

Introduction to IOAI TSP

IOAI Training and Selection Programme 2025

Mar 29, 2025 Sat

Who's training you



- Tan Nian Wei (github.com/tnwei)
- 7 yrs Applied ML Scientist @ PETRONAS Research
- 1 yr Tech Cofounder @ Rosary Labs

My responsibilities

Responsible for student training and selection within the country. That means establishing and running this course

Responsible for the Malaysian delegation at IOAI and IAIO. That includes bringing the team overseas and back and fulfilling on-site duties such as voting in the General Assembly

Thank you very much for your time and

Tan Nian Wei

Head coach and Malaysia team lead
Malaysia AI Olympiad Committee

Our efforts here are supported by the same people who run the other Malaysian olympiads

Purpose of training

YES

- Kickstart your journey towards the AI equivalent of mathematical maturity <- Google this!
- Provide a fair chance for excellent talent (you!) to catch up on AI knowledge

NO

- Directly teach you everything you need to know to be good in AI (not even olympiad!) Maybe only 20%. The rest is up to you!

Neural networks crash course

IOAI Training and Selection Programme 2025

Mar 29, 2025 Sat

Table of contents

- Chapter 1 - From high school math to your first neural network
- Chapter 2 - Extending NNs to classification (next session)

Chapter 1

From high school math to your first neural network

Starting point: Equation of a straight line

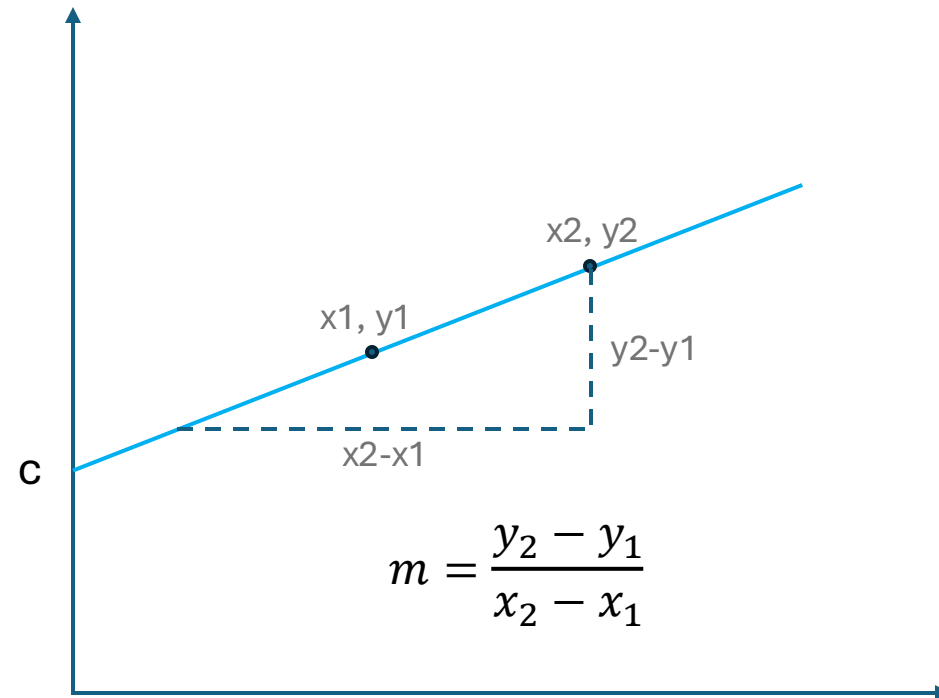
<https://spmaddmaths.blog.onlinetuition.com.my/2020/08/6-4-equation-of-straight-lines-part-2.html>

Equation in the gradient form

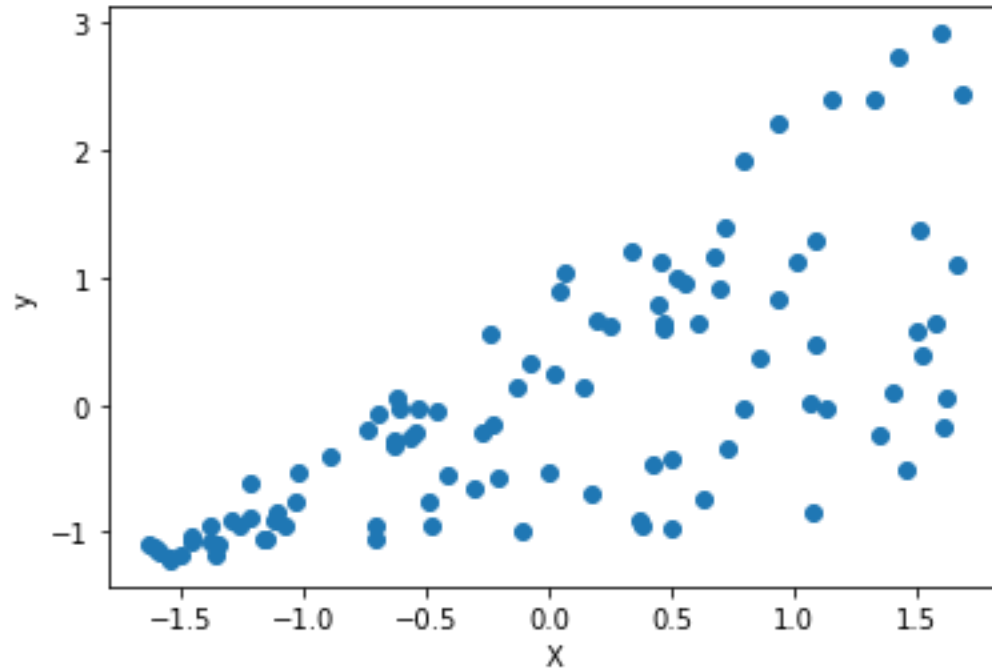
$$y = mx + c$$

m = gradient

c = y -intercept



Fit the best straight line based on data



Let us start with this model:

$$\hat{y} = ax$$

Pronounced as “y hat”

where:

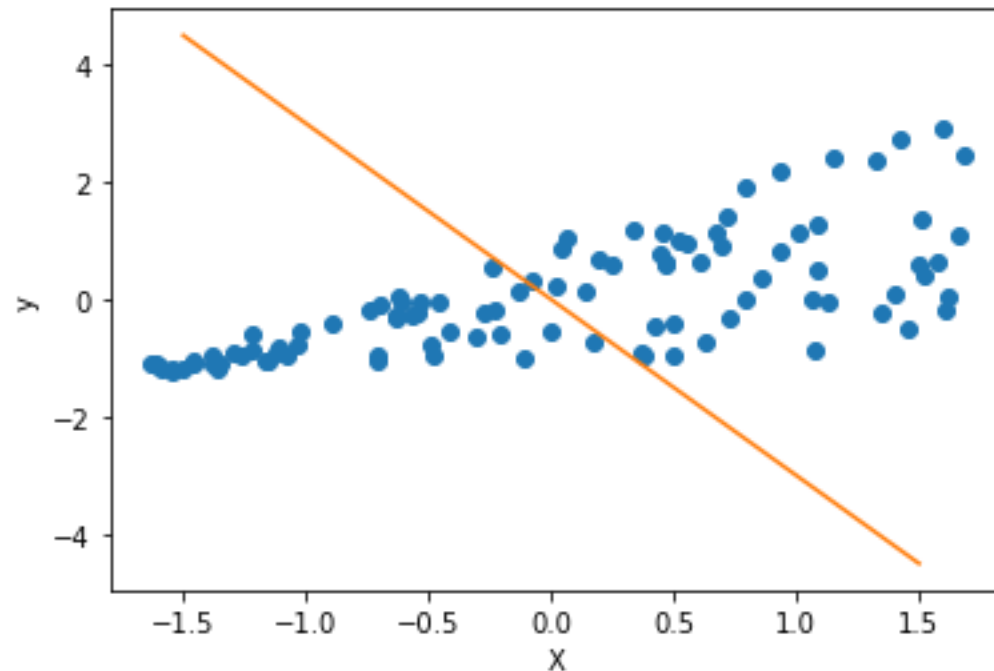
\hat{y} is our prediction

a is the gradient of the best fit line

The only variable that we can change to get a better model is a . We call a a **parameter**.

