CSCI 567: Machine Learning

Vatsal Sharan

Fall 2022

Lecture 1, Aug 25

USC University of Southern California

Logistics

Course website: https://vatsalsharan.github.io/fall22.html

- Logistics, slides, homework etc.

Ed Discussion: https://edstem.org/

- Main forum for communication

DEN: https://courses.uscden.net/d2l/home/23403

- Recordings, homework submission

Prerequisites

This is a mathematically advanced class: that makes it more interesting!

(1) Undergraduate level training or coursework on linear algebra, (multivariate) calculus, and basic probability and statistics;
(2) Basic skills in 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), 1 late day per student (max 1 per HW)

Quizzes: **10/6** and **12/1** during lecture hours (5pm-7:20pm)

Project: Kaggle competition (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

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

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

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.

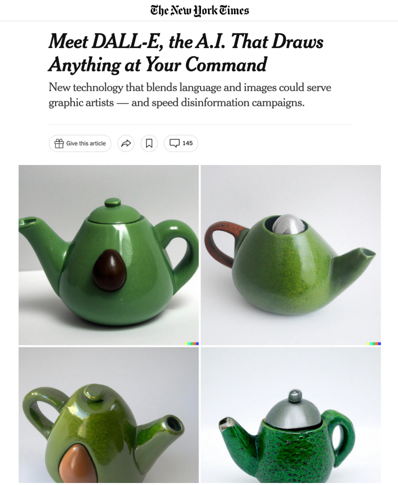
Give this article 608

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

New York Times, August 24, 2022

Look at some examples they mentioned in the article..

Image generation: Dall-E 2


The New York Times

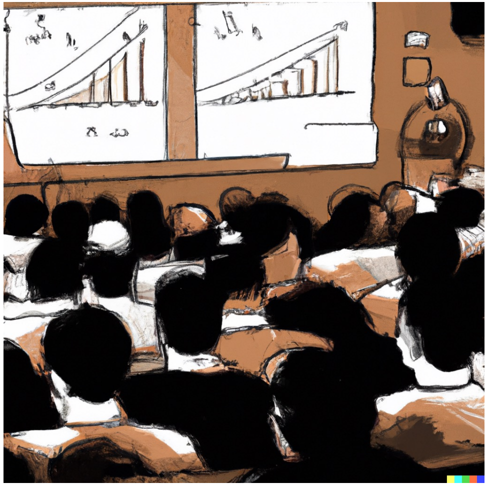
Meet DALL-E, the A.I. That Draws Anything at Your Command

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

Give this article 145

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.



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.

I gave the following prompt to a much simpler model:

Today 250 students at the University of Southern California begin their machine learning class. What all will they learn??

Today 250 students at the University of Southern California begin their machine learning class. What all will they learn??

Machine learning is a fascinating subject that has been at the forefront of computer science for quite some time now. We're also seeing some new interest from those within the field. Here is one of our students here who has been working on machine learning for a number of years.

This is Dr. William Smith. He works on machine learning at the University of Southern California. He is a professor in computer science and the founder of the University of Southern California. It's a fantastic position to be part of the research community.

At the University of Southern California, you have a wide range of research interests. So, what's the most interesting research area to you?

GPT-2-simple response using https://watt-ai.github.io/

Game playing: AlphaGo

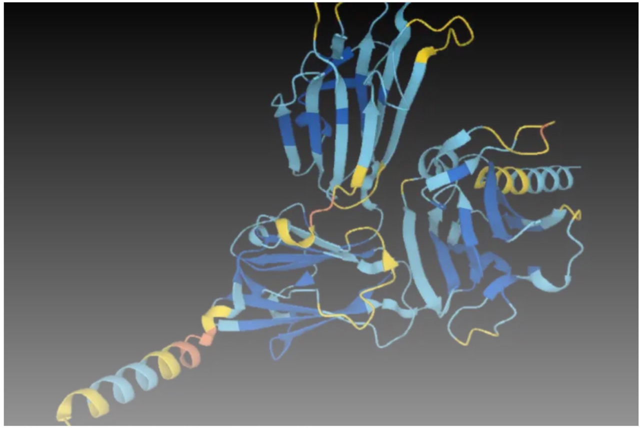
Protein folding: AlphaFold

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 hand-on programming tasks
- Prepare you for studying advanced machine learning techniques

