EECS 3405 Final Review Questions

Short Answer / Concept Checking

INDEX

Perceptron Algorithm: How long will the Perceptron Algorithm run?

If the training set is linearly separable: the perceptron algorithm is guaranteed to terminate after a finite number of updates, and it will return a linear model that perfectly classifies the training set

Otherwise: the algorithm would loop forever (or until a maximum number of iterations as specified.

Does Linear Regression perform well for classification problems? Why or why not?

Linear regression can be easily solved by a closed-form solution, but it may not yield good performance for classification problems. The main reason is that the square error used in training models does not match well with our goal in classification. For classification problems, our primary concern is to reduce the misclassification errors rather than the reconstruction error.

MCE: How is each classification error calculated in MCE?

$$-y_i w^T x_i = \begin{cases} > 0 \Rightarrow \text{mis} - \text{classification} \\ < 0 \Rightarrow \text{correct classification} \end{cases}$$

y_i: is the true label

w^T: the models weight vector (transpose)

x_i: feature vector at i-th data point

Why is y_i negative here?

The reason that y_i is negative is so that if the resulting scalar positive, it's a misclassification, if negative then it's correct

How is MCE specified using the sigmoid function for smooth approximation?

$$E_1(w) = \min_w \sum_{i=i}^{N} \mathrm{l}(-y_i w^T x_i)$$

How does the heavyside/step function work here: $E_0(w) = \sum_{i=i}^{N} H(-y_i w^T x_i)$

```
H(·) is the "heavyside"/ "Step" function: H(z) is >0, it outputs
                        1 and 0 if correct
```

What is the disadvantage of using the Heavyside / Step function?

It's not differentiable and thus not useful for implementing optimization methods

How to determine the derivative of this?

L = sigmoid, consider derivative of sigmoid:

$$\frac{d}{dx}\sigma(x) = \sigma(x)\big(1-\sigma(x)\big)$$

However, $x = (-y_i w^T x_i)$ thus its derivative is accounted for at the end via chain rule

Thus, the derivative of $l(-y_i w^T x_i)$ =

$$\frac{\partial}{\partial w} l(-y_i w^T x_i) = l(-y_i w^T x_i)\big(1 - l(-y_i w^T x_i)\big) \cdot \frac{\partial}{\partial w}(-y_i w^\top x_i)$$
$$= -y_i l(-y_i w^T x_i)\big(1 - l(-y_i w^T x_i)\big)(x_i)$$

How is MCE calculated?

Using Gradient Descent / SGD :

$$\frac{\partial E_1(w)}{\partial w} = -\sum_{i=1}^{N} y_i l(-y_i w^T x_i)\big(1 - l(-y_i w^T x_i)\big)(x_i)$$

What is the alternate way for MCE, and what is the advantage?

Let w' = -w, then:

$$\mathbf{w'}_{\text{MCE}} = \arg\min_{\mathbf{w'}} \; E_2(\mathbf{w'}) = \arg\min_{\mathbf{w'}} \; \sum_{i=1}^{N} l(y_i \mathbf{w'}^\top \mathbf{x}_i)$$

Simpler gradient:

$$\frac{\partial E_2(w)}{\partial w} = \sum_{i=1}^{N} y_i l(-y_i w'^T x_i)\big(1 - l(y_i w'^T x_i)\big)(x_i)$$

How is testing done this way?

$$y = \text{sign}(w^T x) = \text{sign}(-w'^T x) = \begin{cases} +1 \text{ if } w'Tx < 0 \\ -1 \text{ otherwise} \end{cases}$$

Extend the MCE method in Section 6.3 to deal with pattern-classification problems involving K > 2 classes.