Fit the best straight line based on data (cont)

We make a first guess that $a = -3$

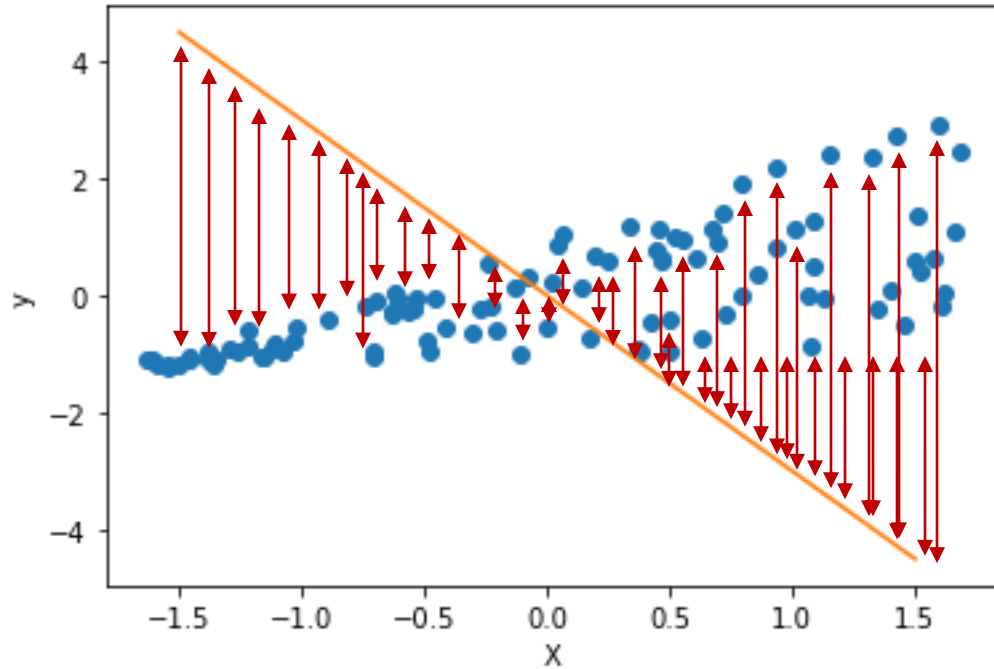


We can easily estimate a here with our eyes, but let's say we can't

How do we know whether to increase or decrease a ?

We need a **loss function** to tell our model what direction to adjust our variables.

Mean squared error (MSE) loss function



Mean Squared Error

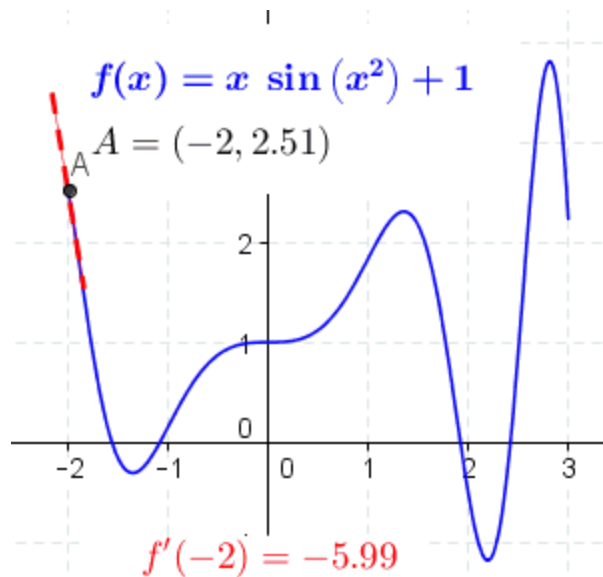
$$L = \sum (y - \hat{y})^2$$

Calculate difference between true labels and prediction, then take sum of their squares

Smaller is better. Hence, we want to **minimize loss** by adjusting a

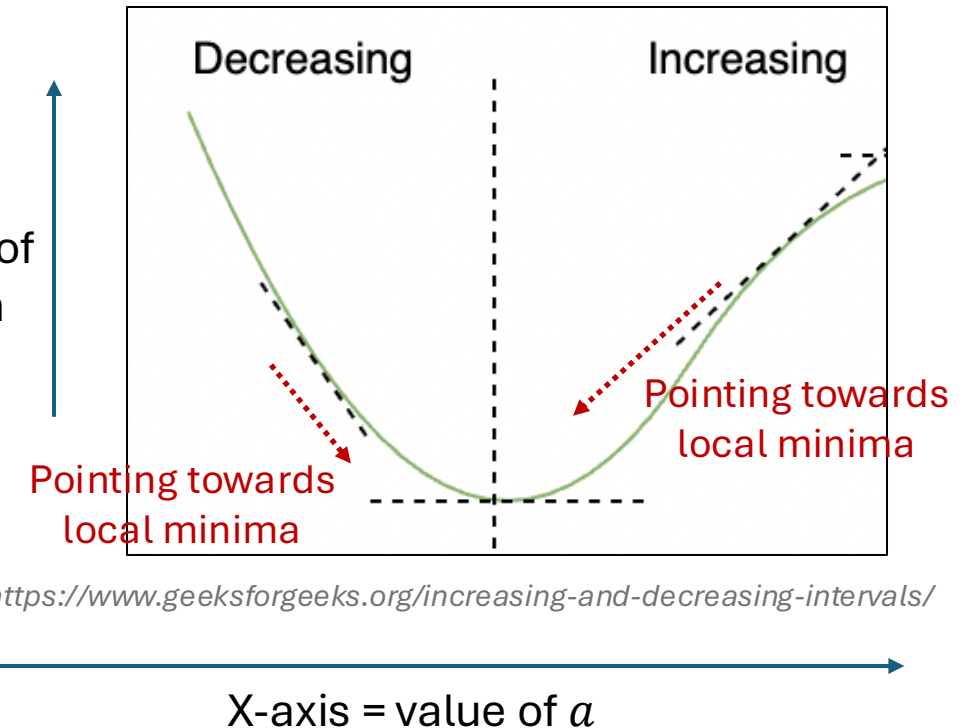
Gradient descent: find local minima w/ slope

