Linear Regression in Python

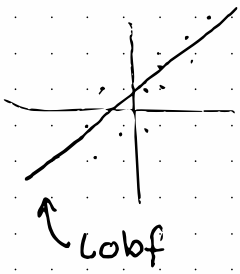
Objective: find line of best fit through a set of points

↳ eventually add more variables

What is ML: predict outcomes from past examples

- supervised $(X \rightarrow y)$
- unsupervised (X) ← learn structure of the data.

supervised → Classification
→ Regression



correlation ≠ causation

↳ randomised control trials

↳ synthesis of many studies (meta-studies)

example of linear equation: $V = IR$ (resistance)

where measuring the V (voltage) & I (current) is what we are changing

↳ we can figure out R by taking measurements & finding slope

if we have $\vec{x}$ & $\vec{y}$ where const relating $\vec{x}$ to $\vec{y}$ is $\hat{y}$

to $\hat{y}_i = ax_i + b$

How do we think of error between $\hat{y}$ & $\vec{y}$?

$E = \sum_{i=1}^{N} (y_i - \hat{y}_i)$?

↳ No, error would be 0 when it's not on line

∴ take abs or sq error

$$MSE = \frac{1}{N} \sum_{x=i}^{N} (y_i - \hat{y}_i)^2$$

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|$$

How do we optimise these equations?

$$E = \sum_{i=1}^{N} [y_i - (ax_i + b)]^2$$

← need to find deriv. wrt a & b

$$\frac{\partial E}{\partial a} = \sum_{i=1}^{N} 2(y_i - (ax_i + b)) * (-x_i)$$

optimise, so $\frac{\partial E}{\partial a} = 0$

$$\therefore \sum_{i=1}^{N} -y_i x_i + a x_i^2 + b x_i = 0$$

$$\therefore \sum_{i=1}^{N} a x_i^2 + b x_i = \sum_{i=1}^{N} y_i x_i$$

$$\frac{\partial E}{\partial b} = 0 = \sum_{i=1}^{N} 2(y_i - (ax_i + b)) * (-1)$$

$$\therefore \sum_{i=1}^{N} a x_i + bN = \sum_{i=1}^{N} y_i$$

doesn't seem to work...

Solve using Linear Algebra

or

$$a = \frac{\bar{xy} - \bar{x}\,\bar{y}}{\bar{x^2} - \bar{x}^2} \qquad b = \frac{\bar{y}\,\bar{x^2} - \bar{x}\,\bar{xy}}{\bar{x^2} - \bar{x}^2}$$

Notation is not clear

Note that $\vec{a} \cdot \vec{b} = \vec{a}^T \vec{b}$

How did we arrive at these results:

1. define equation to estimate
2. define cost; defines "hyperplane"
3. find parameters that minimize cost.
   ↳ lowest cost = solution

> Note: optimization cannot work like this if problem is not convex

→ Gradient descent

init w/ random vars a, b

↳ predict $\hat{y} = aX + b$

find loss $= \frac{1}{N} \sum_{i=1}^{N} (y - \hat{y})^2$

need to update based on error

$$\frac{\partial \hat{y}}{\partial a} \left[\frac{1}{N} \sum_{i=1}^{N} (y - aX - b)^2 \right]$$

$$= \frac{2}{N} \sum_{i=1}^{N} (y - aX - b)(-X)$$

$$\frac{\partial \hat{y}}{\partial b} = \frac{2}{N} \sum_{i=1}^{N} (y - aX - b)(-1)$$

Update based on grad.

$$a -= \alpha * \frac{d\hat{y}}{da} \quad \& \quad b -= \alpha * \frac{d\hat{y}}{db}$$

yay! I can't believe we did db!

R^2 values → defined as $$R^2 = 1 - \frac{\sum_{i=1}^{N}(y-\hat{y})^2}{\sum_{i=1}^{N}(y-\bar{y})^2}$$

- $R^2 = 0$: prediction is the same as taking the mean
- $R^2 = 1$: prediction is perfect
- $R^2 = -1$: prediction gives more error than taking the average.

Multiple Linear Regression

↳ normally multiple effects influence the dependant variable

↳ 1-d : $\hat{y} = \theta^T \vec{x} + b$

↳ $\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

↳ New optimal solution

$$\frac{\partial E}{\partial \theta_j} = \sum_{i=1}^{N} 2(y_i - \theta^T x_i)(-x_{ij}) = 0$$

(y_i: vector; x_{ij}: scalar)

$\nabla b - \frac{\partial(\theta^T x_i)}{\partial \theta_j}$

All other θ are 0 because they are const. $\Rightarrow = -\frac{\partial(\theta_j x_{ij})}{\partial \theta_j}$

