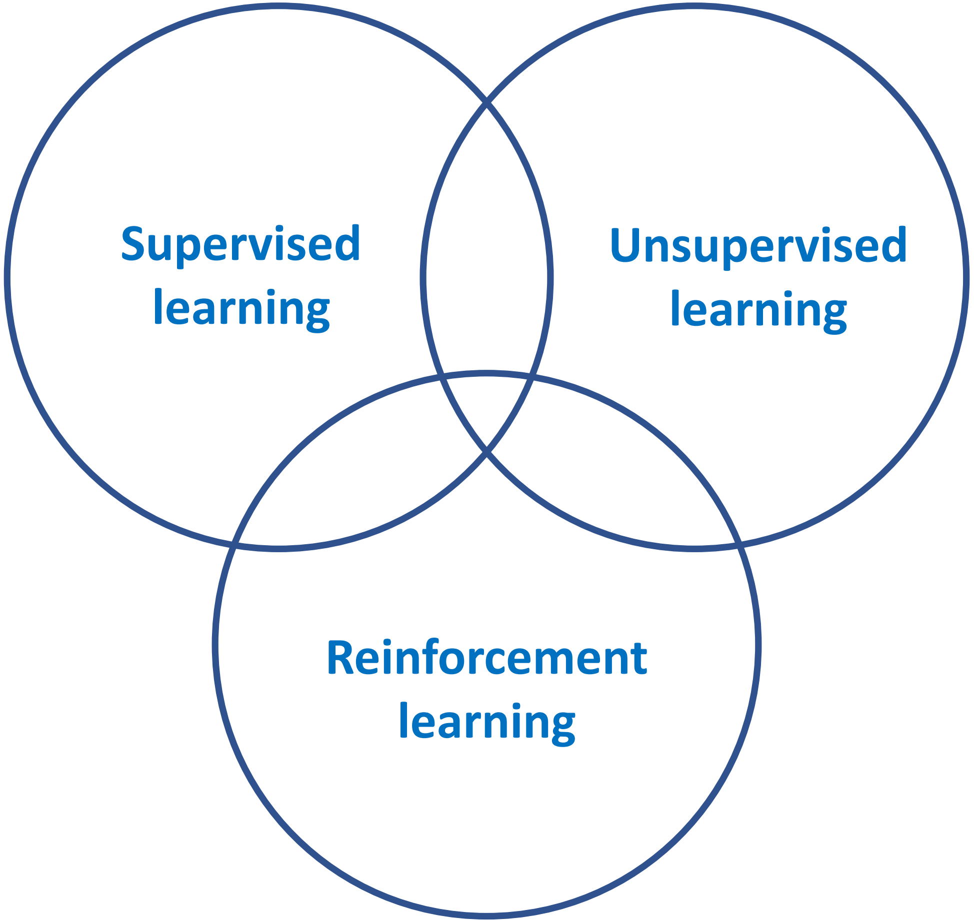
A simplistic taxonomy of ML

Supervised learning:
Aim to predict outputs of future datapoints

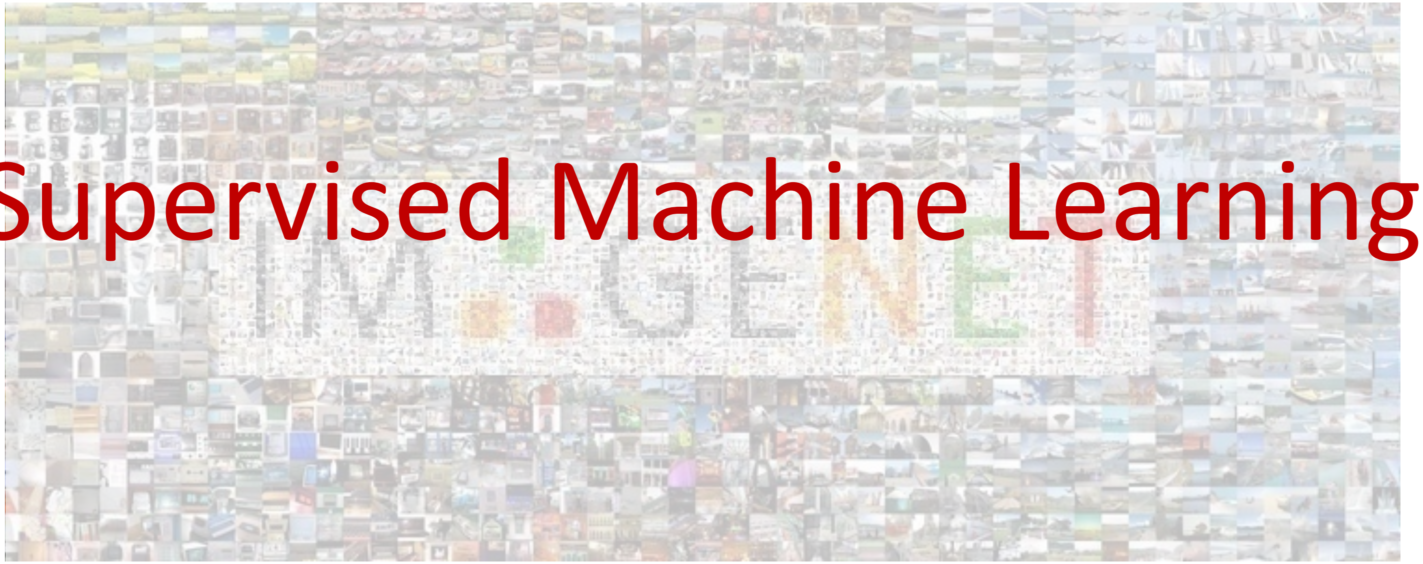
Unsupervised learning:
Aim to discover hidden patterns and explore data