If we can calculate a loss function,
we can calculate its **derivative**
i.e. rate of change



https://en.wikipedia.org/wiki/File:Tangent_function_animation.gif

Y-axis = value of
loss function

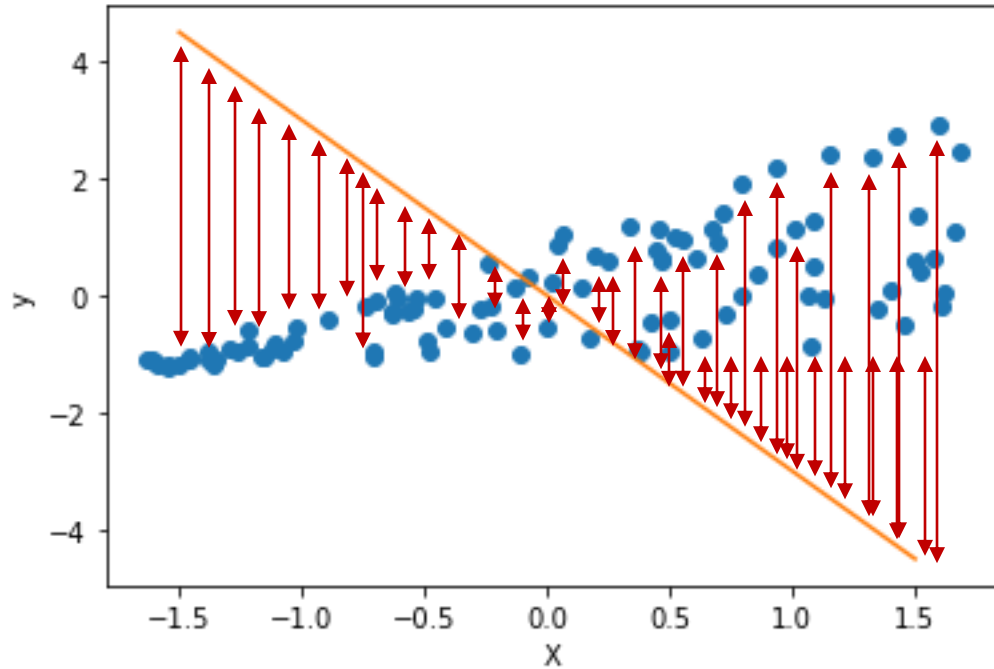


<https://www.geeksforgeeks.org/increasing-and-decreasing-intervals/>

Iteratively adjust the value of a until we arrive at minimal loss

We call this **gradient descent**

Mean squared error (MSE) loss function



Mean Squared Error

$$L = \sum (y - \hat{y})^2$$

Calculate difference between true labels and prediction, then take sum of their squares

Smaller is better. Hence, we want to **minimize loss** by adjusting a

If we can find the derivative, we know what direction to increase or decrease our parameter a to decrease the loss function

Derivative of MSE loss

Power Rule Formula



$$\frac{d}{dx}(x^n) = n \cdot x^{n-1}$$

<https://www.cuemath.com/calculus/power-rule/>

Find expression of loss in terms of a

$$\begin{aligned}\hat{y} &= ax \\ \text{MSE loss, } L &= (y - \hat{y})^2 \\ &= y^2 - 2y\hat{y} + \hat{y}^2 \\ &= y^2 - 2axy + a^2x^2\end{aligned}$$

Find derivative of loss with respect to a

$$\begin{aligned}\frac{d}{da} L &= \frac{d}{da} (y^2 - 2axy + a^2x^2) \\ &\quad \begin{array}{ccc} \swarrow & \downarrow & \searrow \\ \text{No powers of } a, \text{ discard completely} & a^1 \text{ becomes } 1 & a^2 \text{ becomes } 2a^1 \end{array} \\ \frac{dL}{da} &= -2xy + 2ax^2\end{aligned}$$

To find the best value of

Very rudimentary optimization loop

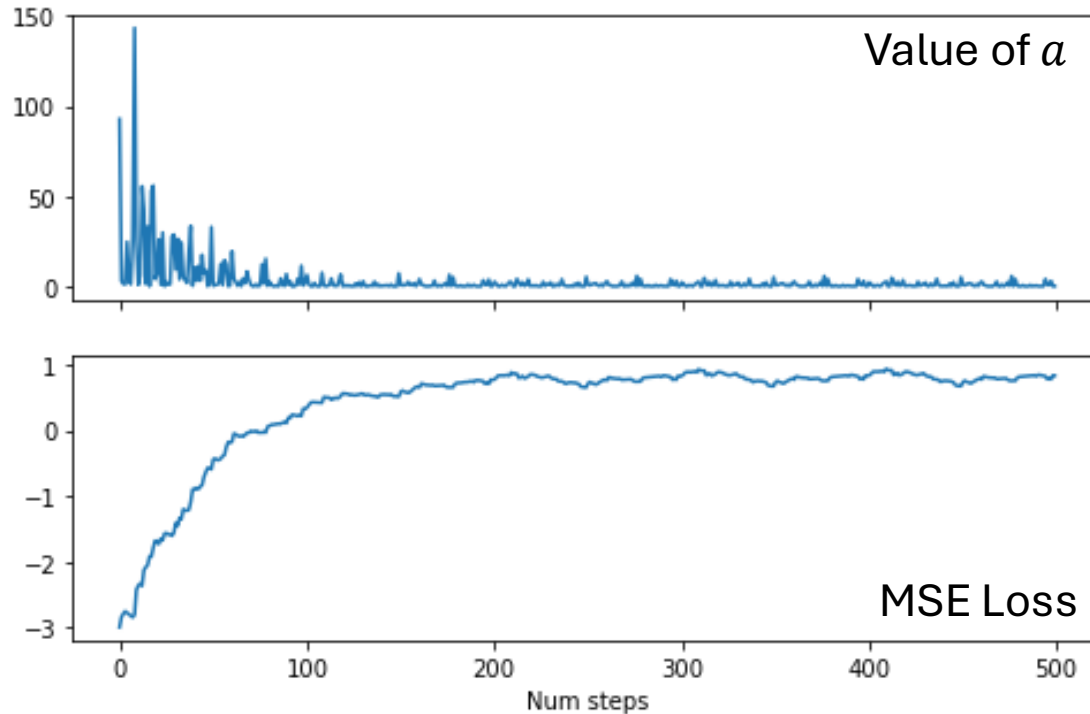
```
def f(a, x):  
    return a * x  
  
def mse(y, y_hat):  
    return (y - y_hat) ** 2  
  
def dl_da(a, x, y):  
    return -2*x*y + 2*a*x ** 2
```

```
history = []  
a = -3 # Initial estimate  
step_size = 1e-2 # Arbitrary step size  
  
# Arbitrary num of loops over dataset i.e. epochs  
for _ in range(5):  
    # Iterate over every single data point  
    for x, y in zip(X_sel, y_sel):  
        # Get y_hat  
        y_hat = f(a, x)  
  
        # Calculate loss  
        current_loss = mse(y, y_hat)  
  
        # Calculate loss derivative  
        current_loss_d = dl_da(a, x, y)  
  
        # Modify a using loss derivative  
        a = a - step_size * current_loss_d  
  
        # Store values for plotting  
        history.append((current_loss, a))
```

Optimization is a complex science. Google or ask ChatGPT about “operations research”