↳ expand: $\sum_{i=1}^{N} y_i(-x_{ij}) - \sum_{i=1}^{N} \theta^T x_i(-x_{ij}) = 0$

we do this to try & solve for θ →

$$\therefore \sum_{i=1}^{N} y_i x_{ij} = \sum_{i=1}^{N} \theta^T x_i x_{ij}$$

(θ^T independent of i)

equation exists for every θ_j

j

1 $\quad \theta^T \sum_{i=1}^{N} x_i x_{i1} = \sum_{i=1}^{N} y_i x_{i1}$

2 $\quad \theta^T \sum_{i=1}^{N} x_i x_{i2} = \sum_{i=1}^{N} y_i x_{i2}$

$\vdots$

D $\quad \theta^T \sum_{i=1}^{N} x_i x_{iD} = \sum_{i=1}^{N} y_i x_{iD}$

$\Rightarrow \theta^T(X^T X) = y^T X$

Hallelujah!

$$\theta^T(X^TX) = y^TX$$

$$(X^TX)\theta = X^Ty$$

$$\theta = (X^TX)^{-1}X^Ty$$

How to solve multiple LR using matrices

Same as last section only entirely in matrix form.

$$J = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

$$= (Y - \hat{Y})^T(Y - \hat{Y}) \qquad Y = X\theta$$

$$= Y^TY - Y^T\hat{Y} - \hat{Y}^TY + \hat{Y}^T\hat{Y}$$

$$= (X\theta)^T(X\theta) - (X\theta)^T\hat{Y} - \hat{Y}^T(X\theta) + (\hat{Y})^T\hat{Y}$$

$$\frac{\partial J}{\partial \theta} = 0 = -2X^T\hat{y} + 2X^TX\theta$$

$$\therefore \theta = (X^TX)^{-1}X^T\hat{y}$$

Shape of Θ

$$\Theta = (X^T X)^{-1} X^T y$$

Say X 100×2, y 100×1

$I = X^T X = 2 \times 2$

$I X^T = (2 \times 2, 2 \times 100) = 2 \times 100$

$\therefore (X^T X)^{-1} X^T y = (2 \times 100, 100 \times 1) = \boxed{2 \times 1}$

$\therefore \Theta = D \times 1$ ← vector length of dimensions

... What about constant?

↳ add extra column to X made of 1

Polynomial Regression

Why is this still linear?

What we predict is a linear combo of weights (θ). Thus, x can change how we want.

Not great my code for this but multiple APIs are capable of this calculation

Interpreting the weights

↳ $y = mx + b$ ← fudge factor that is always present.

↑ "for every increase in x, y is incremented by m."

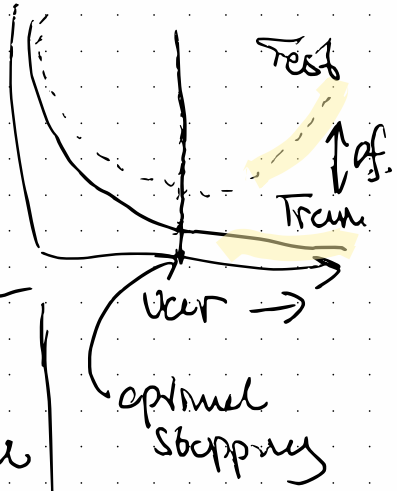
Generalization

Why not just fit data as closely as possible?

↳ doesn't generalize & noise is captured

↳ can't predict the future, only the past.

Train on 80% of data
↳ Test on 20% for generalization



1. Train] Train model
2. Test
3. Valid ← use this when model is best to use

What about categorical vars?
ie degree type: B, M, P

is-B	is-M	is-P
0	0	1
0	1	0
1	0	0

↑ bool values

Probabilistic Interpretation of Sq. error

Show LR is max. likelihood solution to b.o.b

↳ $E = \sum_{i=1}^{N} (y_i - \hat{y})^2$

What is max. likelihood
↳ "What answer is most likely to be true"

ie, what is mean of gaussian dist?

$$P(x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x_i-\mu)^2}{2\sigma^2}\right)$$

If we measure heights, we know they're independent & identically distributed

↳ not related & come from same dist.

$$\therefore P(X) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma}} \exp\left(\frac{-(X_i-\mu)^2}{2\sigma^2}\right)$$

$$= P(X|\mu)$$

↑ prob of data given μ

we want to find μ such that $P(X)$ is max.

$$\frac{\partial P(X)}{\partial \mu} = 0?$$ super hard as $N \uparrow$

$$\therefore \frac{\partial \log(P(X))}{\partial \mu} = 0$$ ✓

So... $L(X|\mu) = \log\left[\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(x_i-\mu)^2}{2\sigma^2}\right)\right]$

$$= \sum_{i=1}^{N}\left[\frac{-1}{2}\log(2\pi\sigma^2) - \frac{1}{2}\frac{(x_i-\mu)^2}{\sigma^2}\right]$$

where $L(x) = \log[P(x)]$

$$\frac{\partial L(X|\mu)}{\partial \mu} = \sum_{i=1}^{N} \frac{-2}{2} * \frac{(x_i-\mu)}{\sigma^2} * (-1)$$

$$= \sum_{i=1}^{N} \frac{(x_i-\mu)}{\sigma^2} = 0$$

solve for μ... $\sum_{i=1}^{N} x_i = \sum_{i=1}^{N} \mu$

$$\therefore \mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

we do this because we summed μ N times!

