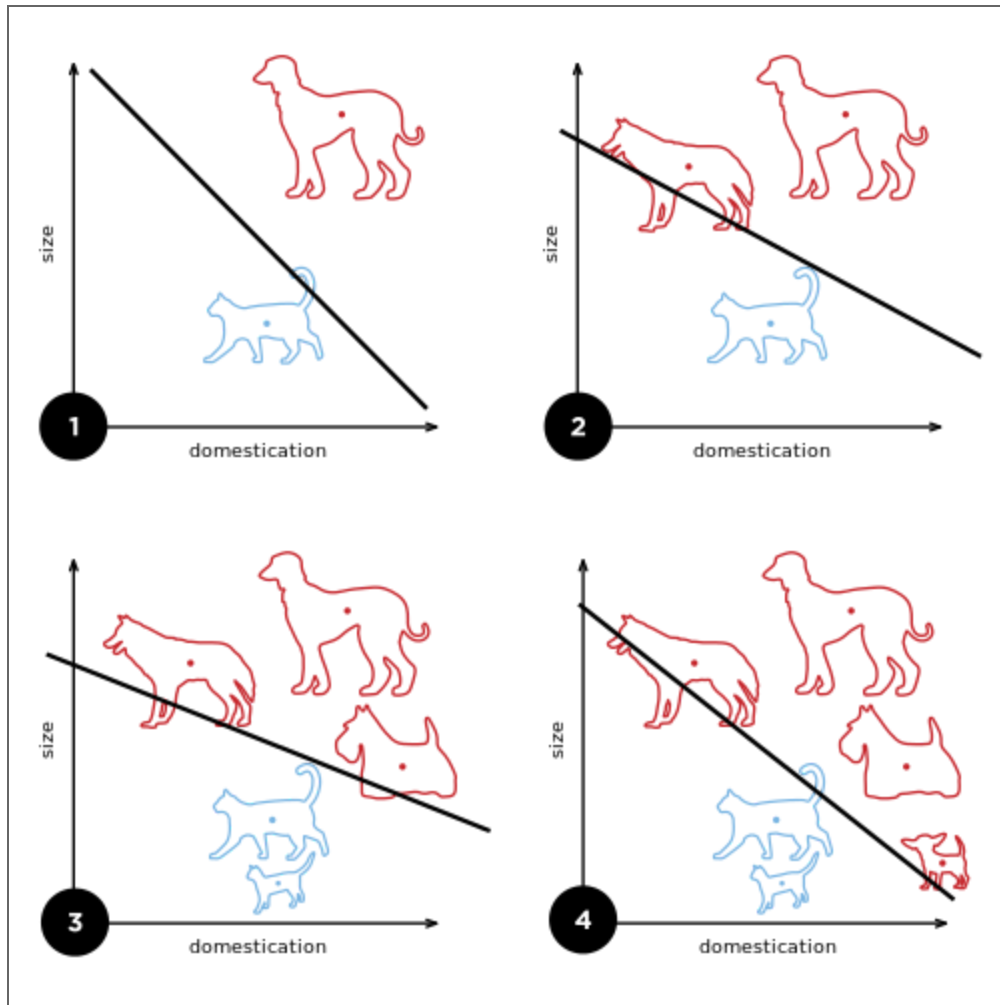


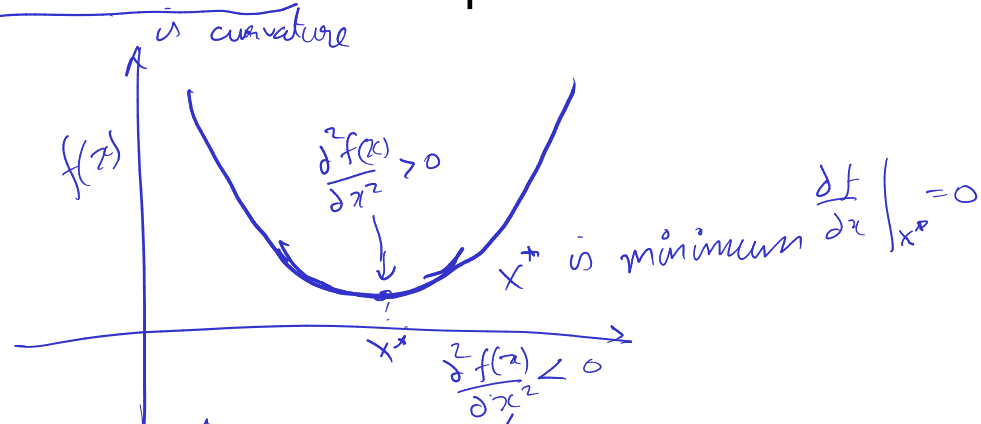
# Perceptron



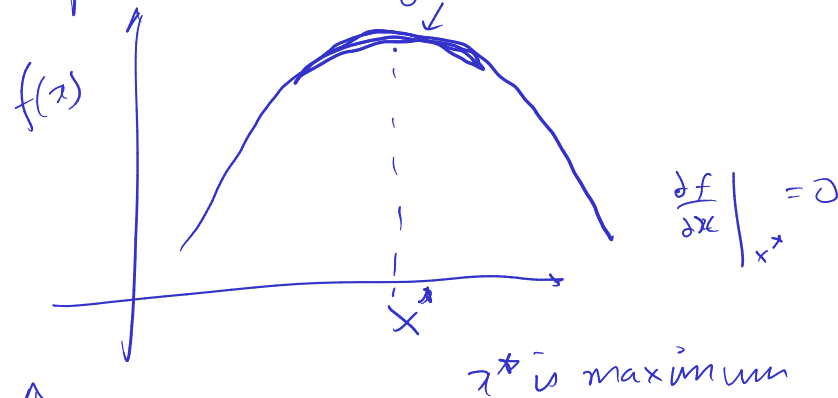
## Second derivative

What is the role of second derivative in optimization?

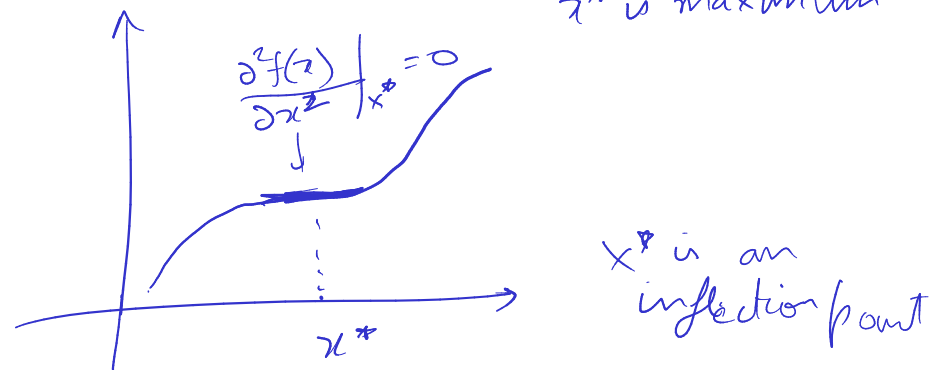
$$\left. \frac{\partial^2 f(x)}{\partial x^2} \right|_{x^*} > 0$$



$$\left. \frac{\partial^2 f(x)}{\partial x^2} \right|_{x^*} < 0$$



$$\left. \frac{\partial^2 f(x)}{\partial x^2} \right|_{x^*} = 0$$



Second derivative

How do the following functions differ?

$$f(x, y) = 2x^2 + 4y^2 - xy - 6x - 8y + 6$$

$$f(x, y) = -2x^2 - 4y^2 - xy - 6x - 8y + 6$$

$$f(x, y) = 2x^2 - 4y^2 - xy - 6x - 8y + 6$$

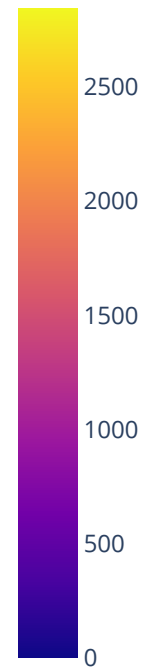
$$f(x, y) = 2x^2 + 4y^2 - xy - 6x - 8y + 6$$

```
In [3]: def f(x, y):
        return 2*x**2 + 4*y**2 - x*y - 6*x - 8*y + 6
plot_surface(f)
```

$\frac{\partial^2 f}{\partial y^2} = 8 > 0$   
 $\frac{\partial^2 f(x, y)}{\partial x^2} = 4 > 0$

Please look at the notebook for visualization.