Optimization results

Loss plateaued to a low value



a stabilizes somewhat at a plateau value

```
df.tail()
```

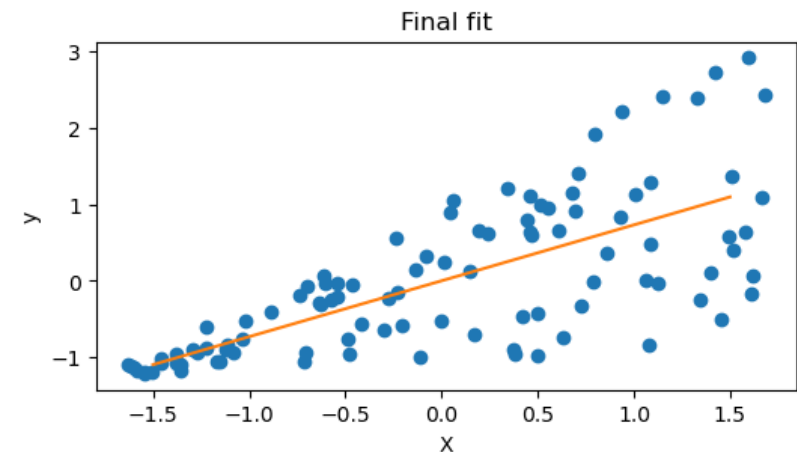
	loss	a
495	0.235708	0.702088
496	0.106807	0.697257
497	1.611657	0.739867
498	0.138629	0.735857
499	0.074217	0.731168

Final a approaches true answer below

```
test_lr = LinearRegression().fit(X_sel.reshape(-1, 1), y)
```

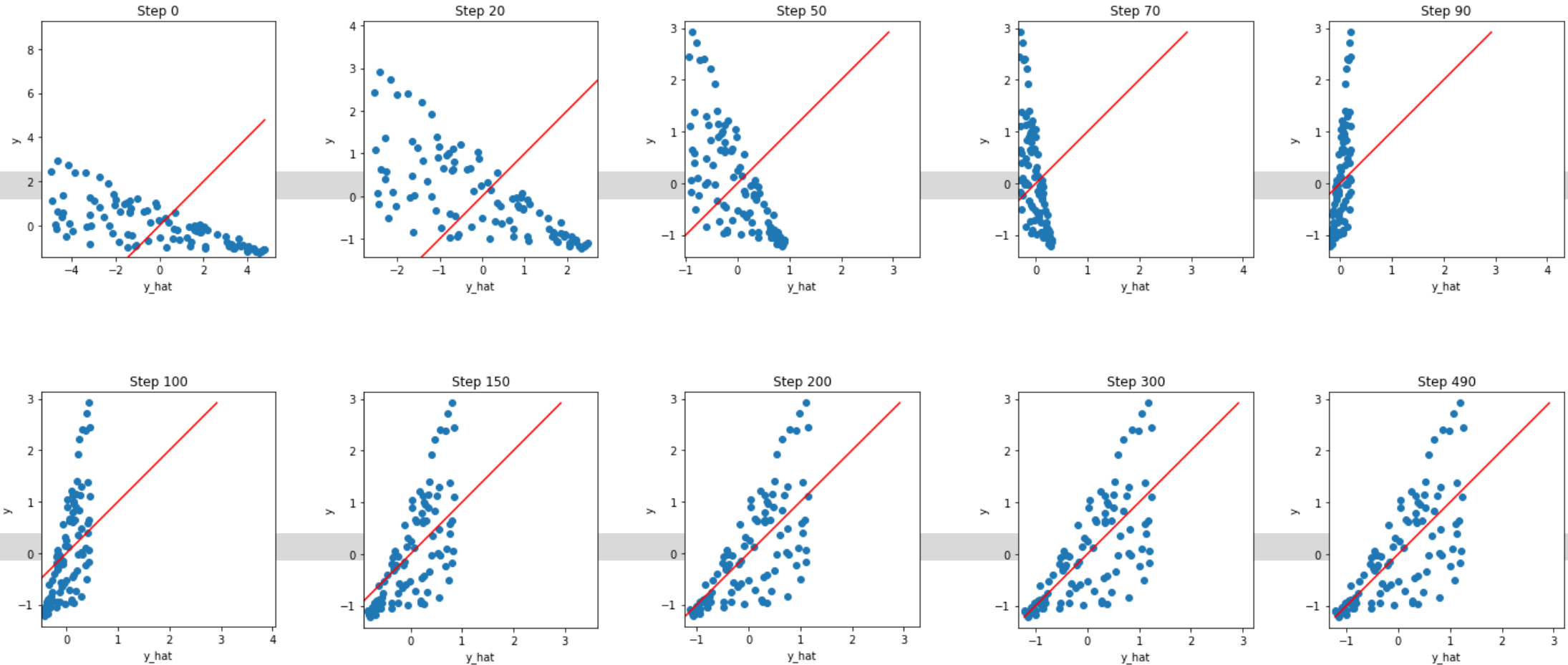
```
test_lr.coef_, test_lr.intercept_
```

```
(array([[0.7088893]]), array([3.96051352e-16]))
```



Change in predictions over steps

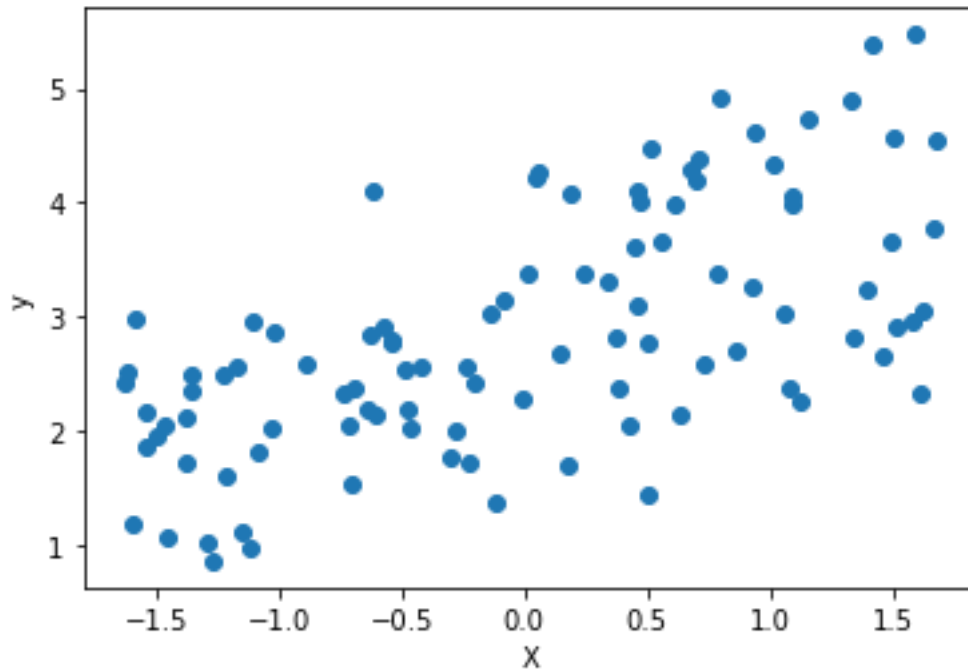
If predictions are perfect, all clusters will be exactly on red line



Slightly more complex data and model

Y is not 0 when x is 0!

This data will require an intercept to
fit a reasonable best fit line



Let's now use

$$\hat{y} = ax + b$$

where:

\hat{y} is our prediction

a is the gradient of the best fit line

b represents the intercept

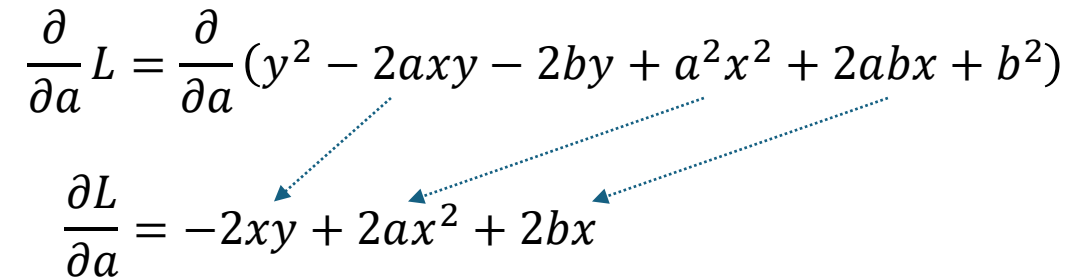
Both a and b are parameters we can
adjust.