Reinforcement learning:
Aim to make sequential decisions

A simplistic taxonomy of ML



Supervised Machine Learning



Supervised ML: Predict future outcomes using past outcomes

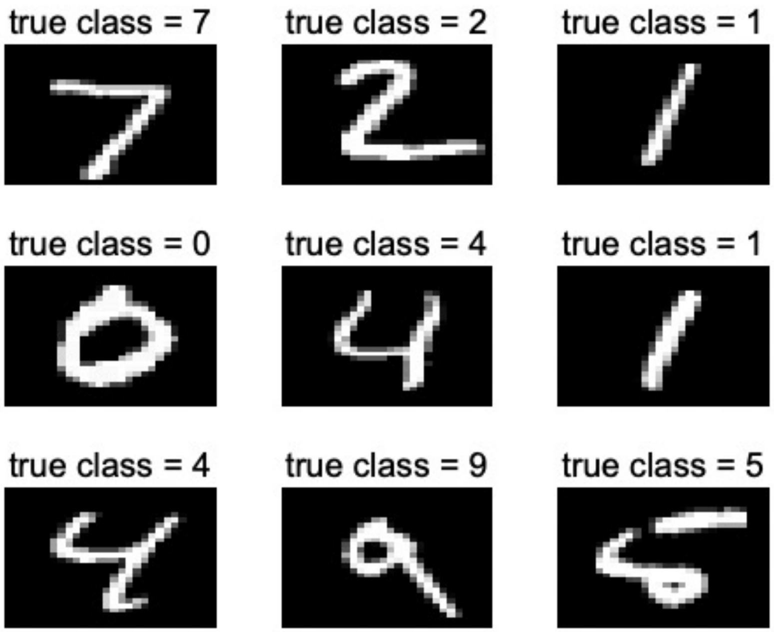
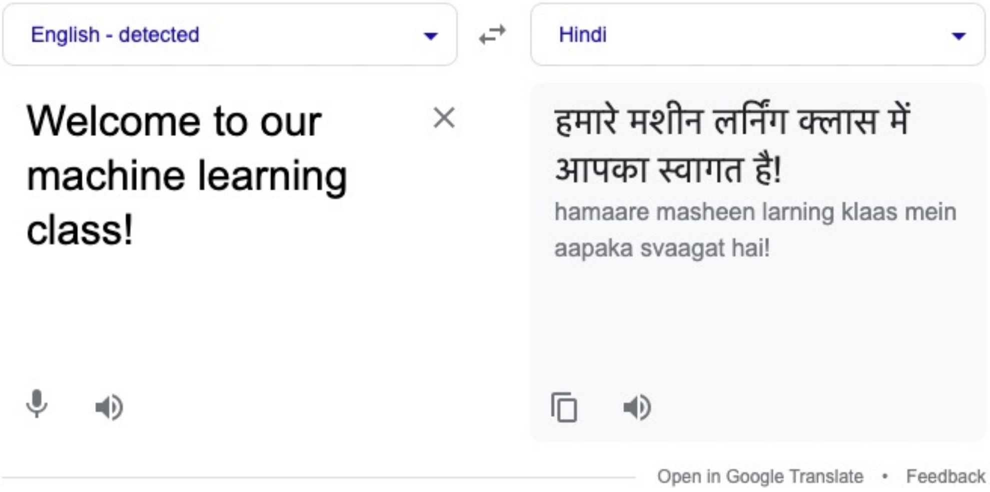
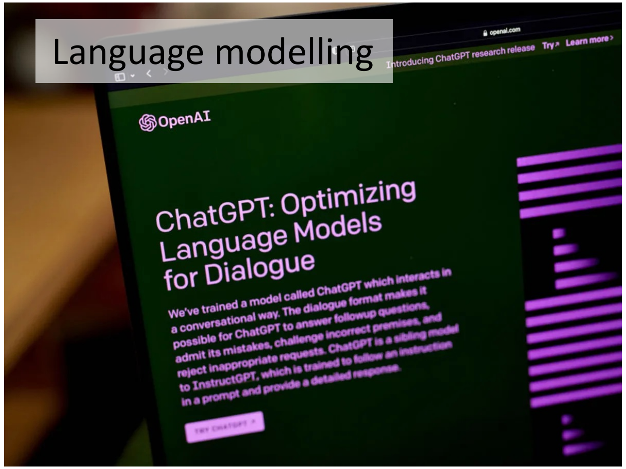


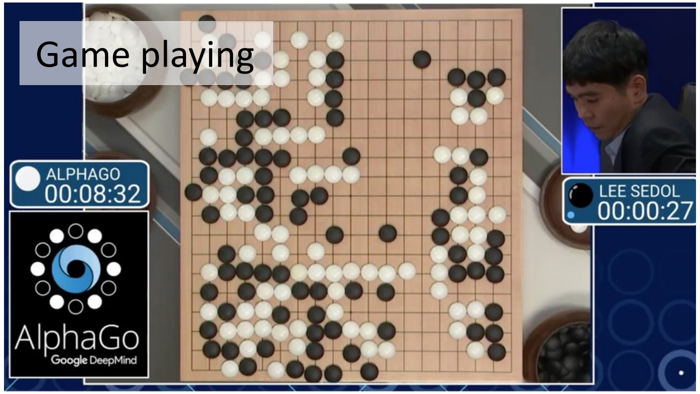
Image classification



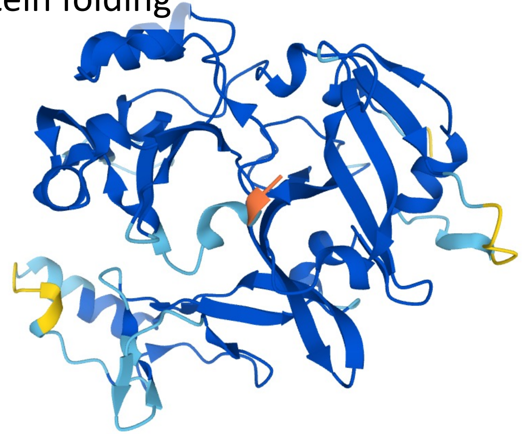
Machine translation

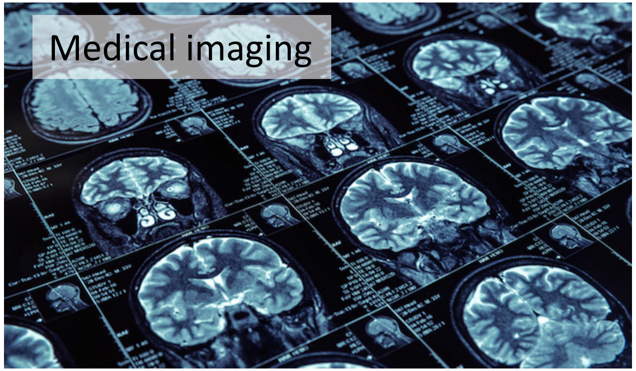
Supervised ML is at the heart of many AI advances





Protein folding

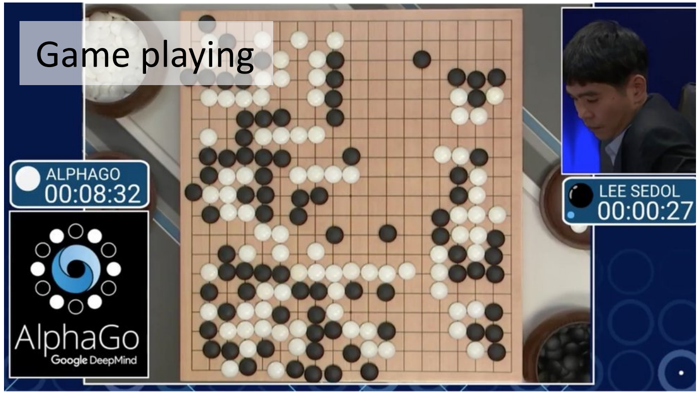




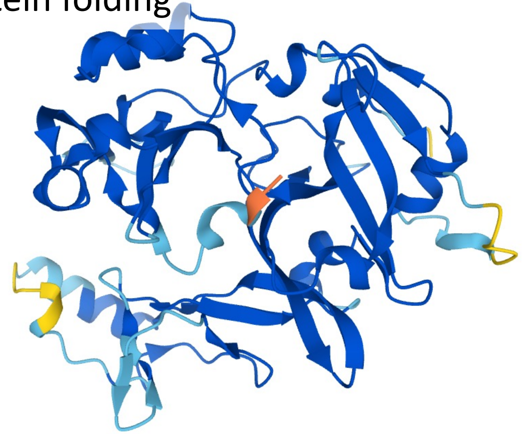
Supervised ML is at the heart of many AI advances