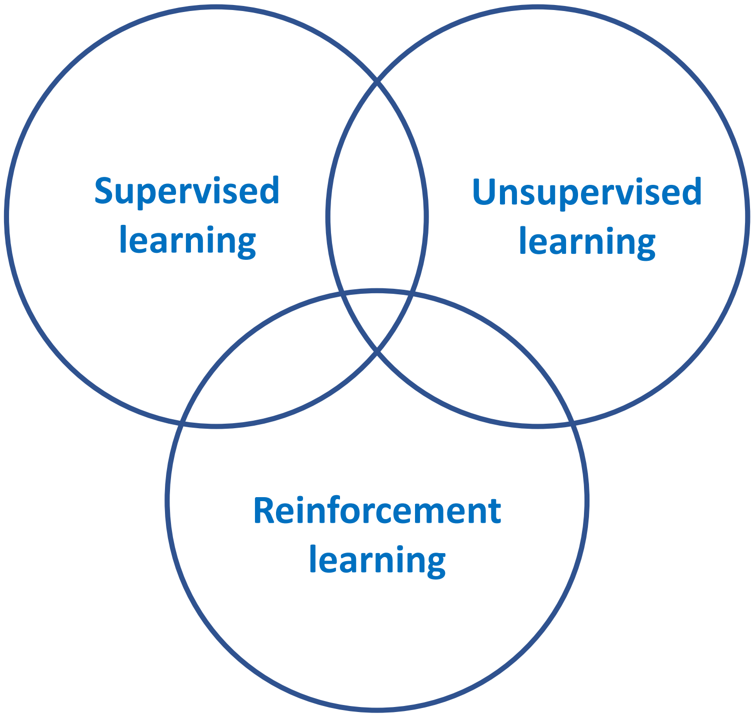
A simplistic taxonomy of ML

Supervised learning:
Aim to predict outputs of future datapoints

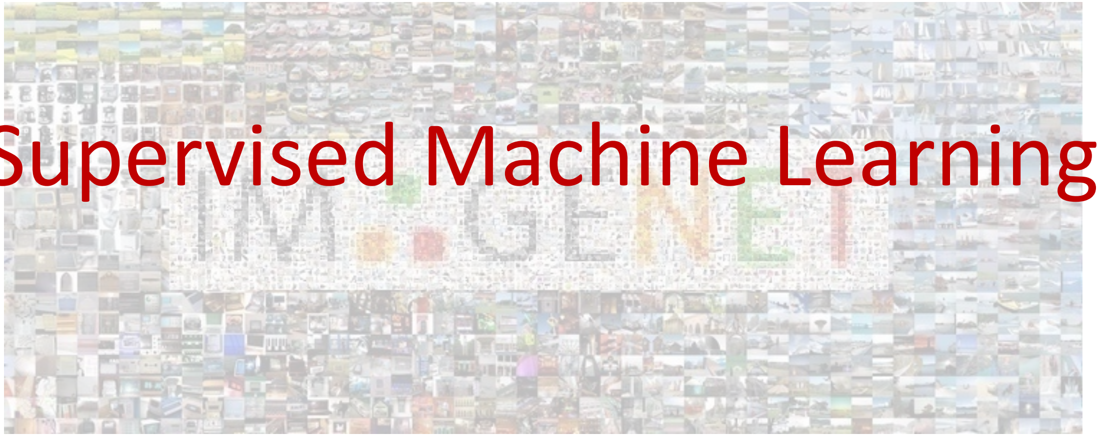
Unsupervised learning:
Aim to discover hidden patterns and explore data