Partial derivatives of loss

Find expression of loss in terms of a and b

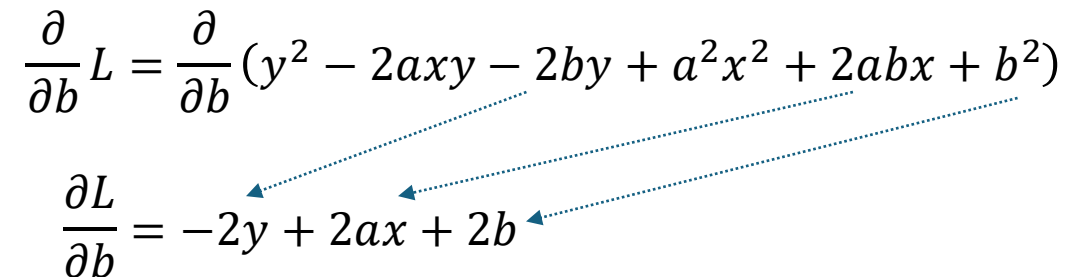
$$\begin{aligned}\hat{y} &= ax + b \\ \text{MSE loss, } L &= (y - \hat{y})^2 \\ &= y^2 - 2y\hat{y} + \hat{y}^2 \\ &= y^2 - 2axy - 2by \\ &\quad + a^2x^2 + 2ax + b^2\end{aligned}$$

Find partial derivative of loss with respect to a

$$\begin{aligned}\frac{\partial}{\partial a} L &= \frac{\partial}{\partial a} (y^2 - 2axy - 2by + a^2x^2 + 2abx + b^2) \\ \frac{\partial L}{\partial a} &= -2xy + 2ax^2 + 2bx\end{aligned}$$


Implication of partial derivative here: no big deal!
Use different symbol and carry on "\('')_/'"

Find partial derivative of loss with respect to b

$$\begin{aligned}\frac{\partial}{\partial b} L &= \frac{\partial}{\partial b} (y^2 - 2axy - 2by + a^2x^2 + 2abx + b^2) \\ \frac{\partial L}{\partial b} &= -2y + 2ax + 2b\end{aligned}$$


Updated optimization loop

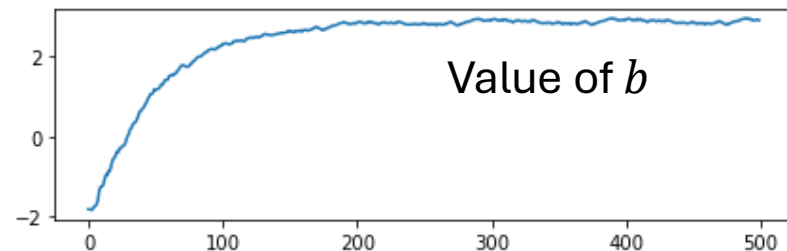
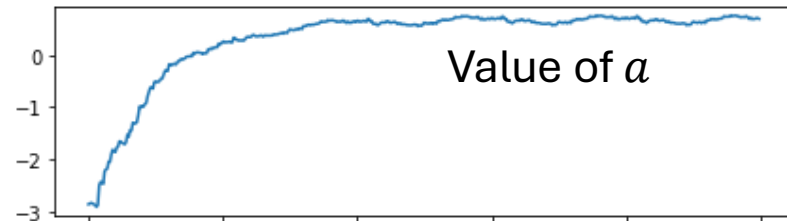
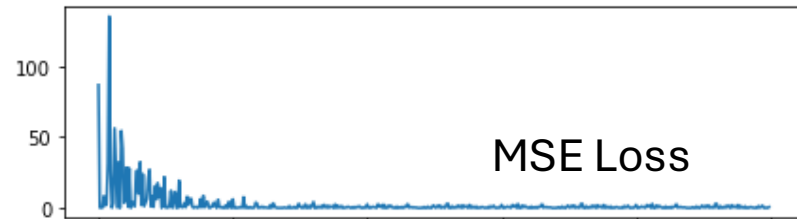
```
def f(a, b, x):  
    return a * x + b  
  
def mse(y, y_hat):  
    return (y - y_hat) ** 2  
  
def dl_da(a, b, x, y):  
    return -2*x*y + 2*a*x ** 2 + 2 * b * x  
  
def dl_db(a, b, x, y):  
    return -2*y + 2*a*x + 2*b
```

```
history = []  
  
# Initialize parameters with estimates  
a = -3  
b = -2  
  
# Arbitrary step sizes  
step_size_a = 1e-2  
step_size_b = 1e-2
```

```
# Arbitrary num of epochs  
for _ in range(5):  
    # Iterate over every single data point  
    for i, j in zip(X_sel, y_sel2):  
  
        # Get y_hat and calculate loss  
        y_hat = f(a, b, x)  
        current_loss = mse(y, y_hat)  
  
        # Calculate loss derivatives  
        current_loss_da = dl_da(a, b, x, y)  
        current_loss_db = dl_db(a, b, x, y)  
  
        # Take one optimization step for parameters  
        a = a - step_size_a * current_loss_da  
        b = b - step_size_b * current_loss_db  
  
        # Store values for plotting  
        history.append((current_loss, a, b))
```

Updated optimization results

Loss plateaued to a low value



a and b stabilize somewhat at a plateau value

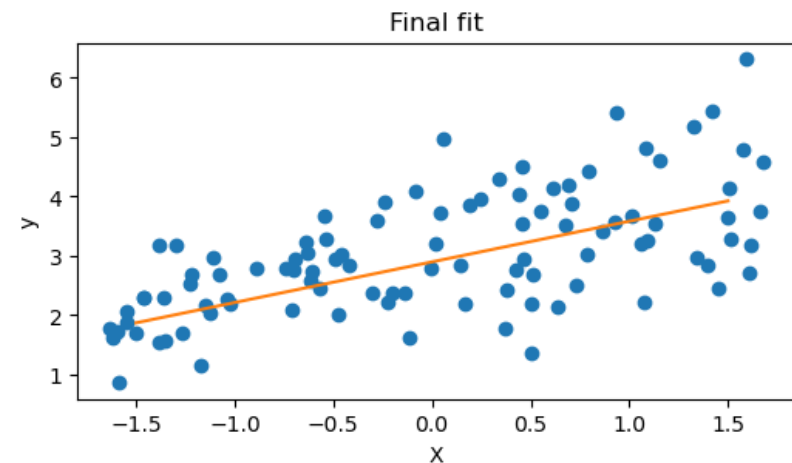
```
df.tail()
```

	loss	a	b
495	0.169888	0.683597	2.901479
496	0.004583	0.684598	2.900125
497	0.253693	0.701503	2.910198
498	0.055372	0.698969	2.914905
499	0.676939	0.684807	2.898449