The surface looks is a bowl because second derivatives are positive.



$$\frac{\partial^2 f}{\partial x^2} = -4 < 0$$

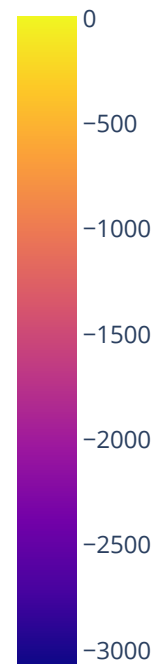
$$f(x, y) = -2x^2 - 4y^2 - xy - 6x - 8y + 6$$

$$\frac{\partial^2 f}{\partial y^2} = -4$$

```
In [4]: def f(x, y): return - 2*x**2 - 4*y**2 - x*y - 6*x - 8*y + 6
        plot_surface(f)
```

Please look at the notebook for visualization.

The surface looks is an upside down bowl because second derivatives are negative.





$$f(x,y) = 2x^2 - 4y^2 - xy - 6x - 8y + 6$$

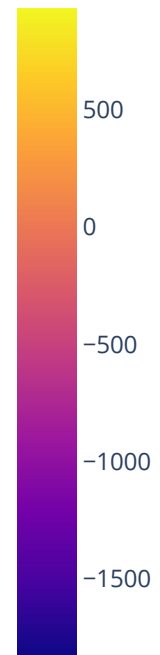
Handwritten notes above the equation:

- $\frac{\partial^2 f}{\partial x^2} = 4 > 0$  (with an arrow pointing to the  $2x^2$  term)
- $\frac{\partial^2 f}{\partial y^2} = -8 < 0$  (with an arrow pointing to the  $-4y^2$  term)

```
In [5]: def f(x, y): return 2*x**2 - 4*y**2 - x*y - 6*x - 8*y + 6
        plot_surface(f)
```

Please look at the notebook for visualization.

The surface looks like a saddle because one of the double derivatives is positive while other is negative.



$$a_{12} + a_{21} = -1$$

$$a_{12} = -k_2$$

$$a_{21} = -k_2$$

Hessians

$$f_1(x) = 2x^2 + 4y^2 - xy - 6x - 8y + 6$$

$$\frac{\partial^2 f(x)}{\partial x^2} = 4$$

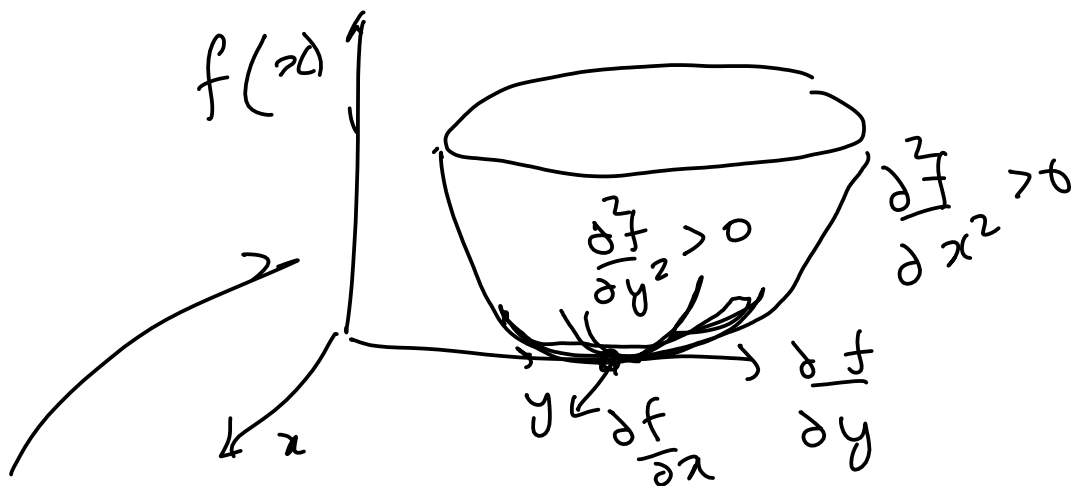
$$\frac{\partial^2 f(x)}{\partial y^2} = 8$$

$$\frac{\partial^2 f(x)}{\partial x \partial y} = -1$$

$$\frac{\partial^2 f(x)}{\partial y \partial x} = -1$$

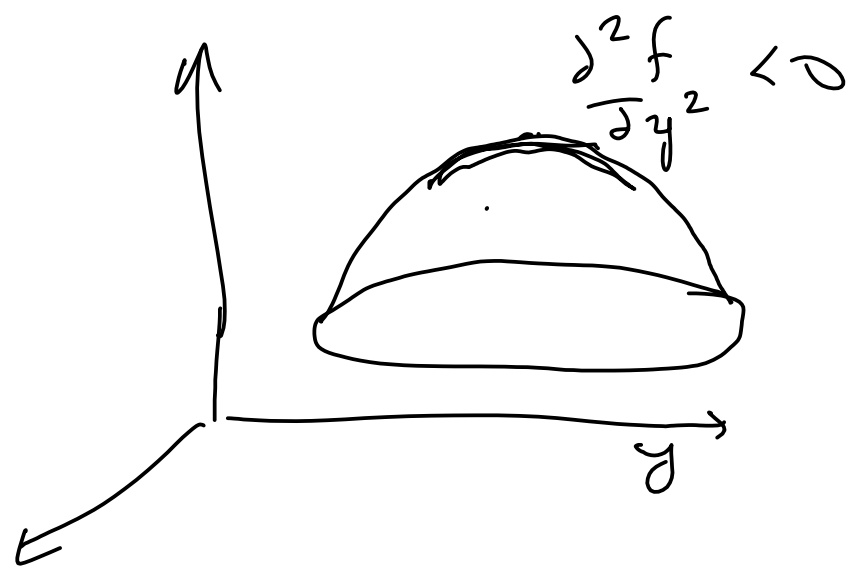
$$\frac{\partial}{\partial y} \left( \frac{\partial f}{\partial x} \right) = 4x - y - 6$$

$$\frac{\partial}{\partial x} \left( \frac{\partial f}{\partial y} \right) = 8y - x - 8$$