Reinforcement learning:
Aim to make sequential decisions

A simplistic taxonomy of ML



Supervised Machine Learning



Supervised ML: Predict future outcomes using past outcomes

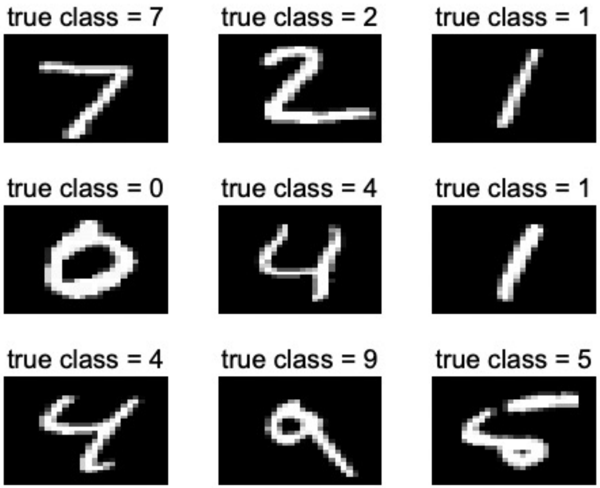
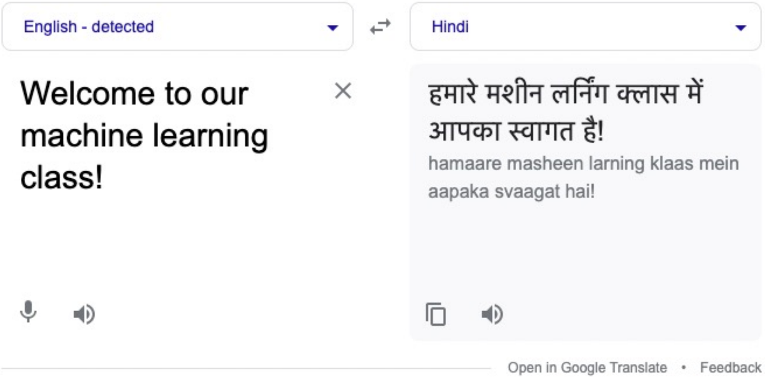
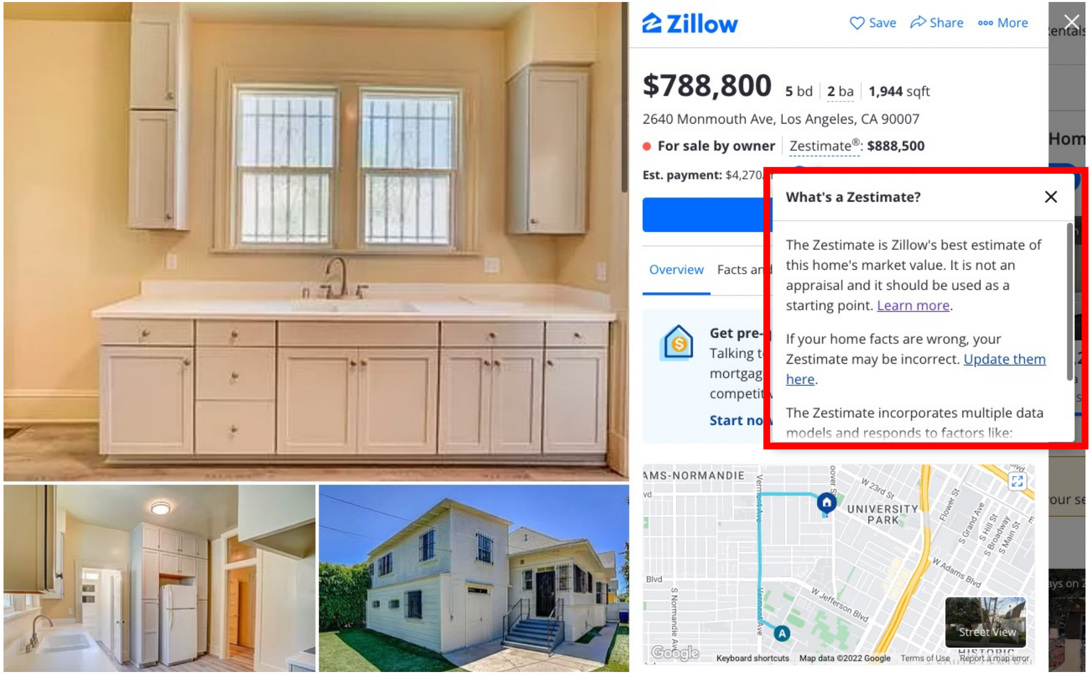


Image classification



Machine translation

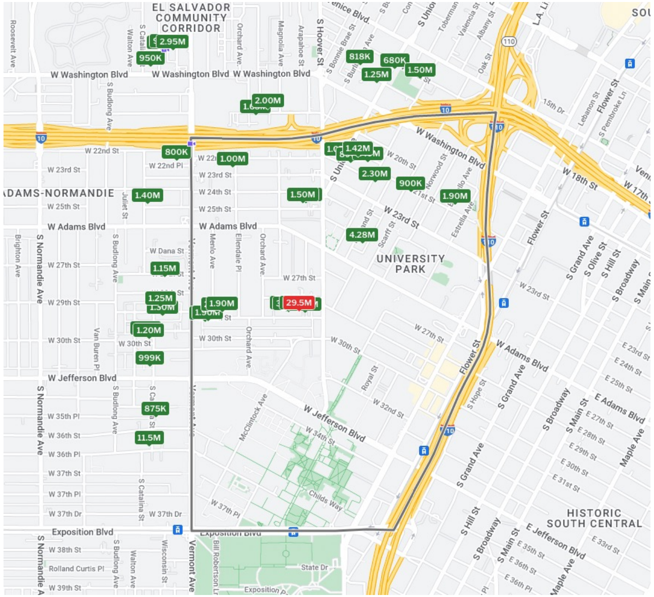
Supervised ML: Predict future outcomes using past outcomes



Predicting sale price of a house

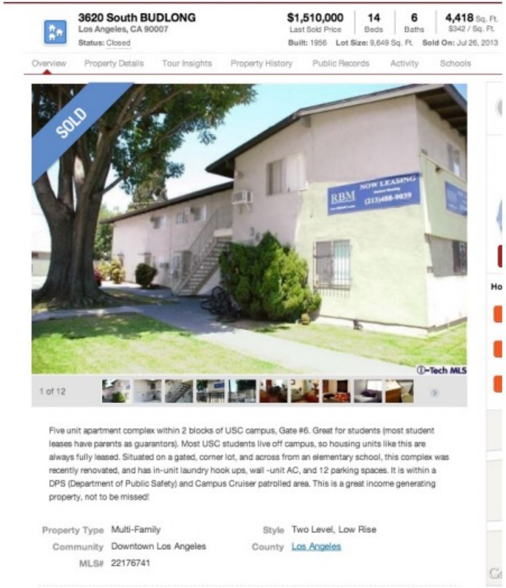
Simplistic version: Predicting sale price of a house