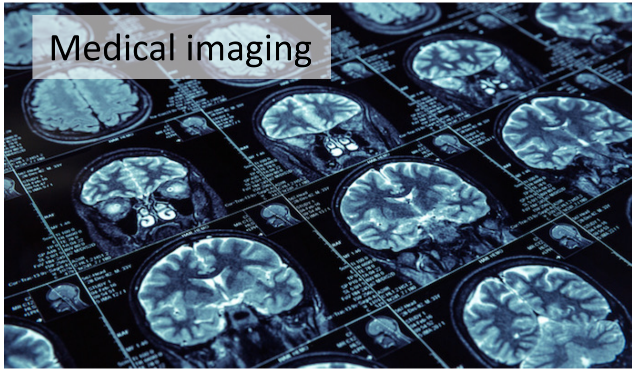
Language modelling

Given previous words ->
Predict next word



Protein folding





Supervised ML is at the heart of many AI advances

Language modelling

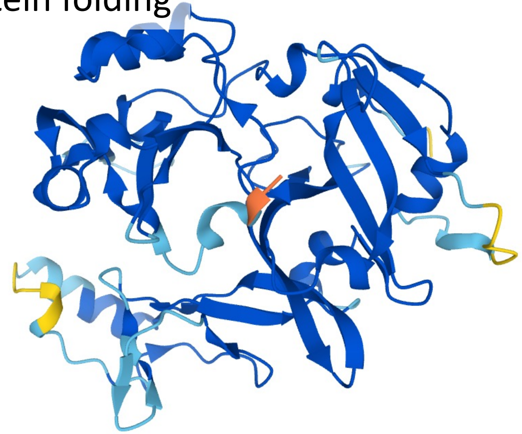
Given previous words ->
Predict next word

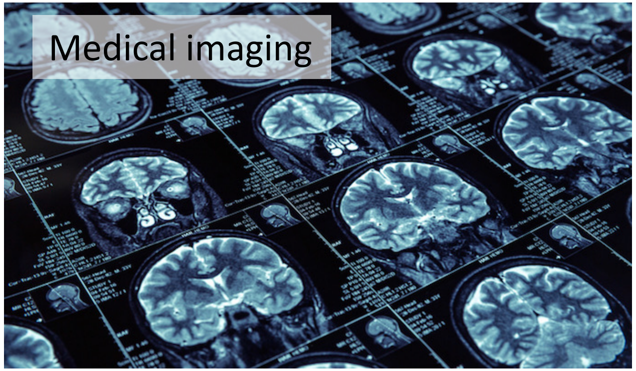
Game playing

Given current board state ->
Predict probability of winning

Generative modelling

Protein folding





Medical imaging

Supervised ML is at the heart of many AI advances

Language modelling

Given previous words ->
Predict next word

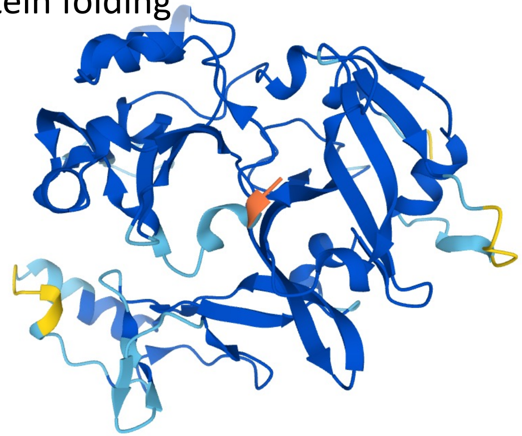
Game playing

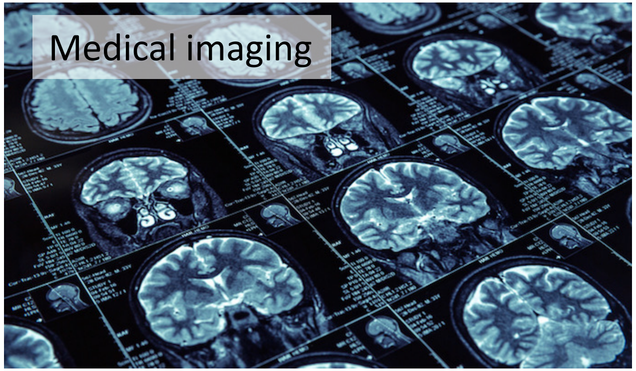
Given current board state ->
Predict probability of winning

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

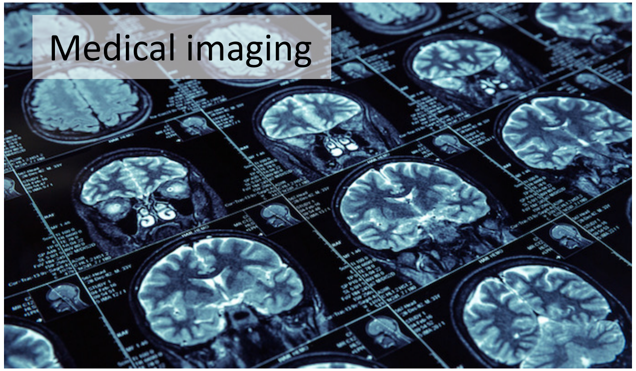
Given current board state ->
Predict probability of winning