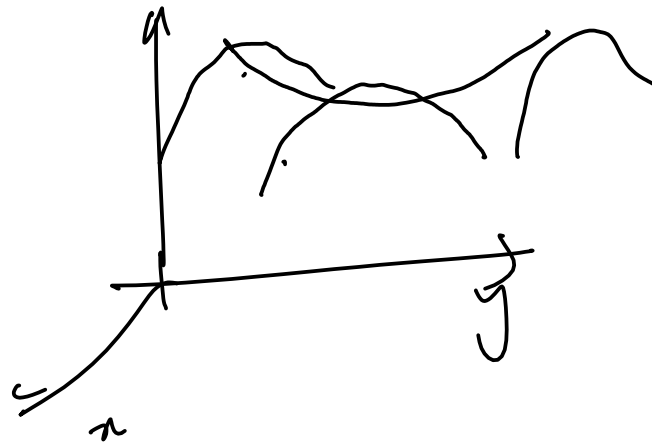


$$\frac{\partial^2 f}{\partial x^2} > 0, \frac{\partial^2 f}{\partial y^2} > 0 \Rightarrow \left( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) \Big|_{x^*} \text{ is minimum}$$

$$\frac{\partial^2 f}{\partial x^2} < 0, \quad \frac{\partial^2 f}{\partial y^2} < 0 \Rightarrow$$



$$\frac{\partial^2 f}{\partial x^2} < 0, \quad \frac{\partial^2 f}{\partial y^2} > 0 \Rightarrow$$



$$H_{xx} f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} \frac{\partial}{\partial x} \frac{\partial f}{\partial x} = \frac{\partial}{\partial x} \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix}$$

$$f(\underline{x}) = \underbrace{\underline{x}^T A \underline{x}} + \underline{b}^T \underline{x} + c$$

$$\frac{\partial}{\partial \underline{x}} f(\underline{x}) = 2 \underline{x}^T A + \underline{b}^T + 0^T$$

$$g(\underline{x}) = 2 \underline{x}^T A + \underline{b}^T$$

$$\frac{\partial}{\partial \underline{x}} g(\underline{x}) = 2 \underline{x}^T [a_1, a_2, \dots, a_n]$$

$$= 2 [\underline{x}^T a_1, \underline{x}^T a_2, \dots, \underline{x}^T a_n]$$

$$\frac{\partial}{\partial \underline{x}} g(\underline{x}) = 2 \begin{bmatrix} \underline{a}_1^T \\ \underline{a}_2^T \\ \vdots \\ \underline{a}_n^T \end{bmatrix} = 2 A^T$$

$$\frac{\partial}{\partial \underline{x}} g(\underline{x}) = 2 A$$

$$\begin{aligned} \frac{\partial}{\partial \underline{x}} \underline{x}^T A \underline{x} \\ = \underline{x}^T (A + A^T) \\ = 2 \underline{x}^T A \end{aligned}$$

$$\frac{\partial}{\partial \underline{x}} \underline{b}^T \underline{x} = \underline{b}^T$$

$$\frac{\partial}{\partial \underline{x}} \underline{x}^T \underline{b} = \underline{b}^T$$

$$\frac{\partial}{\partial \underline{x}} \underline{x}^T A$$

$$\frac{\partial}{\partial \underline{x}} A^T \underline{x}$$

$$A = \begin{bmatrix} 2 & -1/2 \\ -1/2 & 1 \end{bmatrix}, a_1 = \begin{bmatrix} 2 \\ -1/2 \end{bmatrix}$$

$$a_2 = \begin{bmatrix} -1/2 \\ 1 \end{bmatrix}$$

Symmetric  
 $A^T = A$

$$\frac{\partial^2 f(x)}{\partial x^2} = 2A = H_{xx} f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

$$\rightarrow \frac{\partial^2 f}{\partial x^2} > 0, \frac{\partial^2 f}{\partial y^2} > 0 \Rightarrow$$

Any square matrix can be either

① Positive definite  $2A \succ 0$  positive semi-definite  
 a sq matrix  $A$  is PD if for all  $x \in \mathbb{R}^n$   $A \succ 0$   
 $\in \mathbb{R}^{n \times n}$   $x^T A x > 0$   $\forall x, x^T A x \geq 0$

② Negative definite  $A < 0 \Rightarrow x^T A x < 0$

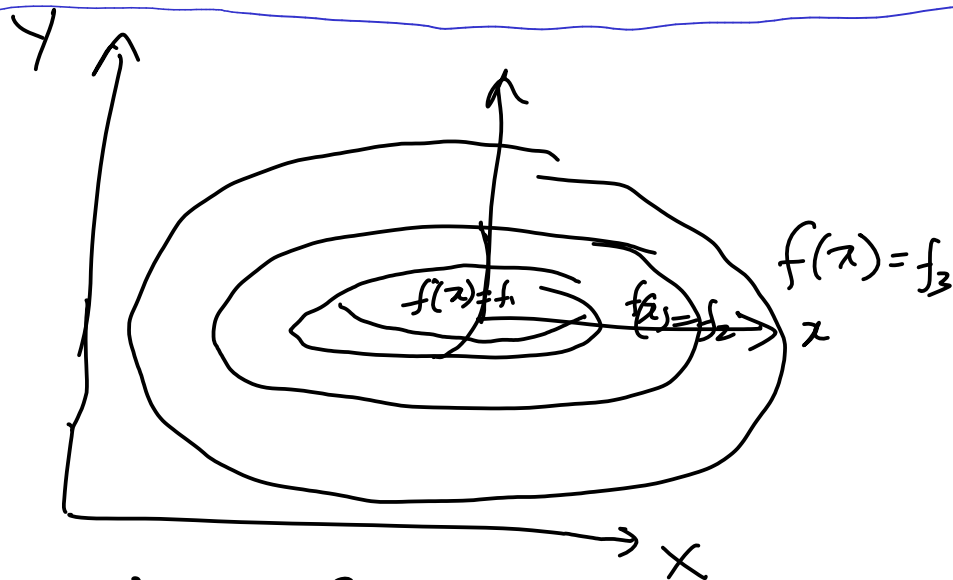
Negative semi-definite  $A \preceq 0 \Rightarrow x^T A x \leq 0$

$$\frac{\partial^2 f}{\partial x^2} < 0, \frac{\partial^2 f}{\partial y^2} < 0$$

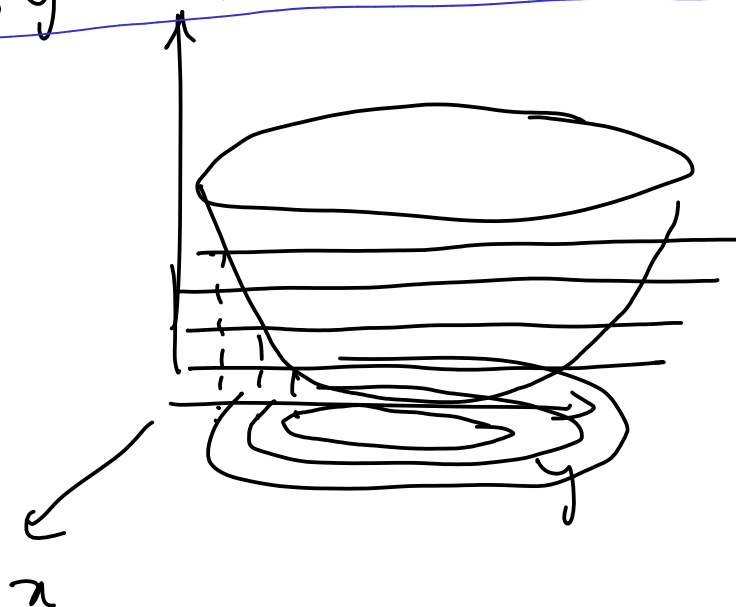
$$A = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & 0 \\ 0 & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

### ③ Indefinite

$$\frac{\partial^2 f}{\partial x^2} < 0, \quad \frac{\partial^2 f}{\partial y^2} > 0$$