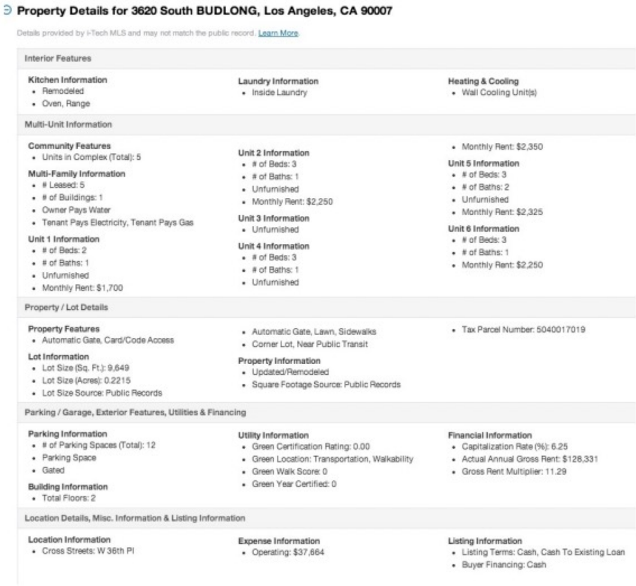
Retrieve historical sales records (training data):



Simplistic version: Predicting sale price of a house

Features used to predict:





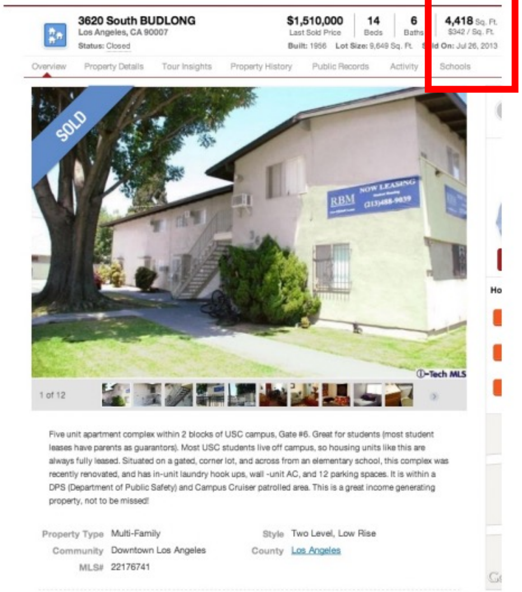
Simplistic version: Predicting sale price of a house

Features used to predict:

Numeric data

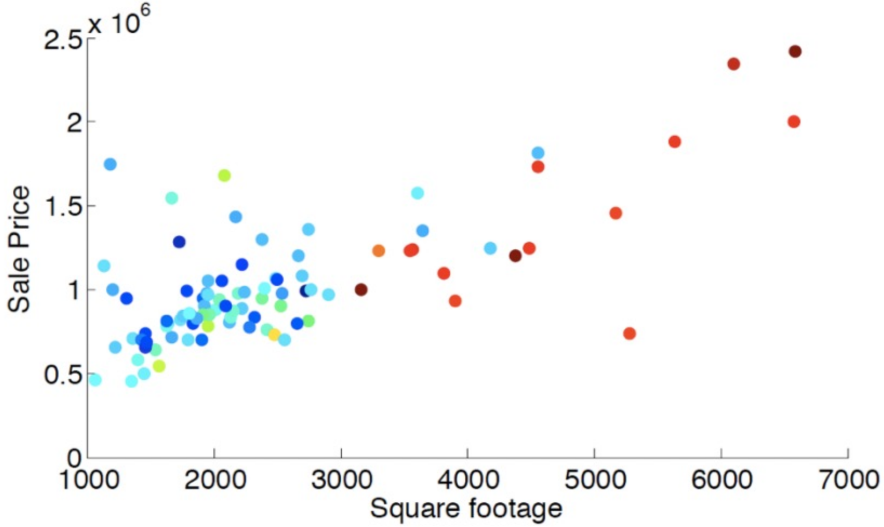
Free-form text

Categorical data



Simplistic version: Predicting sale price of a house

Correlation between square footage and sale price:

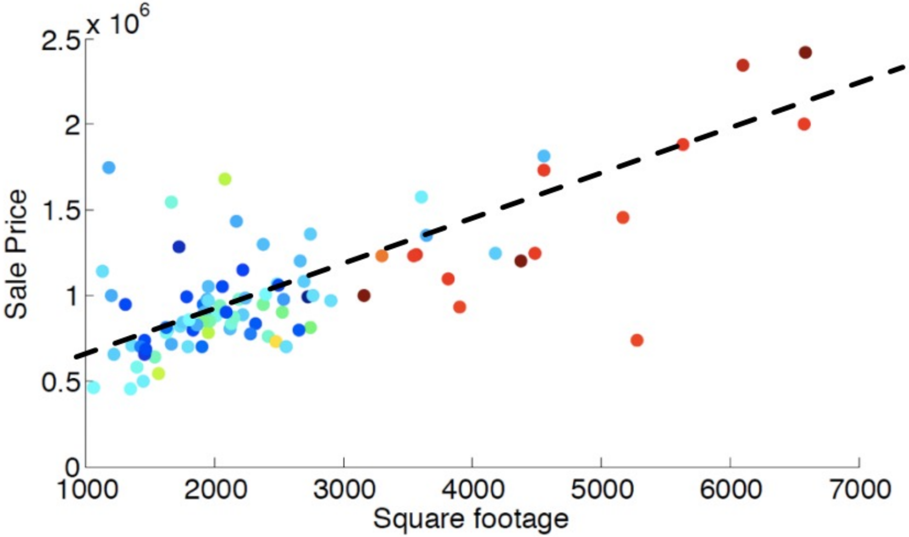


Simplistic version: Predicting sale price of a house

Possibly linear relationship:

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

(slope) *(intercept)*



General framework for supervised learning

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

* Datapoints in d dimensions
* In previous e.g., $d = 1$ (sq. footage)

Feature engineering !

→ An output space : $\mathcal{Y}$

* $\mathcal{Y} \in \mathbb{R}$ for sale price prediction
* $\mathcal{Y} \in \{\pm 1\}$ for binary classification

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

Loss function : $\ell(f(x), y)$. Depends on the task.

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

What to minimize over?

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

Definition : Risk of predictor $f(x)$ is :

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

Challenge : Don't know D

* i.i.d. assumption : We assume that we have a set of labelled instances drawn independently & identically (i.i.d.) from distribution D.

* theoretical abstraction, often useful.

Pay attention to whether this is valid! (need "stationarity")

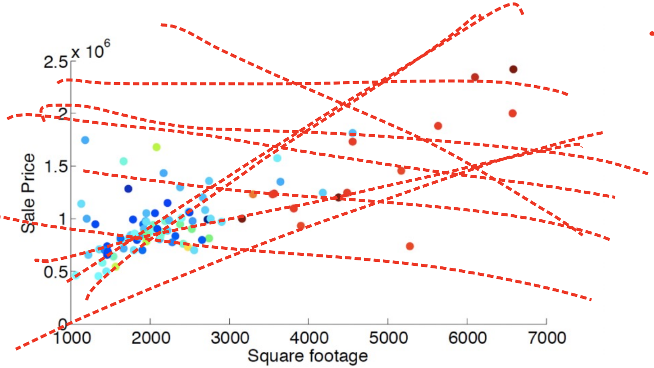
Definition: Given a set of labelled datapoints $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ the empirical risk of any $f: \mathcal{X} \to \mathcal{Y}$ w.r.t. S is $\hat{R}_S(f) = (1/n) \sum_{i=1}^{n} \ell(f(x_i), y_i)$.

Function class

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

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

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 $\mathcal{F} = \{ f : \mathcal{X} \to \mathcal{Y} \}$ & set of labelled datapoints S, ERM corresponds to

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

Generalization

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 known as the generalization gap.

Generalization: How well does our predictor "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.
test set - a " " to test model.

Ideally: only use test set once (or a few times)

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?

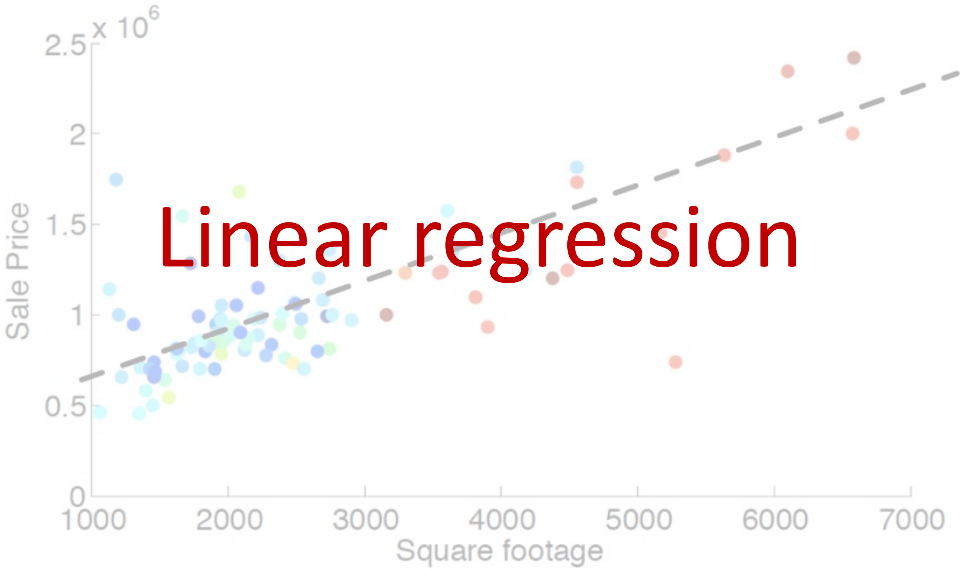
Representation: What class of functions should we use?