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

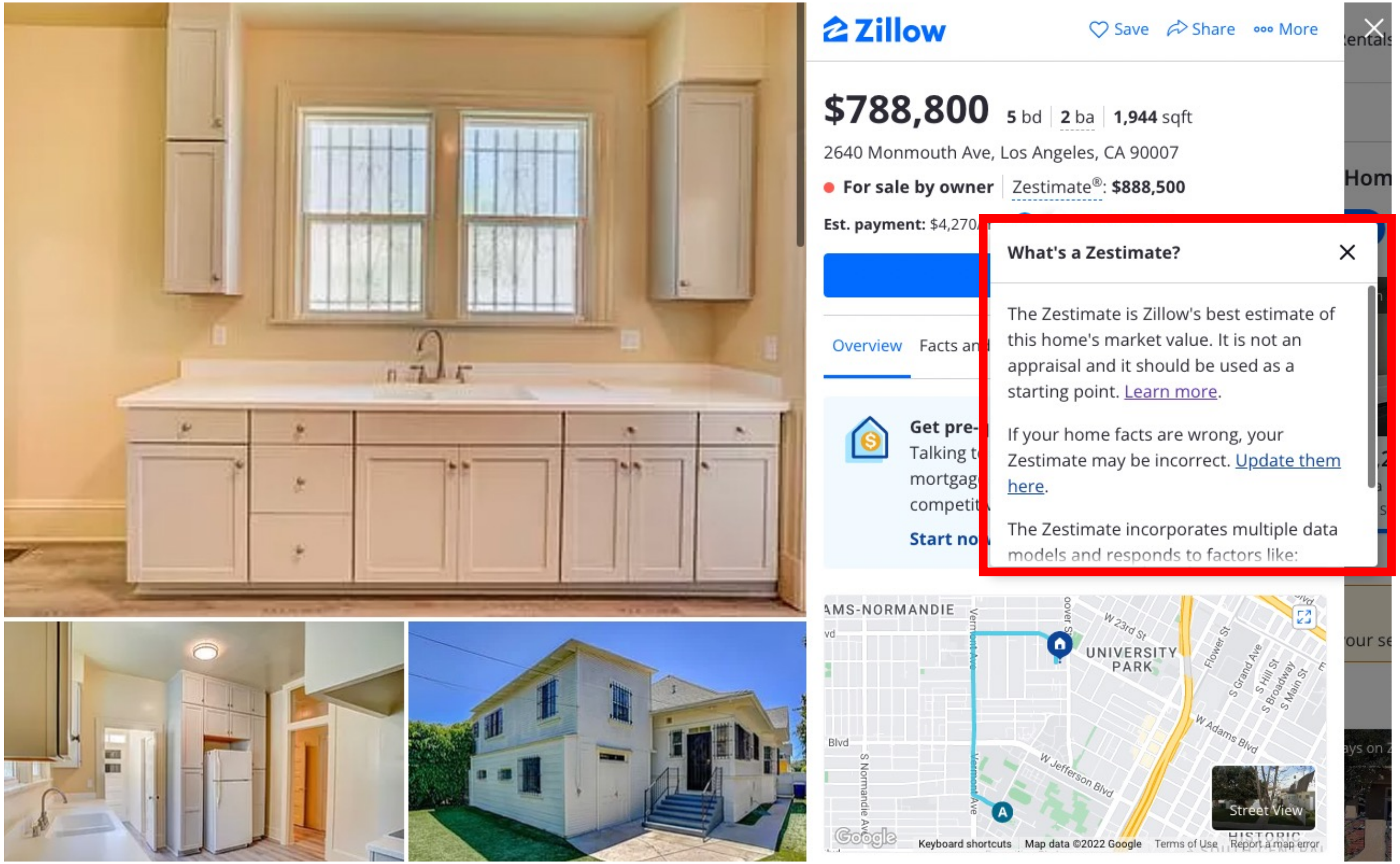
Protein folding

Given protein chain ->
Predict 3D structure

Medical imaging

Given image ->
Predict if there is tumor etc.

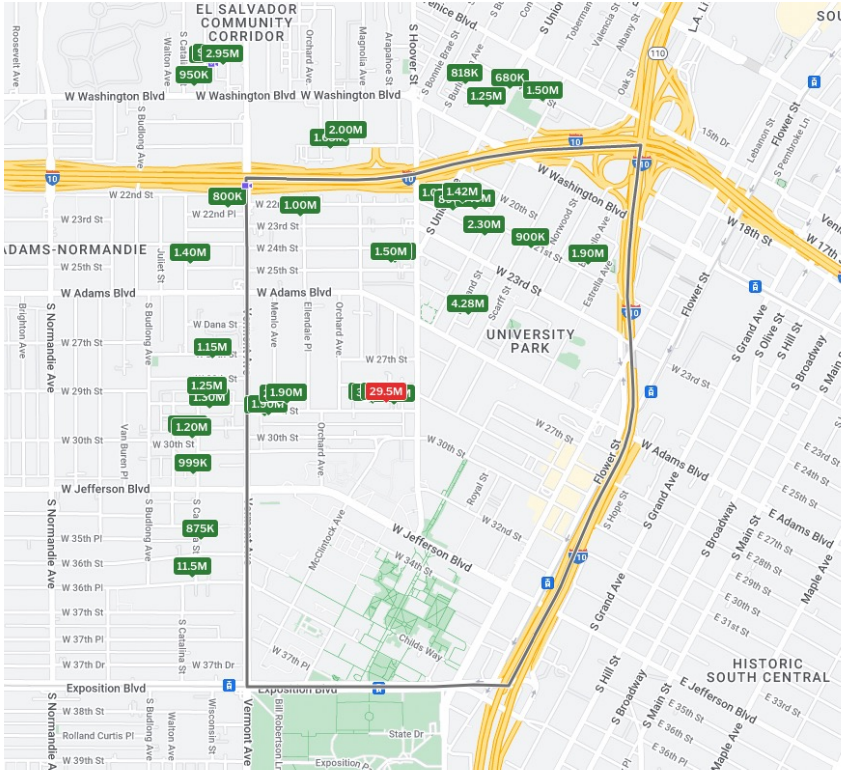
Supervised ML: Predict future outcomes using past outcomes



Predicting sale price of a house

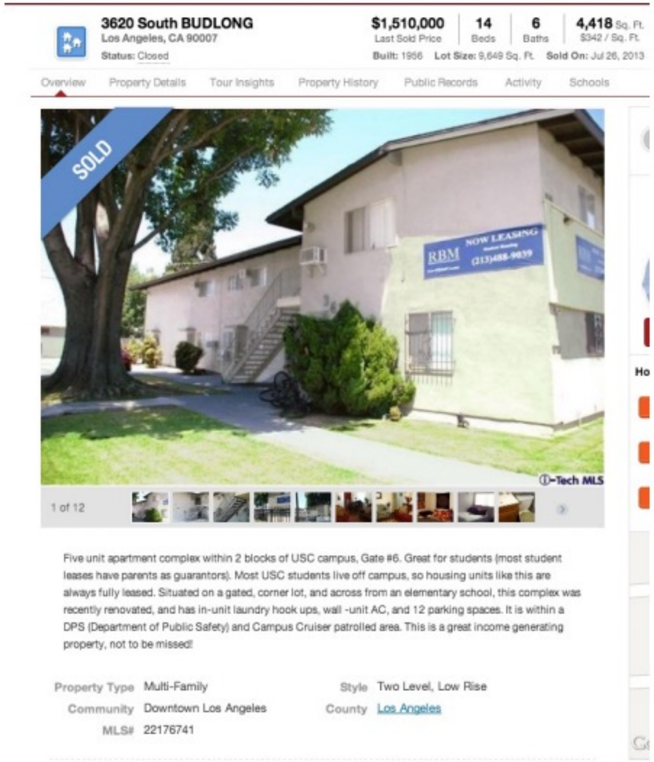
Simplistic version: Predicting sale price of a house