Final a and b approach true answer below

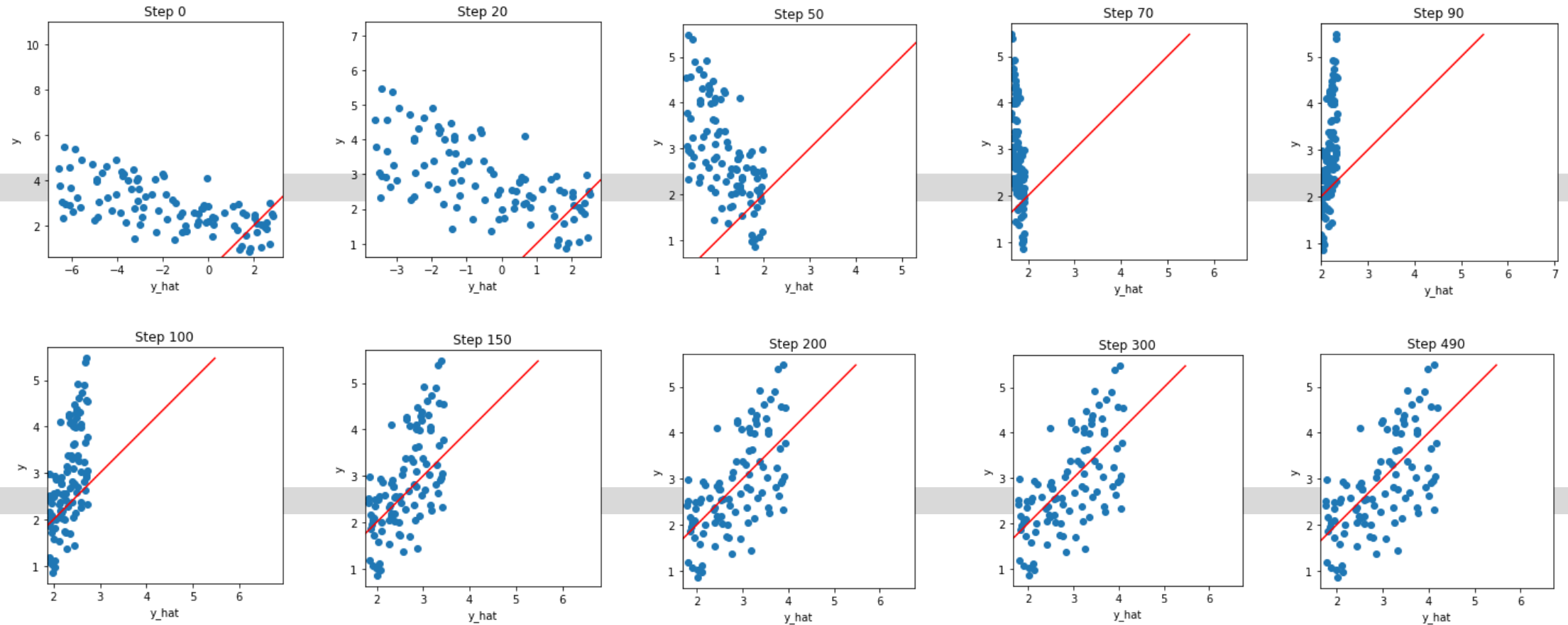
```
test_lr.coef_, test_lr.intercept_
```

```
(array([0.80320787]), 3.030554246423506)
```



Updated change in predictions over steps

If predictions are perfect, all clusters will be exactly on red line



Generalizing our toy model to make it useful

$\hat{y} = ax + b$ is only good for using 1 **feature** to predict 1 **label**
How do we make it useful for real world applications?

2 features

$$\hat{y} = a_1x_1 + a_2x_2 + b$$

3 features

$$\hat{y} = a_1x_1 + a_2x_2 + a_3x_3 + b$$

3 features, 2 labels

$$\hat{y}_1 = a_1x_{11} + a_2x_{12} + a_3x_{13} + b_1$$

$$\hat{y}_2 = a_1x_{21} + a_2x_{22} + a_3x_{23} + b_2$$

We can use matrix math!

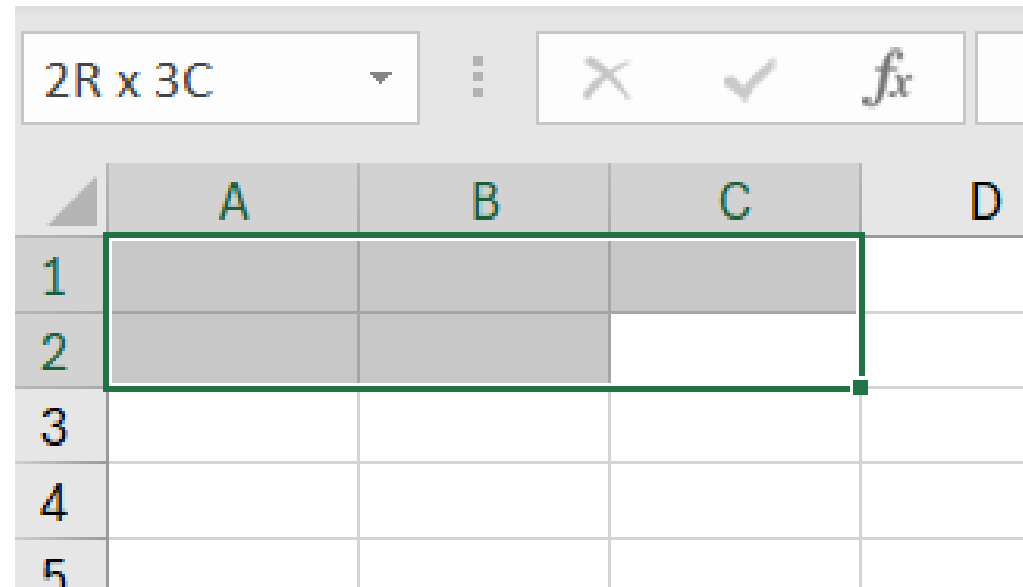
Matrix math nomenclature

Columns

- A, B, C, D are columns. Columns increase to the right!
- Columns are the 2nd dimension -> 3 cols selected in the image below

Rows

- 1, 2, 3, 4, 5 are rows. Rows increase downwards!
- Rows are the 1st dimension -> 2 rows selected in the image on the right



	A	B	C	D
1				
2				
3				
4				
5				

Matrix multiplication rules

Outer dimension
can be anything!

$$\begin{matrix} \mathbf{M} & \times & \mathbf{N} \\ (2, 3) & & (3, 4) \end{matrix} = \begin{matrix} m_{11} & m_{12} & m_{13} \\ \boxed{m_{21}} & \boxed{m_{22}} & \boxed{m_{23}} \end{matrix} \times \begin{matrix} n_{11} & n_{12} & \boxed{n_{13}} & n_{14} \\ n_{21} & n_{22} & \boxed{n_{23}} & n_{24} \\ n_{31} & n_{32} & \boxed{n_{33}} & n_{34} \end{matrix}$$

Inner dimension
must be the same!

$$\begin{matrix} \mathbf{O} \\ (2, 4) \end{matrix} = \begin{matrix} o_{11} & o_{12} & o_{13} & o_{14} \\ o_{21} & o_{22} & \boxed{o_{23}} & o_{24} \end{matrix}$$

where o_{ij} is row of M times col of N

e.g.
$$o_{23} = m_{21}n_{13} + m_{22}n_{23} + m_{23}n_{33}$$

Deriving a general expression

2 features

$$\hat{y} = a_1x_1 + a_2x_2 + b$$

$$\hat{y} = [x_1 \quad x_2] \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} + b$$

3 features

$$\hat{y} = a_1x_1 + a_2x_2 + a_3x_3 + b$$

$$\hat{y} = [x_1 \quad x_2 \quad x_3] \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} + b$$

3 features, 2 labels

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} =$$

(2, 1)

Need a (2,1) here!

