Week 1: Intro to ML, Linear Regression

OVERVIEW OF ML

eg.
Search ranking, image recognition, recommendations, voice recognition, spam detection, energy optimization

This course < algorithms + details of how they work
tips and tricks, practical advice

Historically, traditional AI approches had trouble solving the most complex problems.
↳ ML a path to learn the solution

AGI (Artificial General Intelligence) is a dream for many. How long? Unknown.

ML disrupting industries but also creating demand and new jobs.

SUPERVISED vs. UNSUPERVISED

What is ML? Gives computer ability to learn without being explicitly programmed.

Two main types < Supervised (Most real applications)
Unsupervised

We'll also spend a lot of time on best practices.

Supervised Learning

Learns $X \rightarrow y$, input to output label mappings from being given right answers.

Examples: input (X) → output (Y)

email → spam? — spam filtering
audio → text — speech recognition
English → Spanish — machine translation
(ad, user) → click? — online advertising
(img, radar) → position — self-driving car
image → defect? — visual inspection

(right answers)
Given many pairs (X, Y), try to learn how to predict Y for an unseen X.

1. Regression (inf. set of #s): Predict numerical val. (e.g. cost $) from inputs.
How to choose? Diff. algs. exist.

2. Classification (finite set of classes): Predict {categorical vals} from input. (incl. booleans, e.g. cancer?) (aka classes)

Note there can be multiple inputs!

Unsupervised Learning

Given training data, but no labels.
Goal is not to predict labels, but to discover structure or patterns.

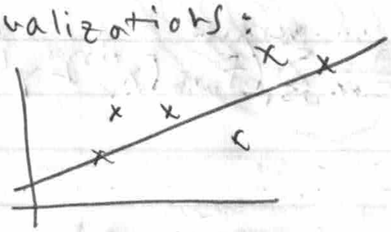
E.g. clustering (groups similar data) — Google News, DNA analysis, customer groups by intent
anomaly detection — find unusual data points
dimensionality reduction — compress data w/ smaller numbers

LINEAR REGRESSION

Fitting a straight line w/one variable (univariate lin. reg.)
Examples:
- Portland house price (y) vs. size sq. ft. (x)
  Input data: $\{(x,y)\}$ for recent sales
  Model: Fit line $y = b + w_0 x$
  Predict: y for incoming x

(Recall: supervised learning predicting $\mathbb{R}$ (inf.) = regression,
supervised predicting finite {classes} = classification)

Visualizations:



plot

x	y
123	456
etc.	

data table

Terminology:
- Training set = data used to train the model
- x = input variable = feature
- y = target = output variable

$\}$ (x,y) = single training example
- m = number of training examples
- f = hypothesis = function estimate = model
- $\hat{y}$ = prediction = estimated output for input = $f(x)$

How to represent f?
→ For lin. reg., $\boxed{f_{w,b}(x) = wx + b} = f(x)$ where we choose w, b (weights) (coefficients) (parameters)

Training algorithm produces f from training set

$\boxed{\text{Cost Function}}$: tells us how well the model is doing

We want to choose f (i.e. w and b for lin. reg.) so that it fits the data well. Cost fn. compares $\hat{y}$ and y = error for various examples in training set.

$$\boxed{\frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 = J(w,b)}$$ = squared error cost fn.

(most common in regression problems)

model: $f_{w,b}(x) = wx+b$

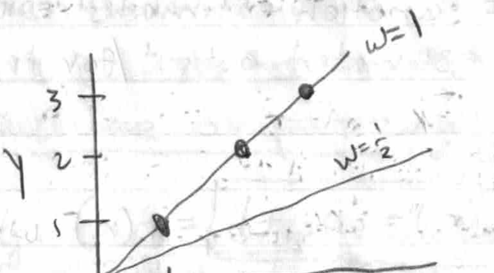
parameters: w, b

cost fn: $J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2$

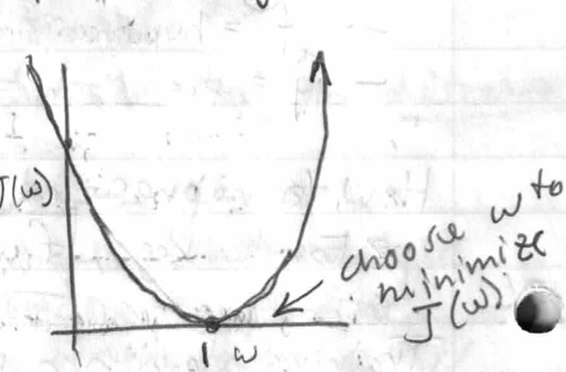
goal: $\min_{w,b} J(w,b)$

Note that for $f_{w,b}$ the params w,b are fixed and so f is a function of x alone, whereas J is a function of w,b w.r.t. fixed training set $(x^{(i)}, y^{(i)})$ for $i \in [1,m]$.