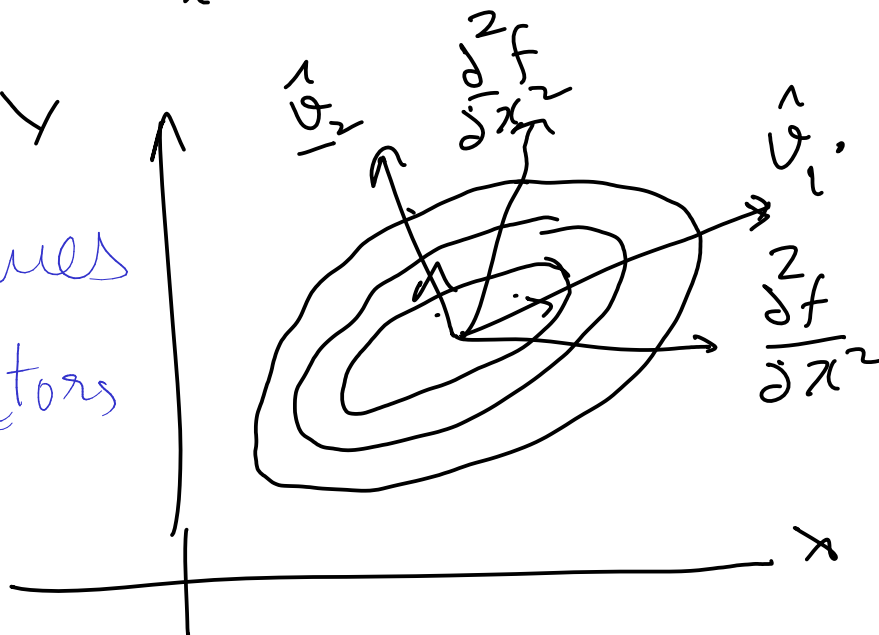


$$\frac{\partial^2 f}{\partial x^2} > 0, \quad \frac{\partial^2 f}{\partial y^2} > 0$$



$\underline{\hat{x}}^T A \underline{\hat{x}} > 0$  eigen values  
 $A \underline{\hat{v}} = \lambda \underline{\hat{v}}$  eigen vectors