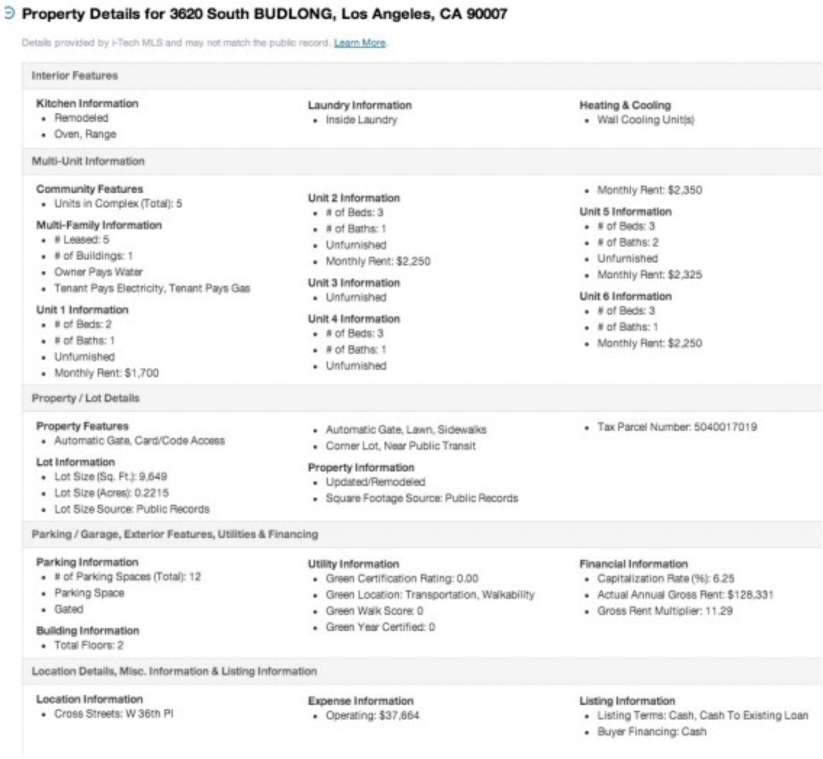
Retrieve historical sales records (training data):



Simplistic version: Predicting sale price of a house

Features used to predict:





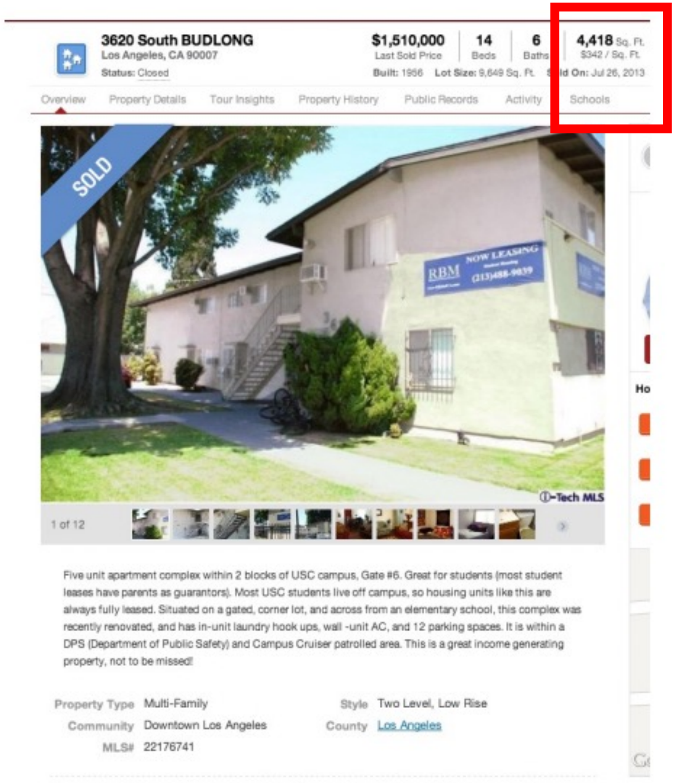
Simplistic version: Predicting sale price of a house

Features used to predict:

Numeric data

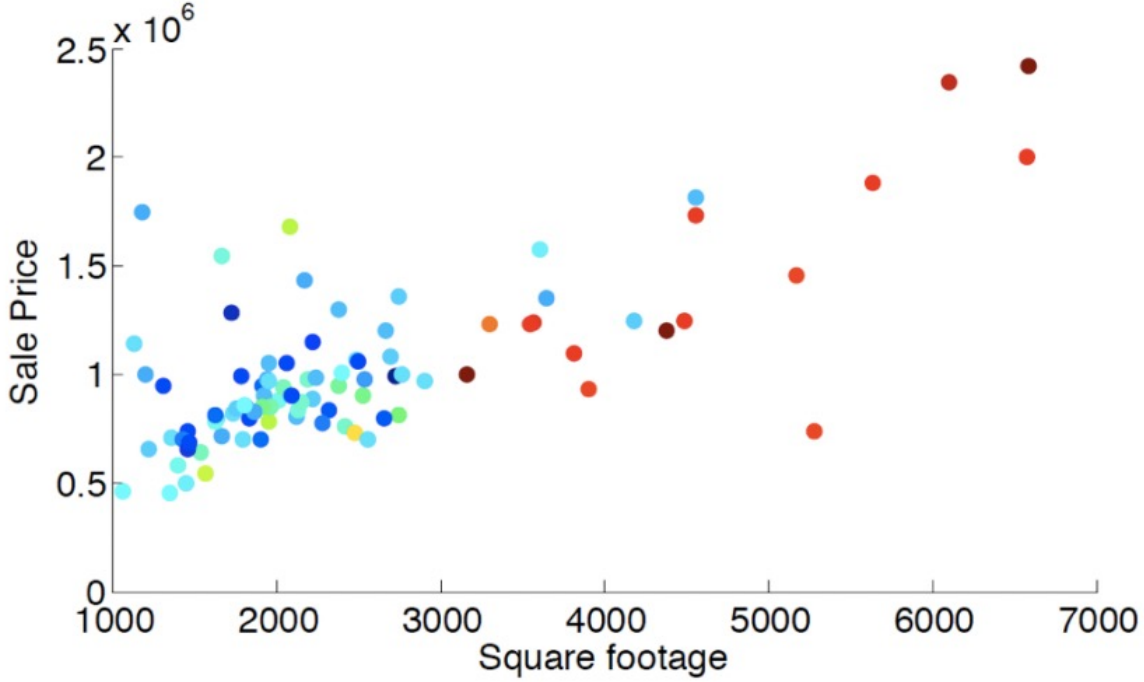
Free-form text

Categorical data



Simplistic version: Predicting sale price of a house

Correlation between square footage and sale price:

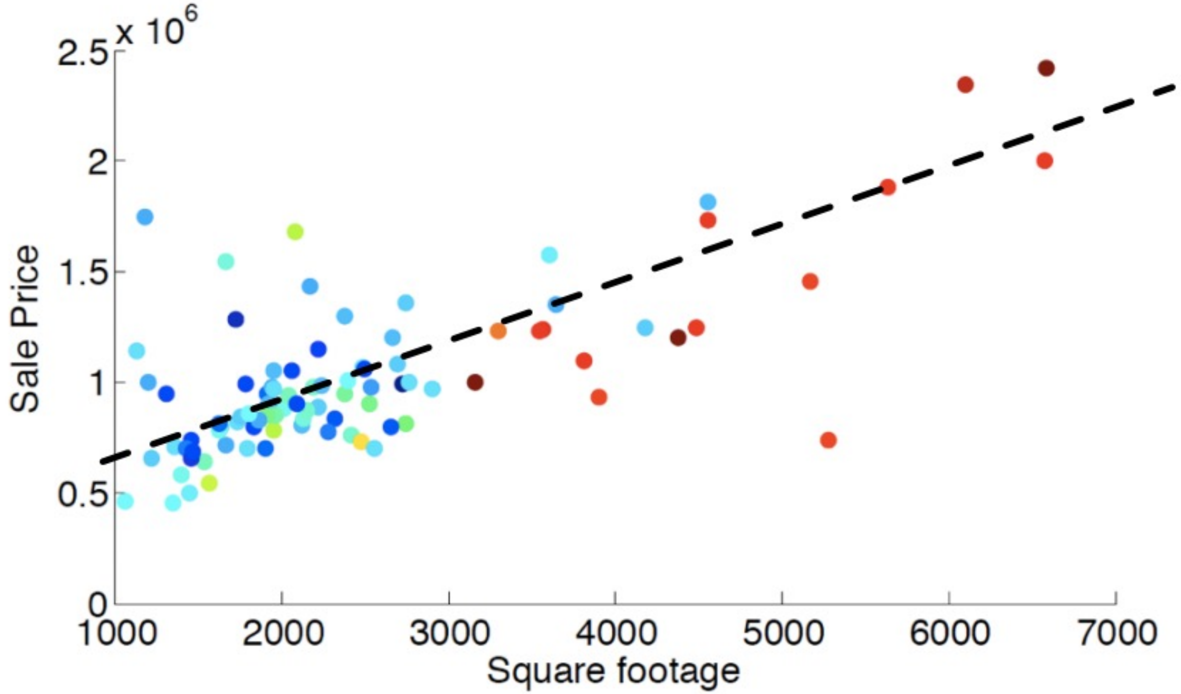


Simplistic version: Predicting sale price of a house

Possibly linear relationship:

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

(slope) *(intercept)*



General framework for supervised learning

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

* Datapoints in d dimensions
* In previous example, $d = 1$

Feature engineering !

→ An output space : $\mathcal{Y}$

* $\mathcal{Y} \in \mathbb{R}$ 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 $\mathcal{Y} = \mathbb{R}$: $\ell(f(x), y) = (f(x) - y)^2$

What to minimize over?

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

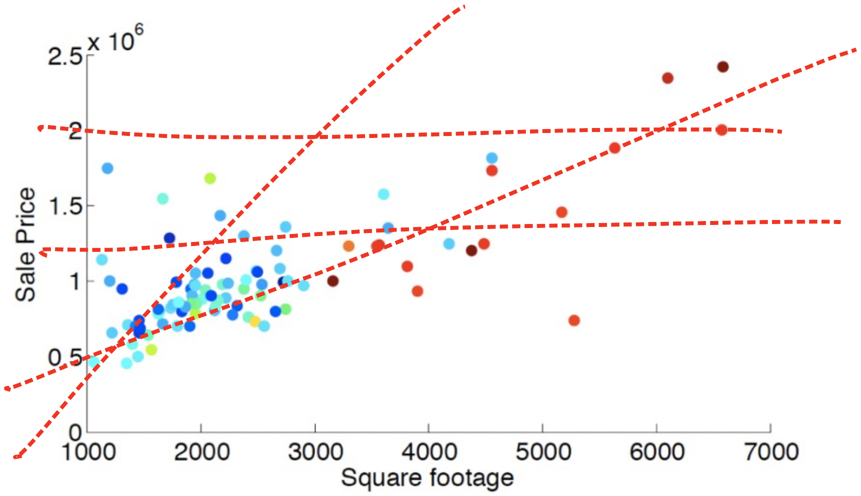
$$\hat{R}_S(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)$$

Function class

Def: A function class (or 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 predictor to generalize on unseen points.

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 used to test model

Generalization gap : Difference b/w test & training set errors

Generalization: More formally

Minimize loss over distribution of instances

Definition: 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:

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

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

<u>**Loss function:**</u> What is the right loss function for the task?

<u>**Representation:**</u> 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?

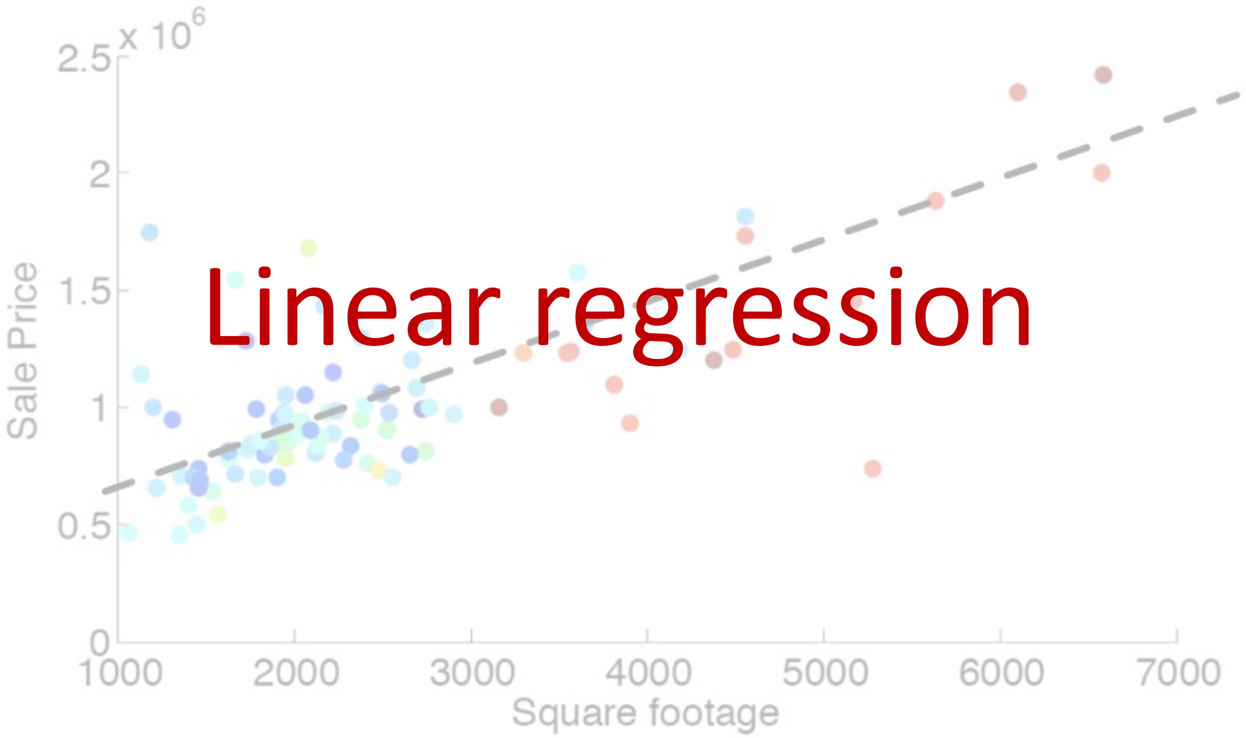
Representation: What class of functions should we use?