Consider training data $\{(1,1),(2,2),(3,3)\}$ and keep $b=0$ fixed. Then for lin. reg.:

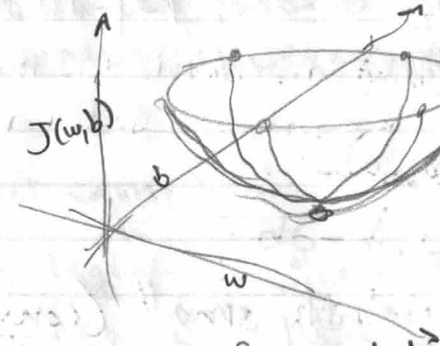


$f_{w,b}(x)$



$J(w)$

For full $J(w,b)$ the visualization is 3d:



For linear regression the shape is bowl-like

surface plot or wirefrant plot

each line corresponds to a fixed $J(w,b)$ with variable w,b

contour plot

GRADIENT DESCENT

Want $\min_{w,b} J(w,b)$ or more generally $\min_{b,w_i} J(b, w_1, \dots, w_n)$

1. Start w/ $w, b, \dots n = 0$
2. Keep updating w, b to make J smaller
3. Stop when reaching ~minimum

How? Take grad = ∇ of f at w,b and step in neg. direction. This is the direction of fastest descent. Repeat until $\nabla f(w,b) \approx 0$.

↳ Note that this reaches a local minimum

↳ So starting point might matter!

(Not for simple lin. reg. though)

As an algorithm:

Given learning rate $\alpha \in \mathbb{R}$, repeatedly:

$$dw = \alpha \frac{\partial}{\partial w} J(w,b)$$

$$db = \alpha \frac{\partial}{\partial b} J(w,b)$$

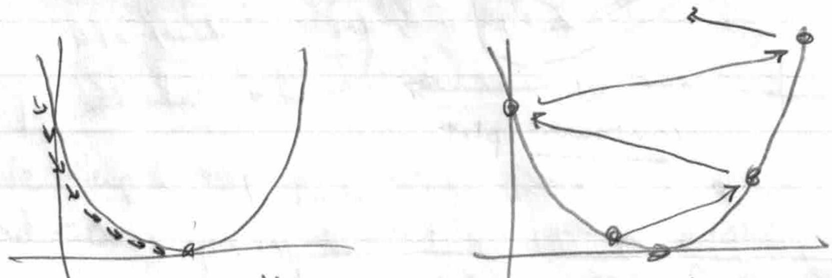
$$w, b = w - dw,\ b - db$$

until dw, db sufficiently small (convergence)

Learning Rate has a huge impact on convergence —

→ too small: convergence is super-slow

→ too large: might diverge!



too small too large

For linear regression using squared-error cost fn:

$$\frac{\partial}{\partial w} J(w,b) = \frac{\partial}{\partial w} \frac{1}{2m} \sum_{i=1}^{m} \left(f_{w,b}(x_i) - y_i\right)^2$$

$$= \frac{\partial}{\partial w} \frac{1}{2m} \sum_{i=1}^{m} \left(w x_i + b - y_i\right)^2$$

$$= \frac{1}{\cancel{2}m} \sum_{i=1}^{m} \cancel{2} \left(w x_i + b - y_i\right) w \quad \text{(chain rule)}$$

$$\boxed{\frac{\partial J(w,b)}{\partial w} = \frac{1}{m} \sum_{i=1}^{m} \left(f(x_i) - y_i\right) x_i}$$

similarly, $\boxed{\frac{\partial}{\partial b} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \left(f(x_i) - y_i\right)}$

(Note that the mean-squared cost fn. for linear regression is convex and so will always converge to the global minimum.)

But compare:

- Batch gradient descent: use all training examples at each step of gradient descent
- others that don't! (e.g. stochastic)