A is PD if  $\lambda_i > 0$   
 all eigen values  $> 0$



$$f(x) = \underline{x}^T A \underline{x} + \underline{b}^T \underline{x} + c$$

$$H_{xx}f(x) = 2A$$

$$\left. \frac{\partial f(x)}{\partial x} \right|_{x^*} = 0$$

at extreme points

for all function

The extreme point is

(1) a minimum point if  $\underbrace{H_{xx}f(x)} > 0$  is PD or PSD

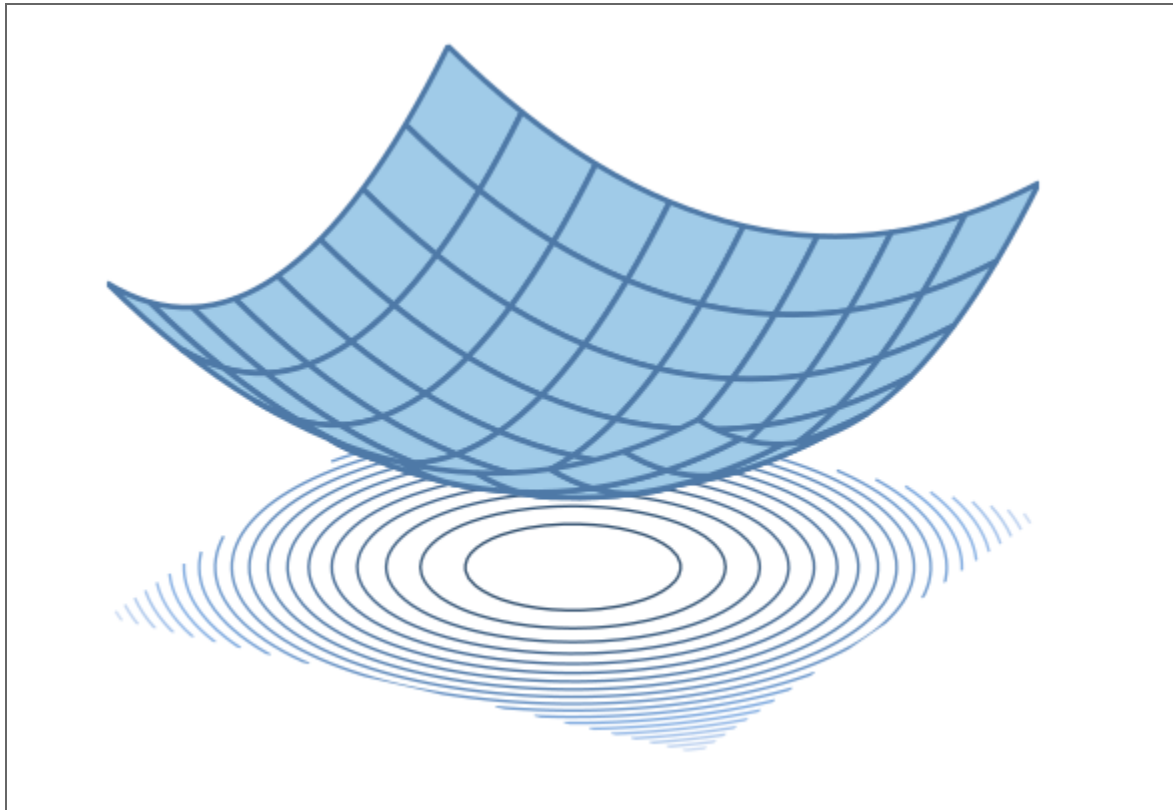
all eigen values  $> 0$

(2) a max " if  $H_{xx}f(x) < 0$  is ND or NSD

(3) either inflexion point or a saddle point

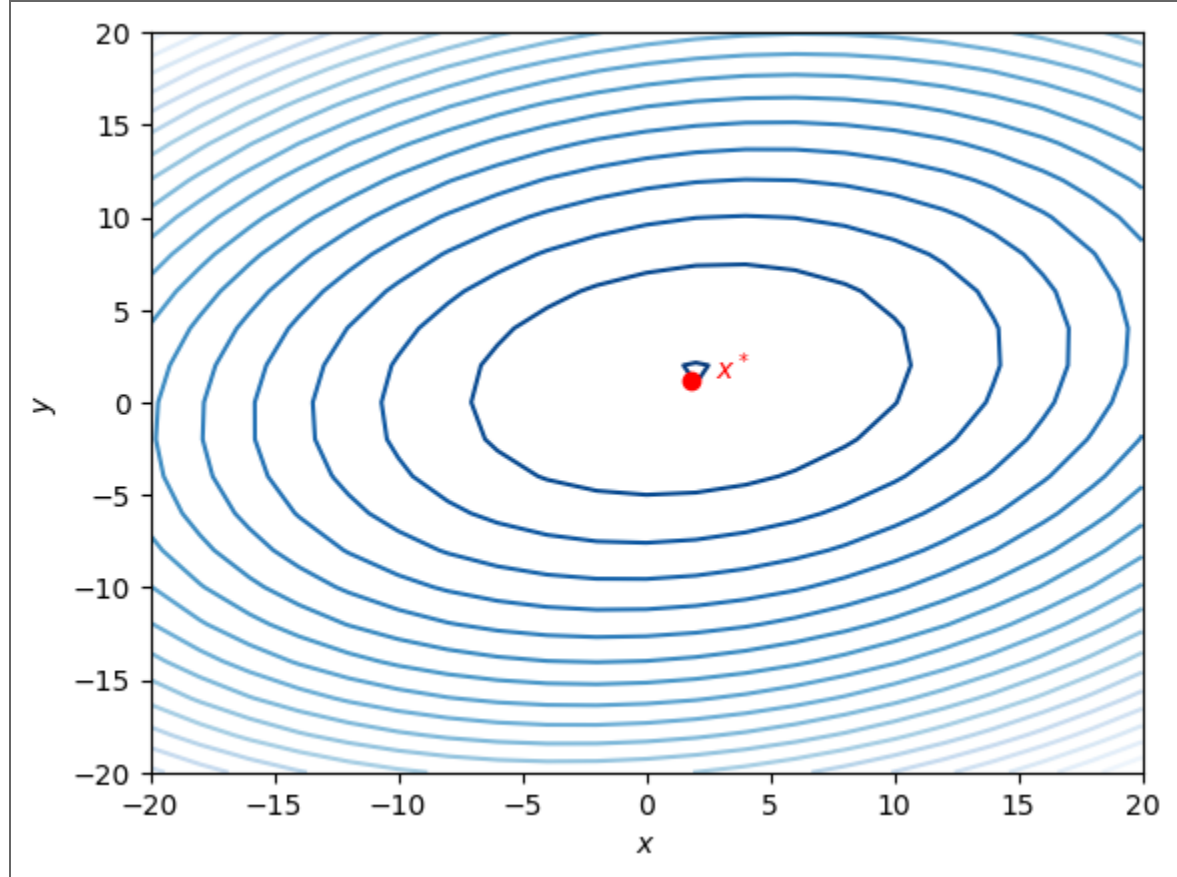


# Contour Plots



```
In [7]: def f(x, y): return 2*x**2 + 4*y**2 - x*y - 6*x - 8*y + 6  
        plot_contour(f)
```

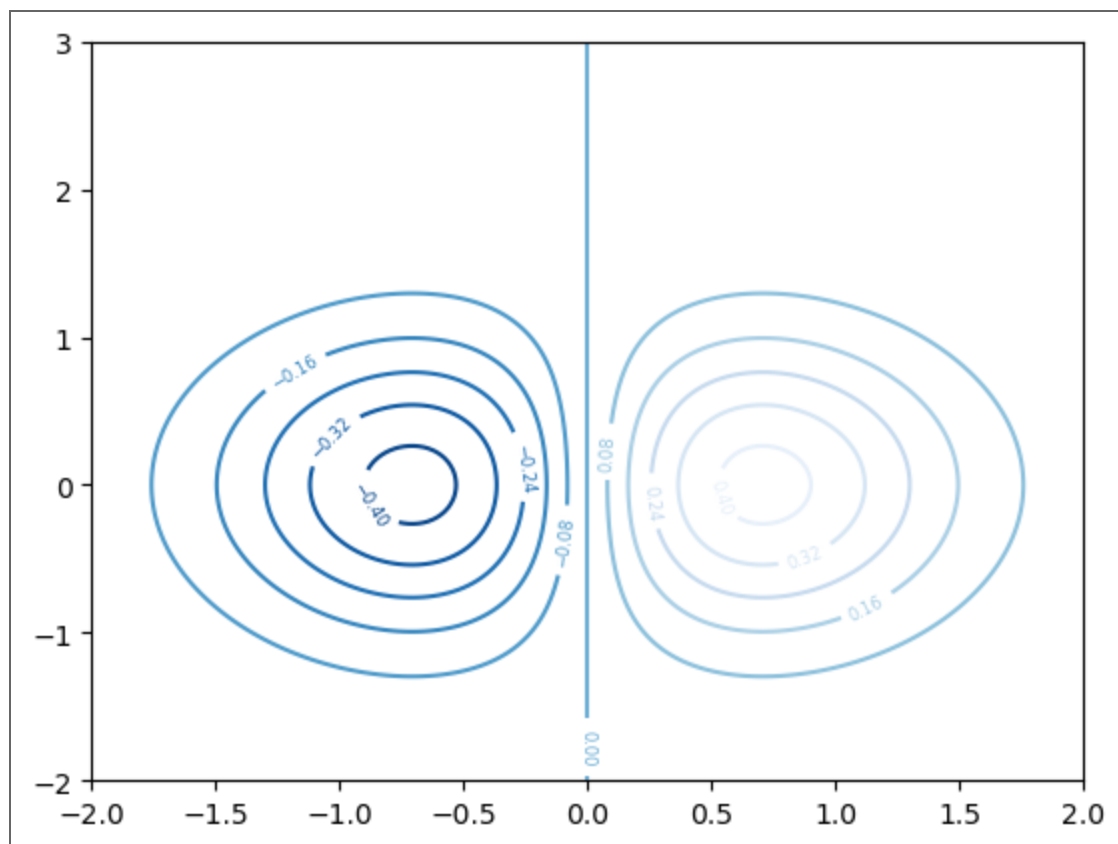




But how about other kinds of functions say:

$$\arg \min_x f(x) = x \exp(-(x^2 + y^2))$$

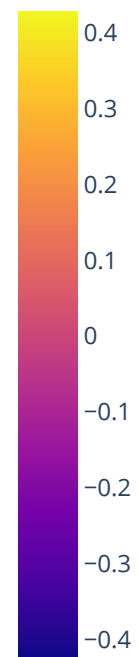
```
In [9]: def f(x,y): return x * np.exp(-(x**2 + y**2))  
        plot_contour(f)
```



```
In [10]: def plot_surface_3d(func):  
          x, y = np.mgrid[-2:2:201j,
```

```
                -2:3:201j]
f = func(x,y)
fig = go.Figure(data=[go.Surface(z=f, x=x, y=y,
                                contours = {
                                    "x": {"start": -2, "end": 2, "size": 0.2},
                                    "z": {"start": -2, "end": 2, "size": 0.2}
                                },
                                )])
fig.update_traces(contours_z=dict(show=True, usecolormap=True, project_z=True))
fig.show()
```

```
In [11]: plot_surface_3d(f)
```