Moral of story?

w/ LR minimizing cost is equal to max. log likelihood.

↳ y is gaussian distb across independent vars.

L2 regularisation "Ridge Regression"

MSE is vulnerable to outliers due to exp. in loss function. How to fix?

↳ penalize large weights

$$J = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda |\theta|^2$$

$|\theta|^2 = \theta^T\theta$

→ Probabilistically

$$P(Y \mid X, \theta) = \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[\frac{-1}{2\sigma^2}(y_n - \theta_n^T x_n)^2\right]$$

Both vars have norm. dist.

Now, we mix in a prior:

$$P(w) = \frac{\lambda}{\sqrt{2\pi}} \exp\left[\frac{-\lambda}{2}\theta^T\theta\right]$$

$$P(w \mid y, X) \propto P(y \mid X, w) * P(w)$$

↑ max a posteriori

So, how the fresh heck do we optimize this!?

↳ find cost, take deriv & set to 0

$$J = (Y - X\theta)^T (Y - X\theta) + \lambda \theta^T \theta$$

Note how we expect a similar form to before

$$\frac{\partial J}{\partial \theta} = -2X^T Y + 2X^T X\theta + 2\lambda\theta = 0$$

$$X^T X\theta + \lambda\theta = X^T Y$$

$$(\lambda I + X^T X)\theta = X^T Y$$

$$\therefore \boxed{\theta = (\lambda I + X^T X)^{-1} X^T Y}$$

I did it! ☺

Dummy variable trap

2 types of encoding → one hot: k-classes = k vars
→ k-1: k classes = k-1 vars

k-1 encoding puts one class into bias variable

Why? $X^T X$ becomes singular

↳ columns no longer linearly indep.

→ How to deal with the trap?

↳ k-1 encoding, drop bias term, use ~~L1~~ L2 norm, use gradient desc.

↖ used through deep learning.

Why doesn't L1 reg work?

↳ L1 can still be non-lin. indep.

→ Other problems: Multicollinearity

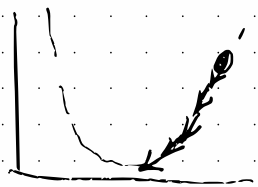
↳ 2+ vars are correlated to each other

Gradient descent

$\Theta \leftarrow$ random setting for weights

$$\Theta := \Theta - \eta \nabla_\Theta J(\Theta)$$

↑ gradient vector of $J(\Theta)$



↑ take steps down a gradient

$$\Theta := \Theta - \left(\eta * X^T(\hat{Y} - Y)\right)$$

↑ updated weights

What is so important about G.D?
↳ used when loss function is not convex.
ie. no global min that's easy to calculate

→ How to choose η?
↳ too large: never converge
↳ too low: slower computation

L1 Regularisation "Lasso"

↳ used for feature selection

$$J = \sum_{n=1}^{N} (y_n - \hat{y}_n)^2 + \lambda \|\theta\|_1$$

↳ probabilistically: $p(\theta) = \frac{1}{2}\exp(-\lambda|\theta|)$
↑ Laplace dist.

$$\frac{\partial J}{\partial \theta} = -2X^T y + 2X^T X\theta + \lambda \, \text{sign}(\theta) = 0$$

↑ can't solve for θ

where $\text{sign}(x) \begin{cases} =1 & \text{when } x > 0 \\ =0 & \text{when } x = 0 \\ =-1 & \text{when } x < 0 \end{cases}$

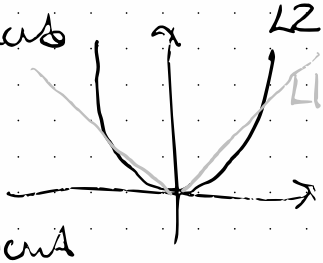
L1 vs L2 reg

→ Help to prevent overfitting

↳ L1 chooses most important features

↳ L2 doesn't allow for huge weights.

This is because L2 penalty is quad & L1 is an abs func



L2

$\theta_i \to 0$; if θ_i is small then it won't change much and become 0

L1

if $\theta_i = 0$ then it will stay there forever (sparsity)

combine L1 & L2 to create elastic net

$$J_{EN} = J + \lambda_1 |w| + \lambda_2 |w|^2$$