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

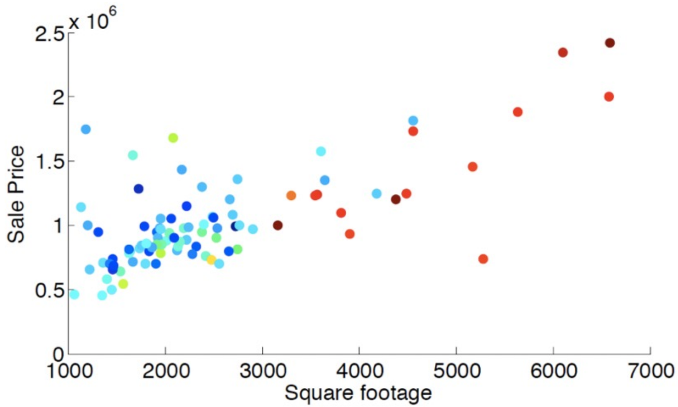
Generalization: Will the predictions of our model transfer gracefully to unseen examples?

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

* **squared error**:

(prediction - sale price)^2 (most common)

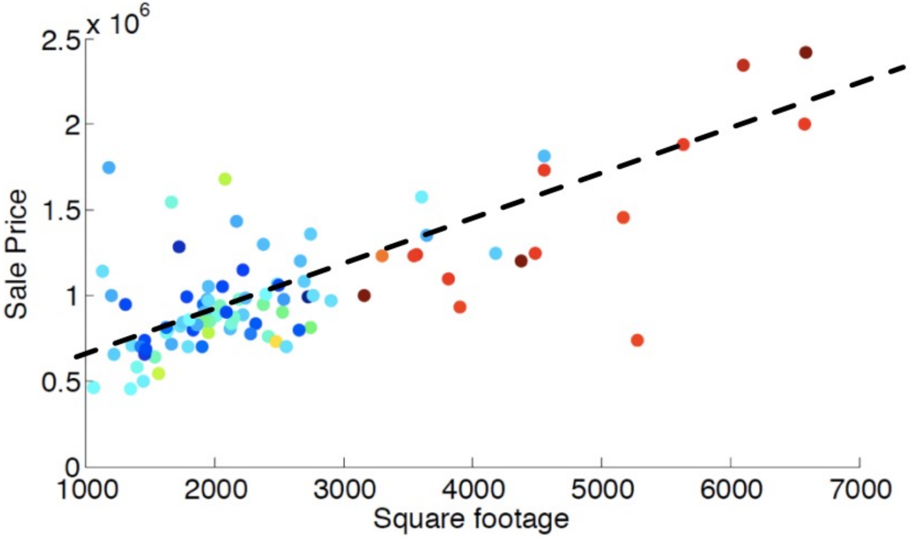
* **absolute error**:

| prediction − 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
$\cdots$	$\cdots$	$\cdots$	$\cdots$
Total			$0 + 67^2 + 228^2 + 740^2 + \cdots$

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

Formal setup for linear regression

Input : $x \in \mathbb{R}^d$

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

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

Linear model : $f : \mathbb{R}^d \to \mathbb{R}$ with $f(x) = w_0 + \sum_{i=1}^{d} w_i x_i$

$= w_0 + w^T x$

* $w = [w_1 \ w_2 \ \ldots \ w_d]^T$ (weights, weight vector)
* bias w_0

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

Minimize total squared error

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

Define (Residual sum of squares):

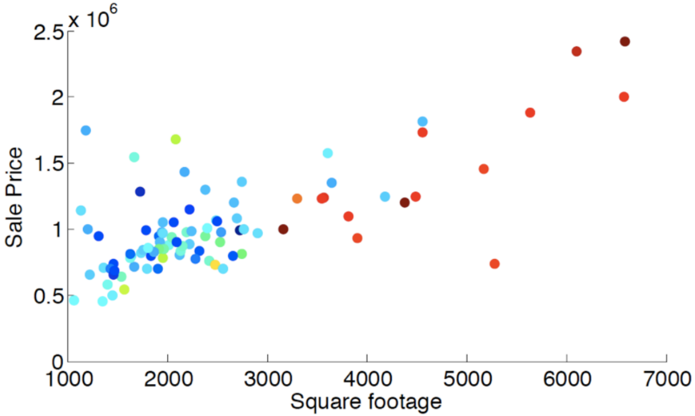
$$RSS(\tilde{w}) = n\,\hat{R}_S(\tilde{w}) = \sum_i \left(\tilde{x}_i^T \tilde{w} - y_i\right)^2$$