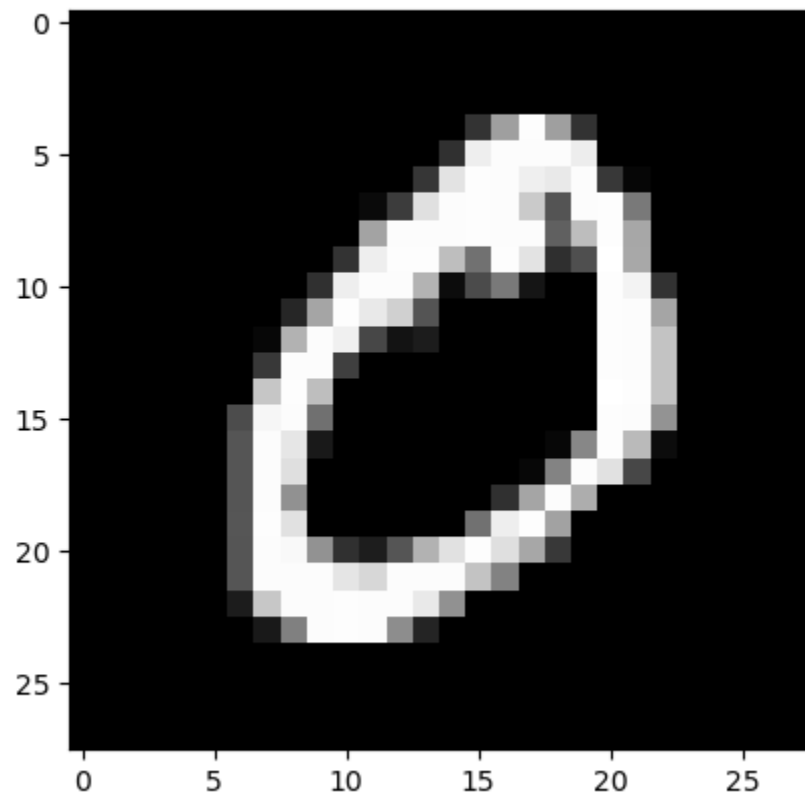
Taylor series

Taylor series vectorized



```
In [15]: zero_images_anim
```

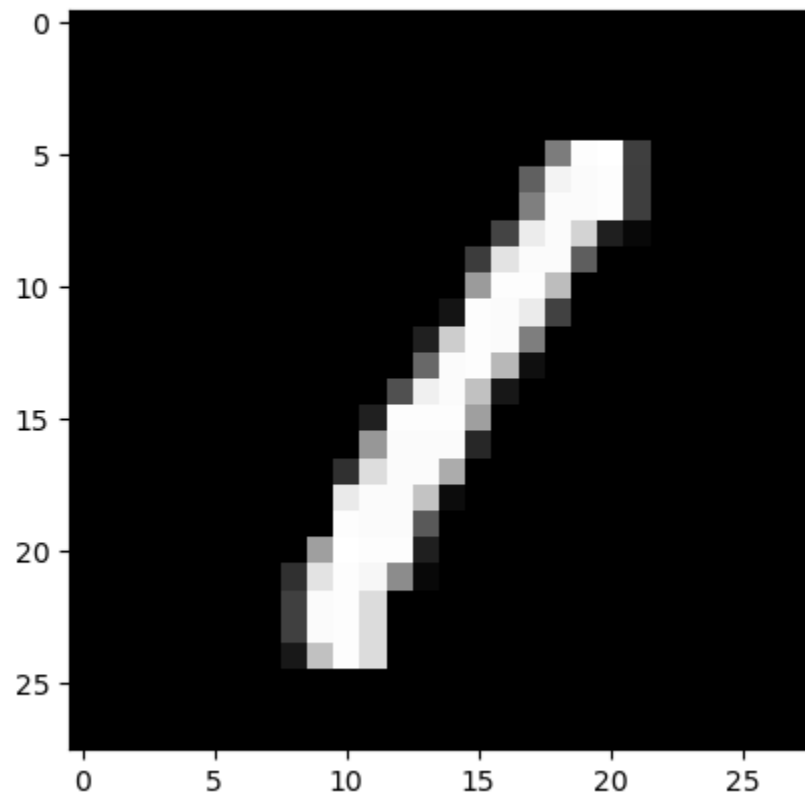
Out[15]:



☒ Once ☐ Loop ☐ Reflect

```
In [17]: one_images_anim
```

Out[17]:



☒ Once ☐ Loop ☐ Reflect

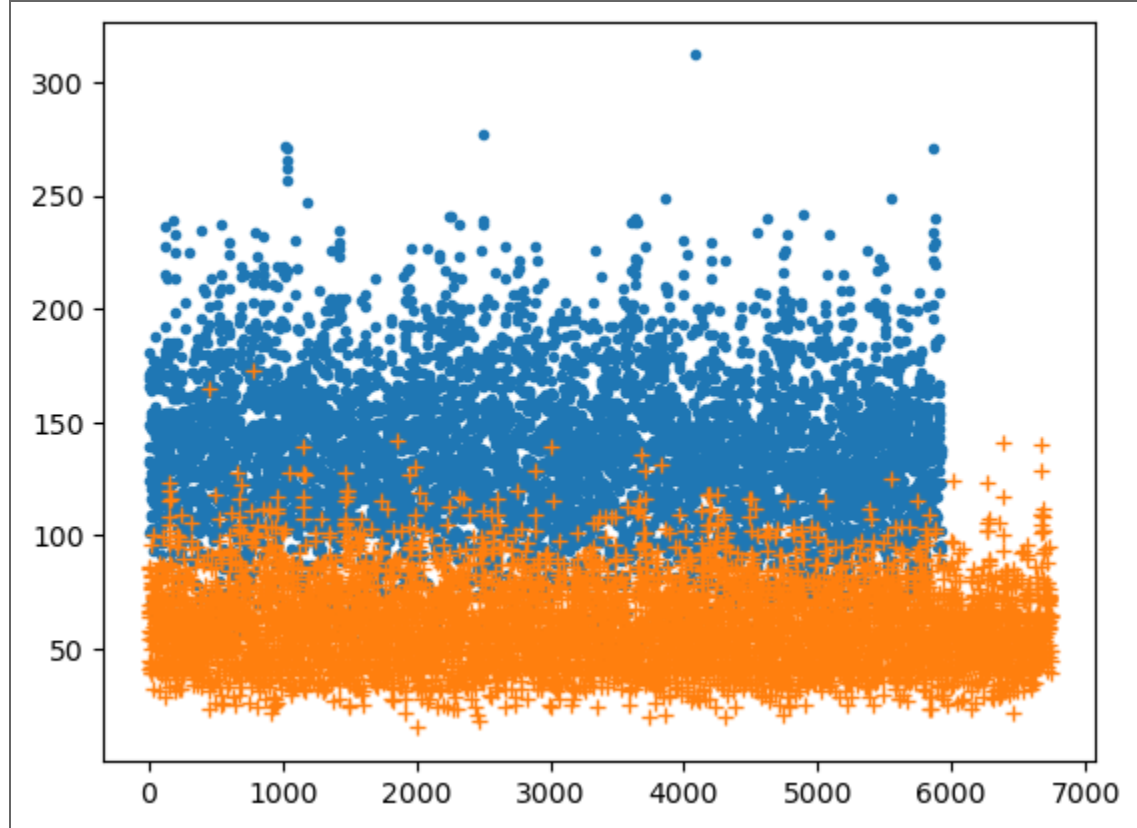
# What is a feature

Any property of data sample that helps with the task.

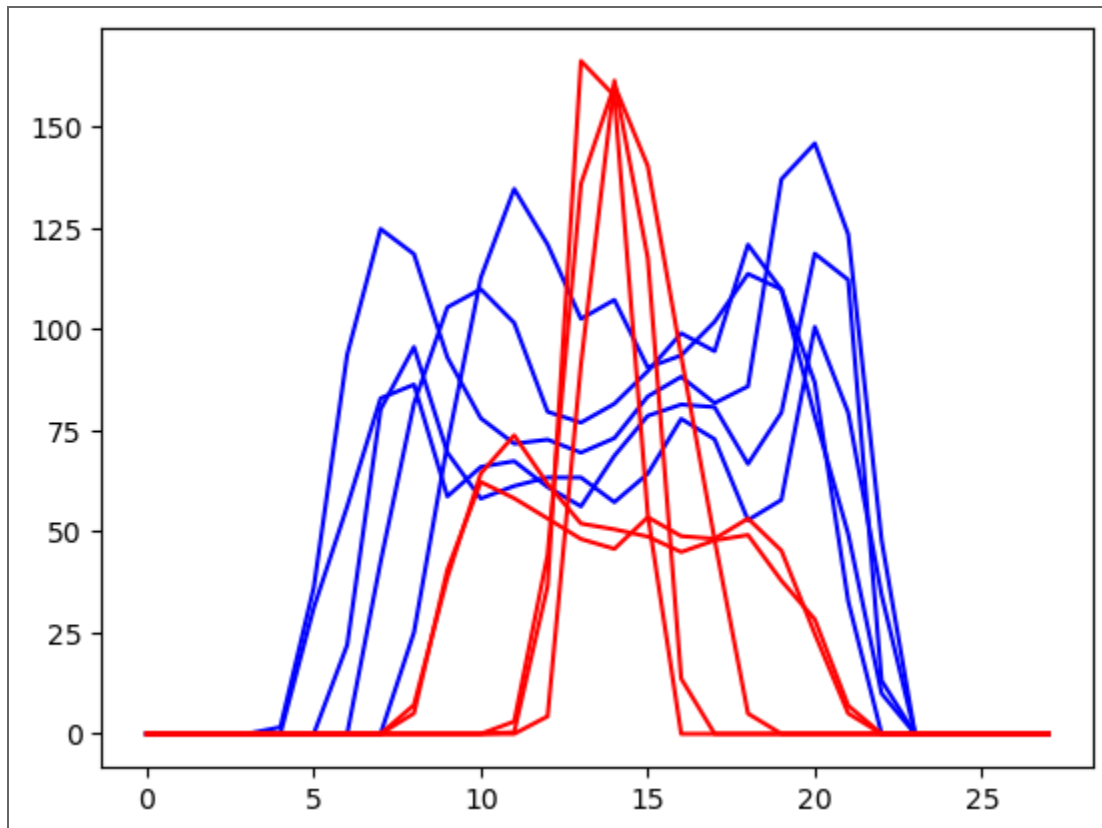
```
In [18]: def feature_n_pxls(imgs):  
         n, *shape = imgs.shape  
         return np.sum(imgs[:, :, :].reshape(n, -1) > 128, axis=1)  
  
n_pxls_zero_images = feature_n_pxls(zero_images)  
n_pxls_one_images = feature_n_pxls(one_images)  
fig, ax = plt.subplots()  
ax.plot(n_pxls_zero_images, '.')  
ax.plot(n_pxls_one_images, '+')
```