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

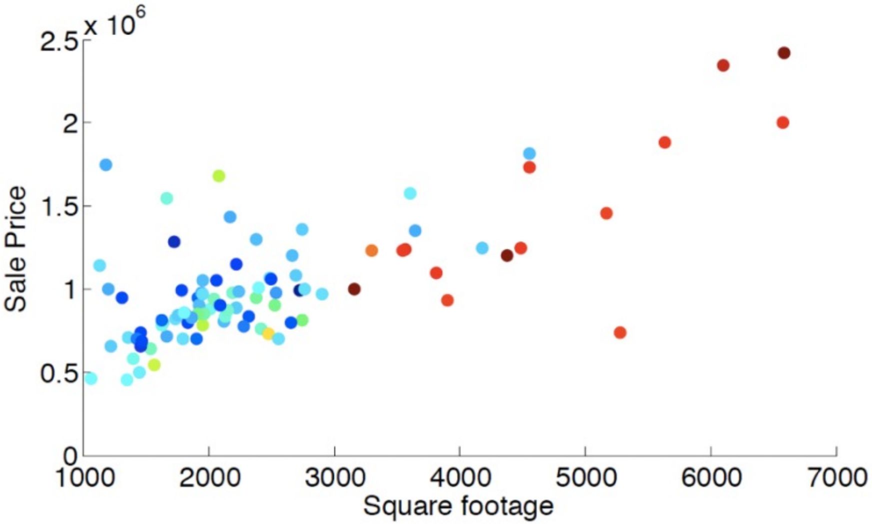
Generalization: Will the predictions of our model transfer gracefully to unseen examples?

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

Linear regression



House price prediction: the loss function



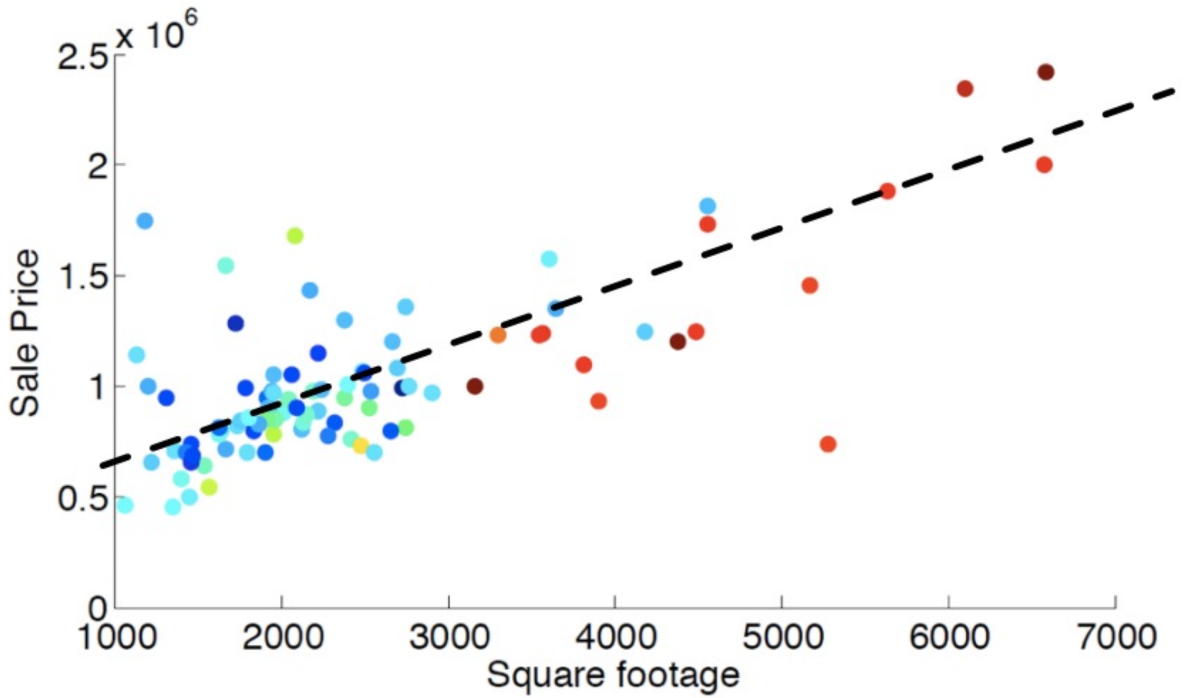
We're looking at real-valued outputs. Some popular loss functions:

- 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 $f : \mathbb{R}^d \rightarrow \mathbb{R}$ 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 $\widetilde{\boldsymbol{w}}$ = [$w_0, w_1, w_2, \ldots, w_d$ $]^T$ 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:

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

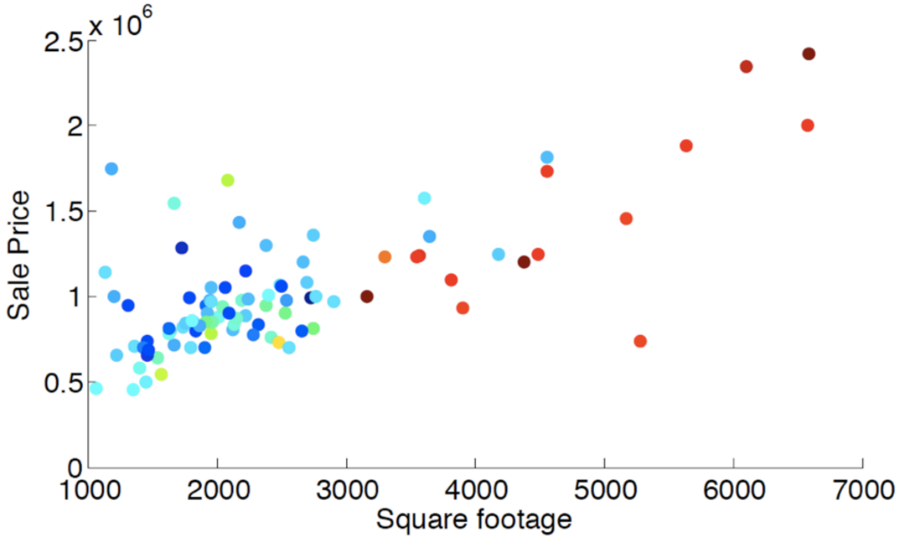
- Goal of empirical risk minimization:

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

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: $d = 0$

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

$$= n w_0^2 - 2\left(\sum_i y_i\right) w_0 + \sum_i y_i^2$$

$$= n\left(w_0 - \frac{1}{n}\sum_{i=1}^{n} y_i\right)^2 + (\text{constant wrt } w_0)$$