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

We calculate (correct − largest wrong): $g_y(x) - \max_{j \neq y} g_j(x)$

$$g_1(x), g_2(x), \ldots, g_K(x)$$

$$d_y(x) = -[g_y(x) - \max_{j \neq y} g_j(x)]$$

Using β to control how "sharp" the function is, we then use this difference instead to show the error for a particular point

$$l(x) = \frac{1}{1+e^{-\beta d_y(x)}}$$

And for the whole data set:

$$E_k(W) = \sum_{i=1}^{N} L(x_i) = \sum_{i=1}^{N} \frac{1}{1+e^{-\beta d_y(x_i)}}$$

What is being calculated in MCE and how does it differ from Logistic Regression?
MCE calculaltes the soft error to misclassify. Whereas logistic regression calculates the probability that you will correctly classify the sample.
How is this correct classification probability calculated?
$\sigma(-y_i w^T x_i)$ is the "smooth approximation" of the probability of miscalculation. Thus:
$1 - \sigma(-y_i w^T x_i) = \sigma(y_i w^T x_i)$. We use this to calculate the probability of correctly classifying the entire dataset.
What is the algorithm of logistic regression?

$$w_{LR} = arg \max_w L(w) = arg \max_w \prod_i^N \sigma(y_i w^T x_i)$$

When implementing logistic regression, what direction to you move in regards to the gradient?
Its essentially gradient *ascent*: moving up along the direction of the gradient
How is this represented in the "log form" of the same thing?

$$\ln\big(L(w)\big) = \ln\left(\prod_i^N \sigma(y_i w^T x_i)\right)$$

Log property: ln(ab)=lna + lnb

$$ln(a \cdot b \cdot c \ldots.) = lna + lnb + lnc + \cdots$$

How is the gradient of Logistic Regression computed?
Taking the gradient of this wrt w:

$$\frac{\partial \ln\big(L(w)\big)}{\partial w} = \sum_{i=1}^N \frac{\partial}{\partial w} \ln\big(\sigma(y_i w^T x_i)\big)$$
$$= \sum_{i=1}^N \frac{1}{\sigma(y_i w^T x_i)} \cdot \frac{\partial}{\partial w} \sigma(y_i w^T x_i)$$

Derivative of sigmoid function (plus chain rule):

$$= \sum_{i=1}^N \frac{1}{\sigma(y_i w^T x_i)} \cdot \sigma(y_i w^T x_i)\big(1 - \sigma(y_i w^T x_i)\big)\,(y_i x_i)$$
$$= \sum_{i=1}^N \frac{1}{1} \cdot \big(1 - \sigma(y_i w^T x_i)\big)\,(y_i x_i)$$
$$= \sum_{i=1}^N \big(1 - \sigma(y_i w^T x_i)\big)\,(y_i x_i)$$

Note: y_i is a scalar, thus acts like a coefficient:

$$\frac{\partial \ln\big(L(w)\big)}{\partial w} = \sum_{i=1}^N y_i\big(1 - \sigma(y_i w^T x_i)\big)x_i$$

How can logistic regression vs. MCE be compared?

- MCE learning focuses more on the boundary cases
- logistic regression generates significant gradients for all mis-classified samples: prone to outliers but faster convergence can be extended to multi-class using the softmax function

Aspect	MCE	Logistic Regression
Focus	More on boundary cases	Generates significant gradients for all mis-classified samples
Convergence	Slower	Faster convergence
Outliers	More robust	Prone to outliers

How can they be extended to multi-class use ?

softmax function

What are the alphas in SVM problem? How can there be more than one Lagrange multiplier?

Each inequality constraint, $y_i(w \cdot x_i + b) \geq 1$, has a Lagrange multiplier α_i such that $\alpha_i \geq 0$. We modify the inequality constraint to compare against zero($y_i(w^T x_i + b) - 1 \geq 0$).

For just one data point:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w^T w + \lambda[1 - y_i(w^T x_i + b)]$$

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w^T w - \lambda[y_i(w^T x_i + b) - 1]$$

Change to alpha

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w^T w - \alpha_i[y_i(w^T x_i + b) - 1]$$

For all the data points:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i[y_i(w^T x_i + b) - 1]$$

How to Derive SVM2 Dual Problem from Langrangian:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}\, w^T w - \sum_{i=1}^{N} \alpha_i (y_i (w^T x - b) - 1)$$

Solution:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i y_i w^T x_i + \alpha_i y_i b - \alpha_i$$