ERM: find $\tilde{w}^* = \underset{\tilde{w} \in \mathbb{R}^{d+1}}{\operatorname{argmin}}\ RSS(\tilde{w})$

Known as 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: $d = 0$

constant means that the term is not dependent on what we're optimizing over (w_0 here)

$$RSS(w_0) = \sum_i (w_0 - y_i)^2$$

$$= n w_0^2 - 2\left(\sum_i y_i\right) w_0 + \text{const}$$

$$= n\left(w_0 - \frac{1}{n}\sum_i y_i\right)^2 + \text{const}$$

(completion of squares)

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

Think about what should be the solution for absolute error ($\ell(f(x), y) = |f(x) - y|$)

Warmup: $d = 1$

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

General approach: find stationary point

(i.e. points with zero gradient)

$$\frac{\partial RSS(\tilde{\omega})}{\partial w_0} = 0 \implies \sum_i (w_0 + + w_1 x_i - y_i) = 0$$

$$\frac{\partial RSS(\tilde{\omega})}{\partial w_1} = 0 \implies \sum_i (w_0 + w_1 x_i - y_i) x_i = 0$$

Warmup: $d = 1$

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

(a linear system)

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

(assuming investible)

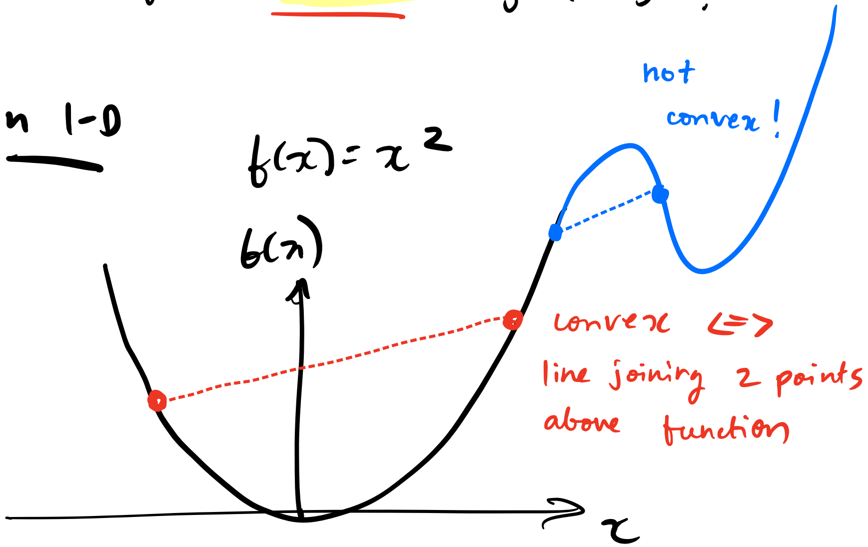
Are stationary points minimizers?

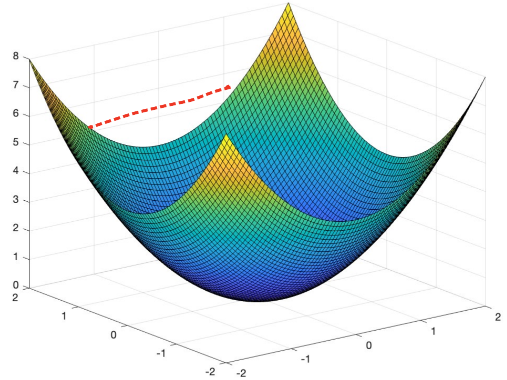
Yes, for convex objectives!

In high dimensions:

In 1-D

$f(x) = x^2$





$f''(x) \geq 0 \ \forall x \longrightarrow \nabla^2(f(x))$ is positive semi-definite (psd)

General least square solution

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

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

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

proportional to

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

$$= (\tilde{X}^T \tilde{X}) \tilde{w} - \tilde{X}^T Y = 0$$

$$\tilde{X} = \begin{pmatrix} \text{———} \; \tilde{x}_1^T \; \text{———} \\ \text{———} \; \tilde{x}_2^T \\ \vdots \\ \text{———} \; \tilde{x}_n^T \; \text{———} \end{pmatrix} \in \mathbb{R}^{(n \times (d+1))}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^n$$

$$(\tilde{X}^T \tilde{X})\, \tilde{w} = \tilde{X}^T Y \implies \tilde{w}^* = \underbrace{(\tilde{X}^T \tilde{X})}_{\text{invertible}}{}^{-1} \tilde{X}^T Y$$

Covariance matrix and understanding LS

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

↳ each row sums to 0

suppose each feature is 0-mean

$\tilde{X}^T \tilde{X}$: covariance matrix

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

each weight is the covariance of the feature with label y. Highly correlated features have higher weights.