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

Warmup: $d = 1$

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

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

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

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

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

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

Warmup: $d = 1$

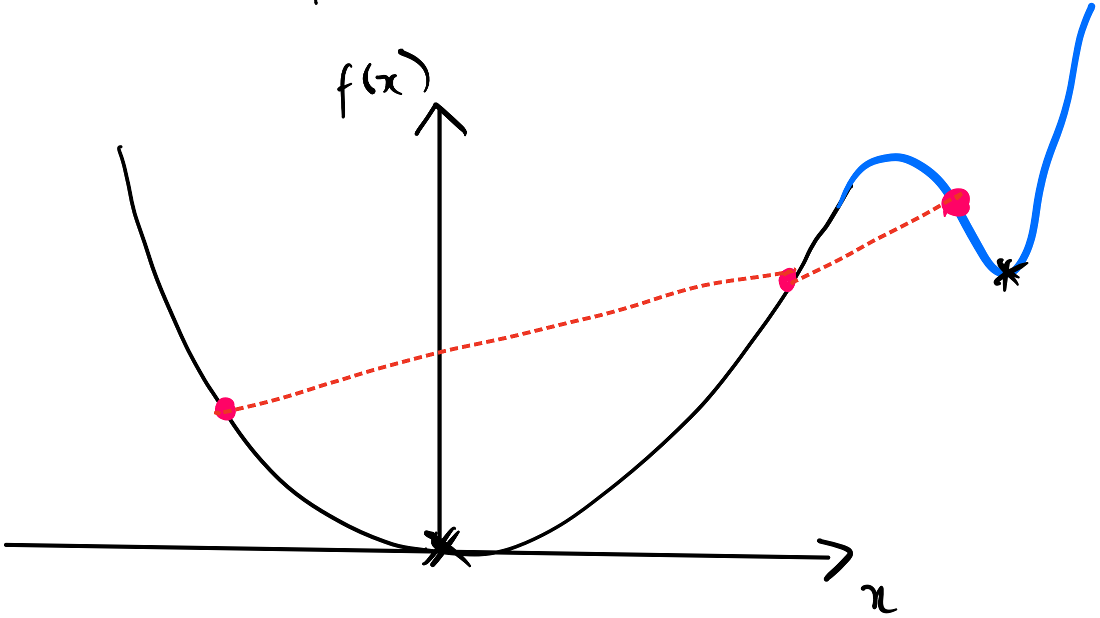
$$\begin{pmatrix} n & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} = \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{pmatrix}$$

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

Are stationary points minimizers?

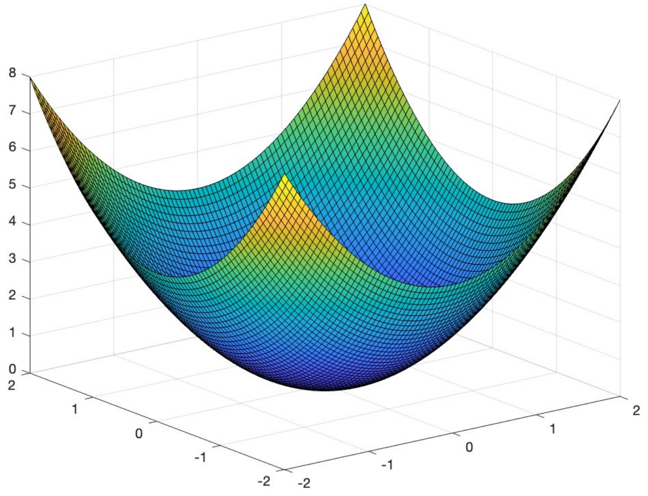
Yes, for ***convex*** objectives!

In 1-D: $f(x) = x^2$



$f''(x) \geq 0$

In high dimensions, this looks like:



$\nabla^2 F$ is positive-semi definite

General least square solution

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

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

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

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

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

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

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

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

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

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

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

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

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

Covariance matrix and understanding LS

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

(Here, we assume all features are centered to be 0 mean, i.e. $\sum_{i=1}^{n} \tilde{X}_{i,j} = 0$ for all features j, $\tilde{X}_{i,j}$ is the (i,j) entry of matrix $\tilde{X}$)

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

each weight w_j is just the covariane of the jth feature with the label

Another approach

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

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

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

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

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

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

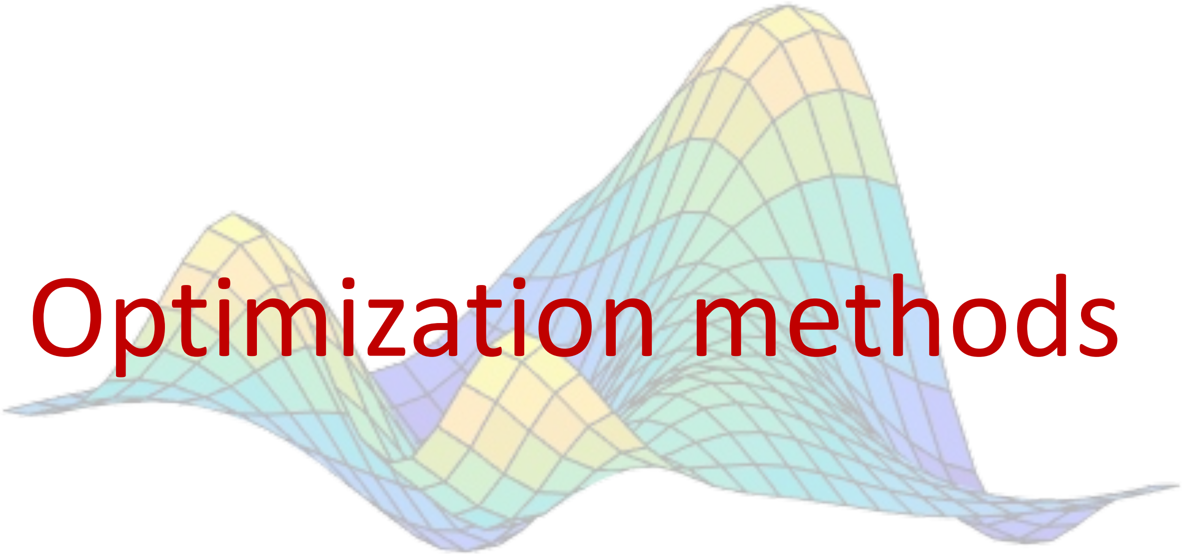
Bottleneck of computing

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

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

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 $t = 0, 1, \ldots$

$$w^{(t+1)} = w^{(t)} - \eta \nabla F(w)\big|_{w=w^{(t)}}$$

where $\eta > 0$ is called 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 [redacted] 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 $w_1^{(0)}$ and $w_2^{(0)}$ (to be 0 or *randomly*), $t = 0$
- do

$$w_1^{(t+1)} \leftarrow w_1^{(t)} - \eta \left[2(w_1 x^{(1)} + w_2 x^{(2)} - y) \cdot x^{(1)} \right]$$

$$w_2^{(t+1)} \leftarrow w_2^{(t)} - \eta \left[2(w_1 x^{(1)} + w_2 x^{(2)} - y) \cdot x^{(2)} \right]$$

$$t \leftarrow t + 1$$

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

Switch to Colab

optimization.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```
    this_theta[1] = last_theta[1] - eta * grad1
    theta.append(this_theta)
    J.append(cost_func(*this_theta))


# Annotate the objective function plot with coloured points indicating the
# parameters chosen and red arrows indicating the steps down the gradient.
for j in range(1,N):
    ax.annotate('', xy=theta[j], xytext=theta[j-1],
                arrowprops={'arrowstyle': '->', 'color': 'orange', 'lw': 1},
                va='center', ha='center')
ax.scatter(*zip(*theta), facecolors='none', edgecolors='r', lw=1.5)

# Labels, titles and a legend.
ax.set_xlabel(r'$w_1$')
ax.set_ylabel(r'$w_2$')
ax.set_title('objective function')

plt.show()
```