Only possible if

(2,3) x (3,1)

$$+ \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

(2, 1)

3 features but calculate 2 data points in one go

$$\hat{y}_1 = a_1x_{11} + a_2x_{12} + a_3x_{13} + b$$

$$\hat{y}_2 = a_1x_{21} + a_2x_{22} + a_3x_{23} + b$$

Original eqns as reference

Hence

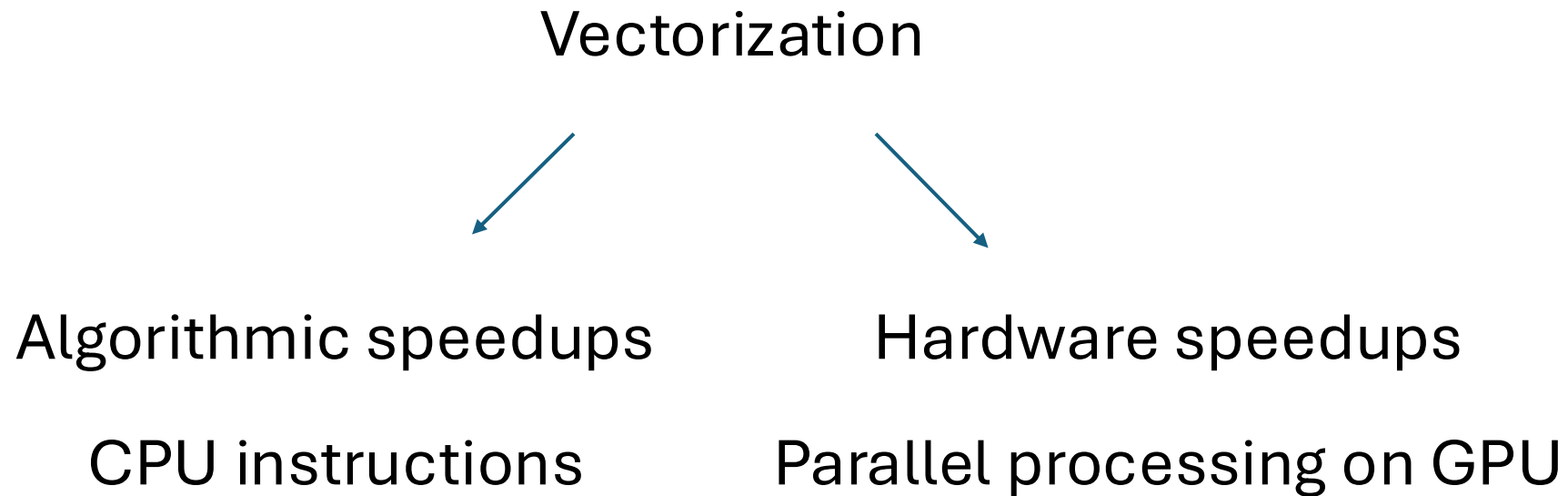
$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

General form

$$\hat{\mathbf{Y}} = \mathbf{X} \mathbf{W} + \mathbf{b}$$

for any dim of inputs and outputs

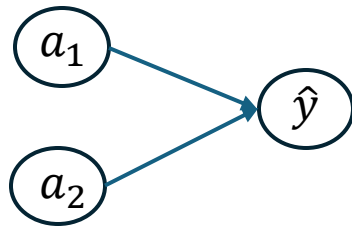
Benefits of expressing as matrix calculations



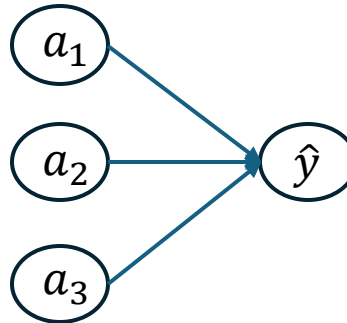
Building block for a single fully-connected neural network layer

Note: Bias is usually excluded from these neural network diagrams. But it's there.

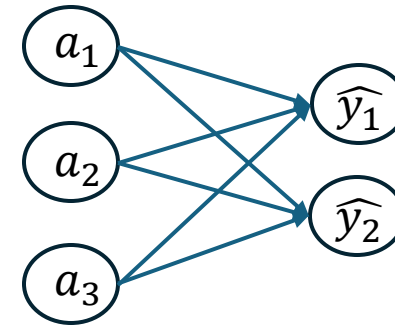
2 features



3 features



3 features, 2 labels



Now you know how a neural network works from the ground up!

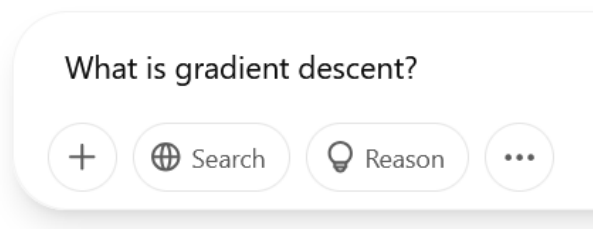
Proper prompting techniques using LLMs to help learning

IOAI Training and Selection Programme 2025

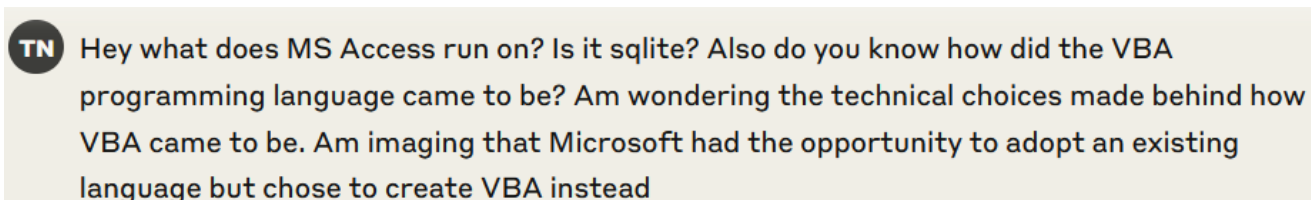
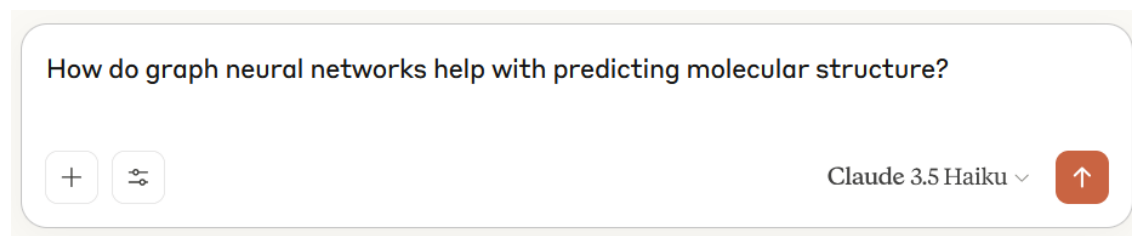
Mar 29, 2025 Sat

LLM assistants are useful at various levels of detail

Ask simple questions when you know don't know much about a subject



Ask long questions when you want to know more in-depth. What LLM assistants know might surprise you!