Another approach

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

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

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

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

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

completion of squares

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

(running time)

Bottleneck of computing

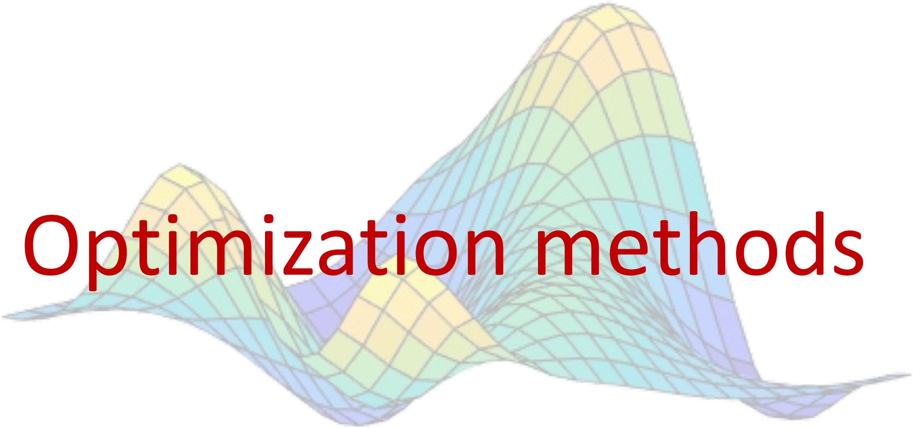
$$\widetilde{\boldsymbol{w}}^* = (\widetilde{\boldsymbol{X}}^T\widetilde{\boldsymbol{X}})^{-1}\widetilde{\boldsymbol{X}}^T\boldsymbol{y}$$

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

Takes time $O(d^3)$

This can be quite expensive in high dimensions

Optimization methods



Problem setup

Given: a function $\mathrm{F}(\boldsymbol{w})$

Goal: minimize $\mathrm{F}(\boldsymbol{w})$ (approximately)

Two simple yet extremely popular methods

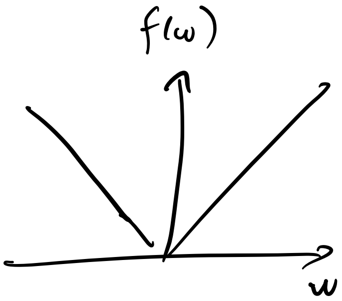
Gradient Descent (GD): simple and fundamental

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 with some $w^{(0)}$. For $t = 0, 1, \ldots T$

$$w^{(t+1)} \leftarrow w^{(t)} - \eta \nabla F(w^{(t)})$$

where $\eta > 0$ is called step size (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

Example: $F(\boldsymbol{w}) = 0.5(w_1^2 - w_2)^2 + 0.5(w_1 - 1)^2$. Gradient is

$$\frac{\partial F}{\partial w_1} = 2(w_1^2 - w_2)w_1 + w_1 - 1 \qquad \frac{\partial F}{\partial w_2} = -(w_1^2 - w_2)$$

GD:

- Initialize $w_1^{(0)}$ and $w_2^{(0)}$ (to be 0 or *randomly*), $t = 0$
- do

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

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

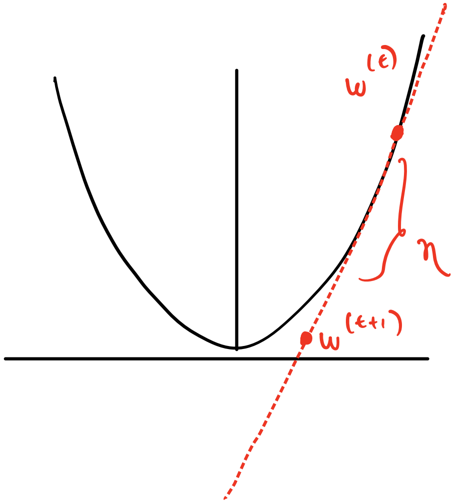
$$t \leftarrow t + 1$$

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

Why GD?

Intuition: First-order Taylor approximation

$$F(w) \approx F(w^{(t)}) + \nabla F(w^{(t)})^T (w - w^{(t)})$$



$$F(w^{(t+1)}) \approx F(w^{(t)}) - \eta \underbrace{\|\nabla F(w^{(t)})\|_2^2}_{\geq 0}$$

$$\Rightarrow F(w^{(t+1)}) \overset{\approx}{\leq} F(w^{(t)})$$

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

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()
```