Out[18]:

```
[<matplotlib.lines.Line2D at 0x7f838d398490>]
```



```
In [19]: fig, ax = plt.subplots()
         for i in range(5):
             ax.plot(zero_images[i].mean(axis=0), 'b-')
         for i in range(5):
             plt.plot(one_images[i].mean(axis=0), 'r-')
```





```
In [20]: wts = zero_images[0].mean(axis=0)
          mean = (np.arange(wts.shape[0]) * wts).sum() / np.sum(wts)
          var = ((np.arange(wts.shape[0]) - mean)**2 * wts).sum() / np.sum(wts)
          var
```

Out[20]:

22.811061800377757

```
In [21]: def feature_y_var(img):
          wts = img.mean(axis=0)
          mean = (np.arange(wts.shape[0]) * wts).sum() / np.sum(wts)
          var = ((np.arange(wts.shape[0]) - mean)**2 * wts).sum() / np.sum(wts)
          return var
feature_y_var(zero_images[0]), feature_y_var(one_images[0])
```

Out[21]:

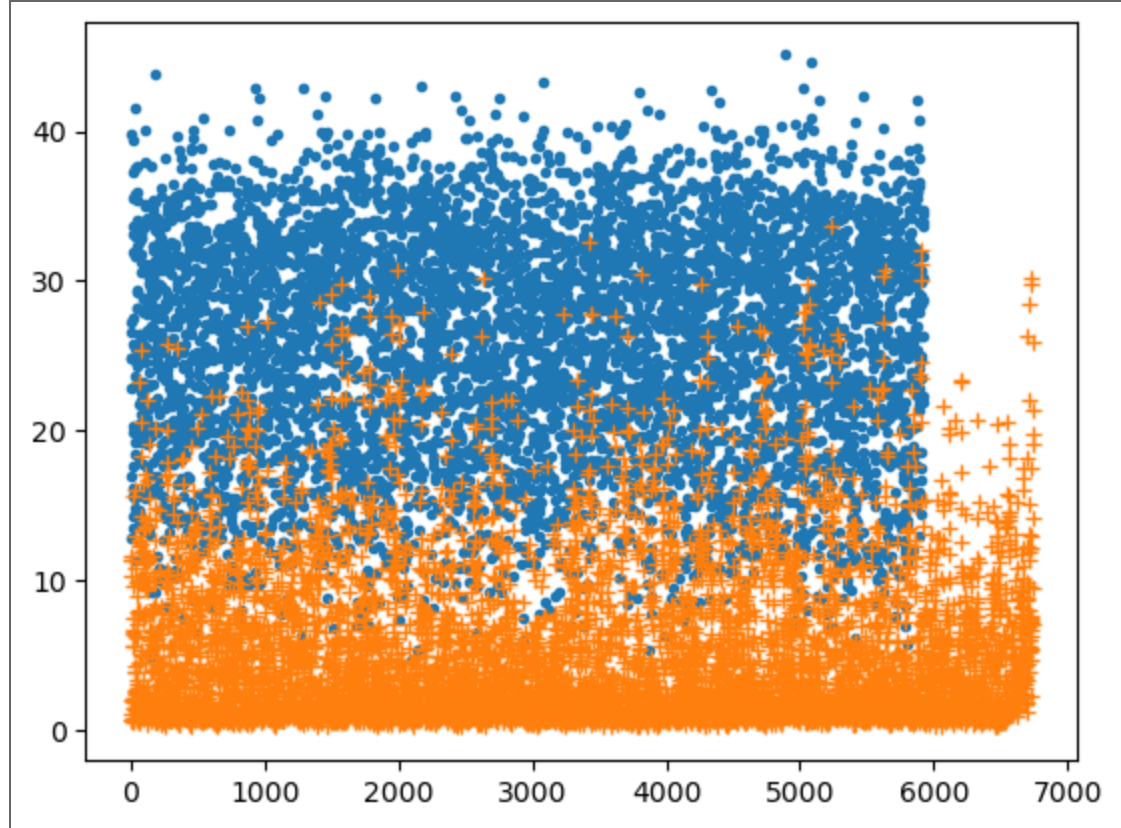
(22.811061800377757, 11.384958735403274)

```
In [22]: def feature_y_var(imgs):
          wts = imgs.mean(axis=-2)
          arange = np.arange(wts.shape[-1])
          mean = (arange * wts).sum(axis=-1) / wts.sum(axis=-1)
          mean = mean[:, np.newaxis]
          var = ((arange - mean)**2 * wts).sum(axis=-1) / wts.sum(axis=-1)
          return var

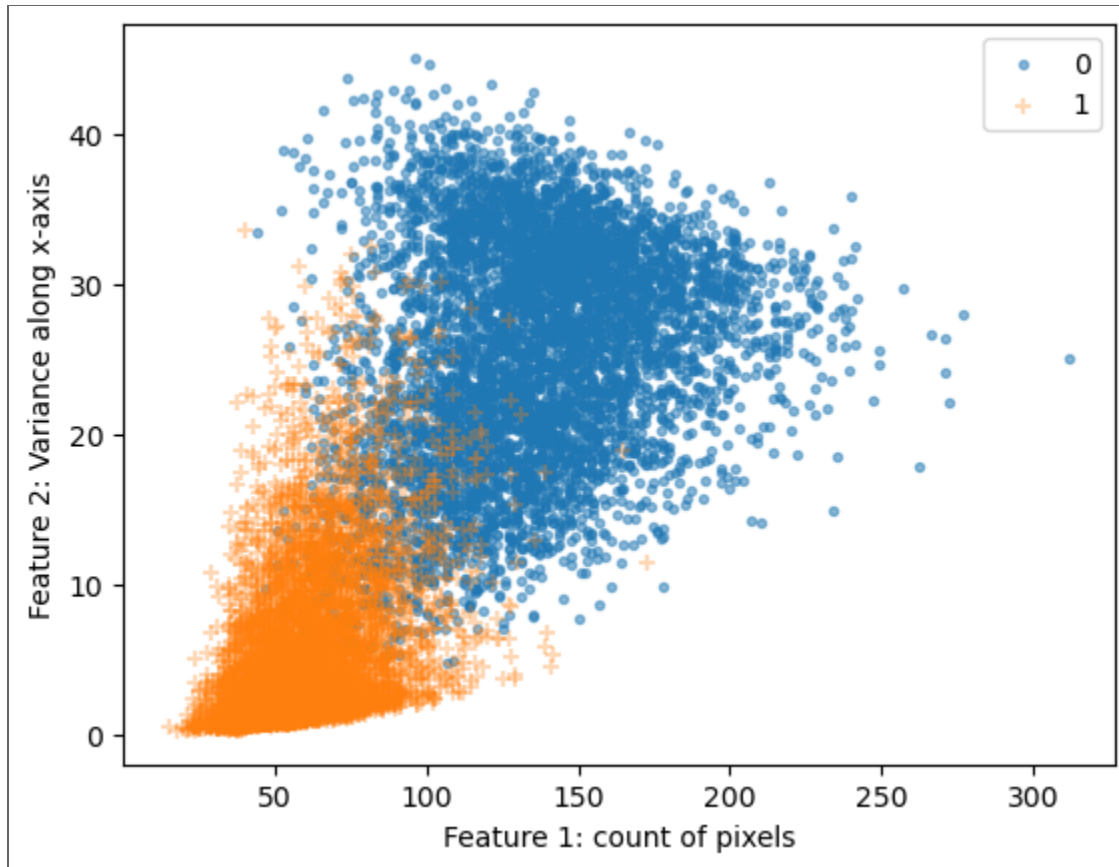
fig, ax = plt.subplots()
ax.plot(feature_y_var(zero_images), '.')
ax.plot(feature_y_var(one_images), '+')
```