The more detail you provide, the better they can answer you

TN Right now I have this code:

(skipped lots of code)

Originally I just had the ThreadedConnectionPool be implemented as a single `POOL` variable in this module. Was that a singleton? In any way, in the refactored code above, I realize I can potentially create multiple copies of the connection pool.

So for more context, I used to have this conn pool in my `core/` logic module when I am supposed to put it in my business logic module in `api/`. Im thinking right now that as long as I am careful in only defining a single instance of this database in my `api/` logic, I should be fine? Let me know what you think

TN I am writing a retrieval augmented generation app where I have split the code into the core logic and the application code. I am currently refactoring the app as previously everything was in one script (crazy I know!). I am putting the code into `core/` and `api/` respectively.

Right now my `api/server.py` import block looks like this:

(skipped lots of code)

See that one bit there that imports from `core.sql` ? I have that module attached below.

I feel very iffy that the AzurePDFDownloader bit is in the core, and so is the database connection. FYI the other files in the core are mostly just logic, like so:

(skipped lots of code)

Can you advise me a bit here? I do think some stuff in `core/sql.py` should be put into the services, but I don't know how to preserve the remaining logic in that legitimately are tied to executing SQL code on a server

The more context you type in, the more it helps

ELI5 trick if you are completely lost

“Explain like I’m 5” will prompt LLMs to use simple terms and analogies

ELI5 how does Snell's law of refraction work?

Imagine this:

- You're running on a beach, and there's water ahead.
- Running on the sand is easy, but moving through the water slows you down.
- If you run straight into the water, you'll keep going straight (just slower).
- But if you enter at an angle, your inside foot hits the water first and slows down, making you turn slightly.

Snell's Law in action:

- Light does the same thing! When it moves from one material (like air) to another (like water), it changes speed.
- This change in speed causes the light to bend at the boundary.
- The amount of bending depends on the **refractive index** of the materials.

Ask specific questions to scope the answer

Yes:

Okay can you explain Snell's law to me again? The part that I really don't get is how light changes speed. I thought speed of light is $3 \cdot 10^8$ m/s?



Search



Reason



No:

Re-ask the exact same question expecting a better answer (might work sometimes but don't bank on it)

Explain Snell's law

Explain Snell's law

Explain Snell's law

Explain Snell's law

Vague questions

What Snell's law

Why Snell's law



Search



Search

Gotchas

- LLMs are trained to be agreeable. They tend to say yes to you on complex topics when it is not true. Challenge them when you think something is off. You'd be surprised how easily it changes its mind.
- LLMs are good at writing a bit of code but are not very good at writing a lot of code. If you only write code using LLMs, your code will get complicated very fast and reach a point where you don't know how you got there. Use LLM assistance in writing code sparingly for your own sake. Better if you learn coding without touching LLMs at all
- If LLM gave you a wrong answer, it's your responsibility! You didn't fact check it and you trusted what it gave you

ChatGPT and geng can be your
greatest friend in learning.
Use it well!

Writing good code

IOAI Training and Selection Programme 2025

Mar 29, 2025 Sat

Don't put too much in one cell!

Each notebook cell should have one purpose only

- Your code will be harder to read
- You will be accidentally running more code when you make changes
- Try and keep imports and function definitions in their own cell

```
# convert dataset to a loader that could be feed during tra
def tokenize_captions(captions, is_train=True):
    inputs = tokenizer(
        captions, max_length=tokenizer.model_max_length, pa
    )
    return inputs.input_ids

# set manual seed to reproduce the exact model
import random
torch.random.manual_seed(3)
random.seed(3)

train_transforms = transforms.Compose(
    [
        transforms.RandomResizedCrop(resolution),
        transforms.RandomAffine(5),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.5], [0.5]),
    ]
)

def preprocess_train(examples):
    images = [image.convert("RGB") for image in examples['i
    examples["pixel_values"] = [train_transforms(image) for
    examples["input_ids"] = tokenize_captions(examples['tex
    return examples

train_dataset = dataset['train'].with_transform(preprocess_

def collate_fn(examples):
    pixel_values = torch.stack([example["pixel_values"] for
    pixel_values = pixel_values.to(memory_format=torch.cont
    input_ids = torch.stack([example["input_ids"] for examp
    return {"pixel_values": pixel_values, "input_ids": input

train_dataloader = torch.utils.data.DataLoader(
    train_dataset,
    shuffle=True,
    collate_fn=collate_fn,
    batch_size=train_batch_size,
    num_workers=2,
)
```

Be Hansel and Gretel and leave breadcrumbs. Write down your thought process throughout your notebook

- It is good for you because writing is conscious thinking
- It is good for the reader because unless you write very clear code, it takes a lot of effort for the reader to understand what you are trying to do. Most of the time, this reader is yourself in the future.

We first need to combine data across all three folders before we begin.

```
[67]: chihuahua_test_tens = []
      muffins_test_tens = []

      for i, j in zip(test_tens, test_labels):
          if j == 0:
              chihuahua_test_tens.append(i)
          elif j == 1:
              muffins_test_tens.append(i)

      chihuahua_test_tens = torch.stack(chihuahua_test_tens, dim=0)
      muffins_test_tens = torch.stack(muffins_test_tens, dim=0)
```

```
[68]: all_chihuahua_tens = torch.concat([chihuahua_tens, chihuahua_test_tens], dim=0)
      all_muffins_tens = torch.concat([muffins_tens, muffins_test_tens], dim=0)
```

```
[69]: # This cell can take some time
      # If you run out of memory, model.forward() each image one at a time
      all_chihuahua_z = model.forward(all_chihuahua_tens)
      all_muffins_z = model.forward(all_muffins_tens)
```

```
[70]: all_z = torch.concat([all_chihuahua_z, all_muffins_z], dim=0)
```

Fitting a PCA to project the embeddings to a lower and more interpretable set of dimensions.

```
[71]: # The output of PCA are numpy arrays!
      # Import numpy to work w/ them
      import numpy as np
```

```
[72]: fitted_pca = PCA(n_components=2).fit(all_z.detach().numpy())
```

```
[73]: dim2_chihuahua = fitted_pca.transform(all_chihuahua_z.detach().numpy())
      dim2_muffins = fitted_pca.transform(all_muffins_z.detach().numpy())
```

```
[74]: dim2_chihuahua.shape, dim2_muffins.shape
```

```
[74]: ((18, 2), (18, 2))
```

```
[75]: dim2_chihuahua_proto = np.mean(dim2_chihuahua, axis=0)
      dim2_muffins_proto = np.mean(dim2_muffins, axis=0)
```

Don't be afraid to write long prose to summarize your efforts!

```
print("Logreg precision / recall",  
      precision_score(y, y_pred_logreg, zero_division=0, pos_label=1),  
      recall_score(y, y_pred_logreg, zero_division=0, pos_label=1),  
      f1_score(y, y_pred_logreg, zero_division=0, pos_label=1),  
      sep=" ")
```

Logreg precision / recall 1.0 1.0 1.0

My first thought from plotting the dataset is that it has to do with coordinates. After visualizing, my first thought was something about graphing based on distance and grouping them up. I tried K-means clustering but if num_cluster is low, it will group up some of the body part as well but if num_cluster is high, it will group correctly but doesn't help in the logistic regression part.

My final solution was a bit more informatics than AI though. If the distance between ...

Good Python

- Try to use descriptive variable names when you can unless it is convention. Instead of ``a``, say ``min_val``
- Use consistent spacing and formatting. (Use black / ruff formatter if you can)