Out[22]:

[<matplotlib.lines.Line2D at 0x7f838d58a1a0>]



```
In [24]: fig, ax = plt.subplots()
         draw_features(ax, zero_features, one_features)
         plt.show()
```



```
In [27]: def error(X, Y, bfm):
         # YOUR CODE HERE
         raise NotImplementedError()

def grad_error(Xw, Yw, bfm):
    # YOUR CODE HERE
```

```
raise NotImplementedError()
```

```
def train(X, Y, lr = 0.1):  
    # YOUR CODE HERE  
    raise NotImplementedError()
```

```
OPTIMAL_BFM, list_of_bfms, list_of_errors = train(X, Y)  
fig, ax = plt.subplots()  
ax.plot(list_of_errors)  
ax.set_xlabel('t')  
ax.set_ylabel('loss')  
plt.show()
```

-----  
-----

NotImplementedError Traceback (most recent call last)

Cell In[27], line 13

```
9 def train(X, Y, lr = 0.1):  
10     # YOUR CODE HERE  
11     raise NotImplementedError()  
---> 13 OPTIMAL_BFM, list_of_bfms, list_of_errors = train(X, Y)  
14 fig, ax = plt.subplots()  
15 ax.plot(list_of_errors)
```

Cell In[27], line 11, in train(X, Y, lr)

```
9 def train(X, Y, lr = 0.1):  
10     # YOUR CODE HERE  
---> 11     raise NotImplementedError()
```

NotImplementedError:

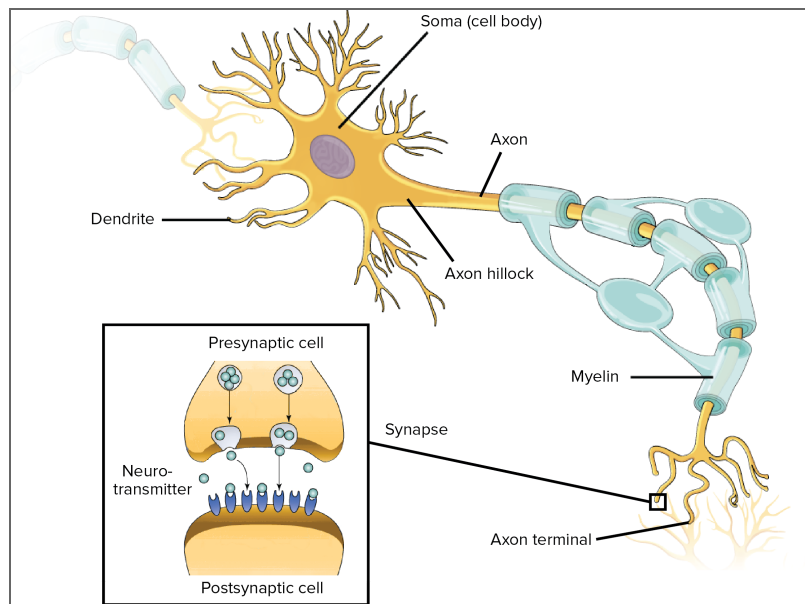


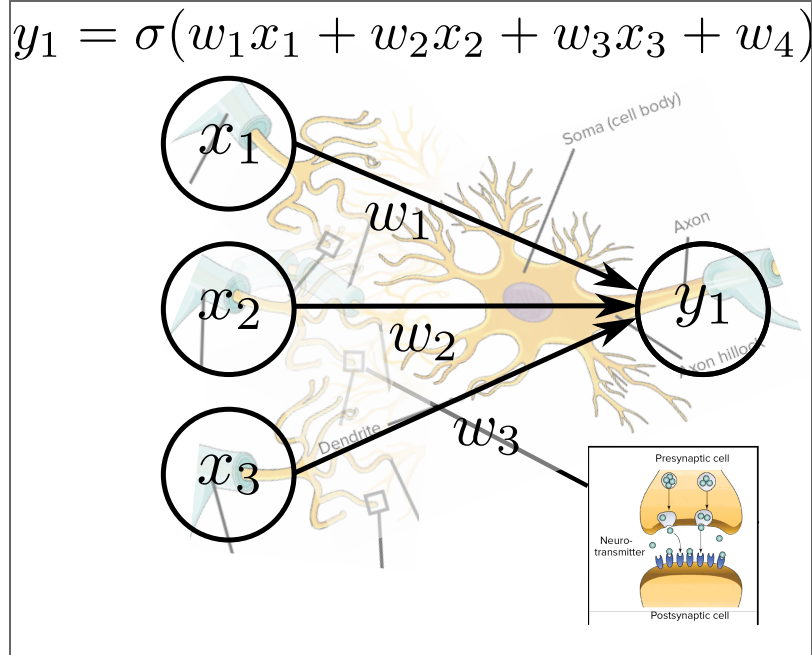
# Single Layer Neural Networks

## Read Chapter 2 and 3 of UDL Book

*Notes* Single Layer Neural Networks are simplest kind of neural networks. But before we dive into single layer neural networks, may be we should focus on the name *neural* networks. The name neural networks comes from biological neurons.

Similarities between Artificial neuron and Biological neuron





1. The excitation or firing of a biological neuron can be equated to a high positive value of units ( $x_1, x_2, x_3$ ) in artificial neurons.
2. The synapse in biological neuron determines which input excitations will have excitatory or inhibitory impact on output excitations. Synapses can strengthen, weaken, disconnect or form new connections during biological learning. Similarly to excitatory synapses, positive weights can cause positive input values to contribute to positive output values.
3. Usually multiple excitatory inputs are required to excite the output neuron.

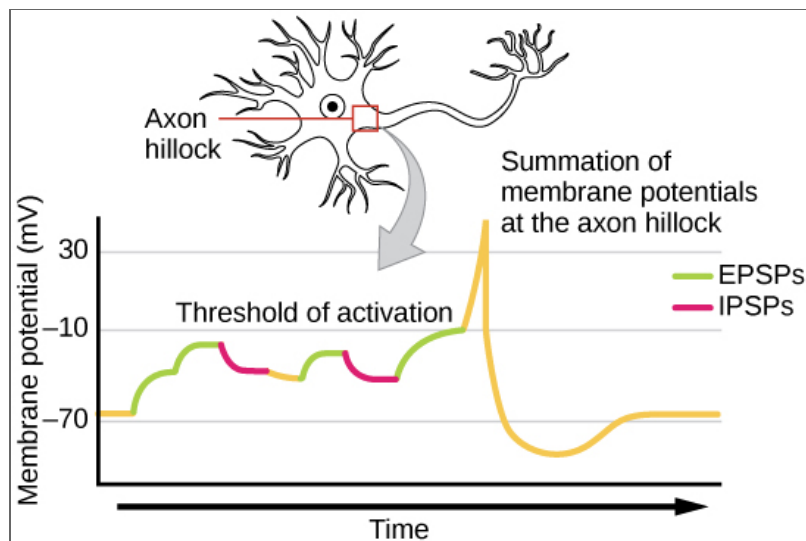
References:



# 1. <https://openstax.org/books/biology/pages/35-2-how-neurons-communicate>

## Differences

1. Biological neuron is all or None
2. Biological neuron has a time component



## notes

1. The activity of the biological neuron is an "all-or-none" process. Artificial activations are typically continuous range. Even when sigmoid or softmax nonlinearities.
  2. Biological neuron has time dynamics. The input activations are integrated over time.
-

---