B is just a constant:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i y_i w^T x_i - b \sum_{i=1}^{N} \alpha_i y_i + \sum_{i=1}^{N} \alpha_i$$

$$\begin{aligned} w^T w &= \left(\sum_{i=1}^{N} \alpha_i y_i x_i \right)^T \left(\sum_{j=1}^{N} \alpha_j y_j x_j \right) \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i y_i x_i^T \alpha_j y_j x_j \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j \end{aligned}$$

Thus: $\frac{1}{2} \mathrm{w}^{\mathrm{T}}\, \mathrm{w} \;=\; \frac{1}{2} \sum_{\mathrm{i}=1}^{\mathrm{N}} \sum_{j=1}^{N} \alpha_i\, \alpha_j\, y_i\, y_j\, x_i^T\, x_j$

Next term:

$$\begin{aligned} -\sum_{i=1}^{N} \alpha_i y_i w^T x_i &= -\sum_{i=1}^{N} \alpha_i y_i \left(\sum_{j=1}^{N} \alpha_j y_j x_j \right)^T x_i \\ &= -\sum_{i=1}^{N} \alpha_i y_i \sum_{j=1}^{N} \alpha_j y_j x_j^T x_i \\ &= -\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_j^T x_i \\ &= -\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j \end{aligned}$$

Now we have thes terms expandeded we next take the gradient:

We take partial derivatives:

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \rightarrow w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0 \rightarrow w^* = \sum_{i=1}^{N} \alpha_i y_i x_i$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0 = \alpha^\top y$$

Using this with the original langrangian:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i y_i w^T x_i - b \sum_{i=1}^{N} \alpha_i y_i + \sum_{i=1}^{N} \alpha_i$$

We can make substitutions:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \left(\sum_{i=1}^{N} \alpha_i y_i x_i \right)^\top \left(\sum_{i=1}^{N} \alpha_i y_i x_i \right) - \sum_{i=1}^{N} \alpha_i y_i \left(\sum_{i=1}^{N} \alpha_i y_i x_i \right)^\top x_i - b(0) + \sum_{i=1}^{N} \alpha_i$$

And combine like terms:

$$\mathcal{L}(w^*, b, \alpha) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{N} \alpha_i$$

$$= -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{N} \alpha_i$$

$$\max_{\alpha_i \geq 0} \left[\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^\top x_j \right] \; s.t. \; \alpha^\top y = 0$$

Considering the constraint $0 = \boldsymbol{\alpha}^\top \boldsymbol{y}$ (vectorized form of $\sum_{i=1}^{N} \boldsymbol{\alpha_i y_i}$), why is there is constraint?
Each α_i "pulls" the hyperplane toward its point. $y_i = +1$ for positive class, -1 for negative class.
Example:

Point	Class (y_i)	α_i
x_1	+1	0.5
x_2	+1	1.0
x_3	-1	1.5

$$\sum_i \alpha_i y_i = (0.5)(+1) + (1.0)(+1) + (1.5)(-1) = 0.5 + 1.0 - 1.5 = 0$$

This ensures the hyperplane sits exactly in the "middle" between the closest points of each class—the support vectors.

When is a 0 versus greater than 0?

α_i measures how important point x_i is in defining the decision boundary.

- $\alpha_i = 0$: Point x_i is **NOT a support vector**. It's far from the boundary and doesn't matter.
- $\alpha_i > 0$: Point x_i **IS a support vector**. It lies on the margin boundary and is critical!
- **Larger** α_i: Point is more "influential" in determining w

How is α_i found?
Different methods exist, but Quadratic program (SVM2) can be used

What is the optimal weight vector as derived from the dual problem?

$$w^* = \sum_{\{i=1\}}^{N} \alpha_i^* y_i x_i$$

After solving the quadratic programming problem to obtain all $\boldsymbol{\alpha^* = \{\alpha_i^* \mid i = 1, 2, \dots, N\}}$ and the optimal bias term $\boldsymbol{b^*}$, explain why nonlinear SVMs can be represented as:

$$y = sign\left(\sum_{i=1}^{N} \alpha_i^* y_i \Phi(x_i, x) + b^*\right)$$

Consider the optimum dividing line between the two classes: $\boldsymbol{y = w^{*\top} h(x) + b^*}$
We also us a mapping function: $x \rightarrow h(x)$
Subbing in our optimal weight vector:

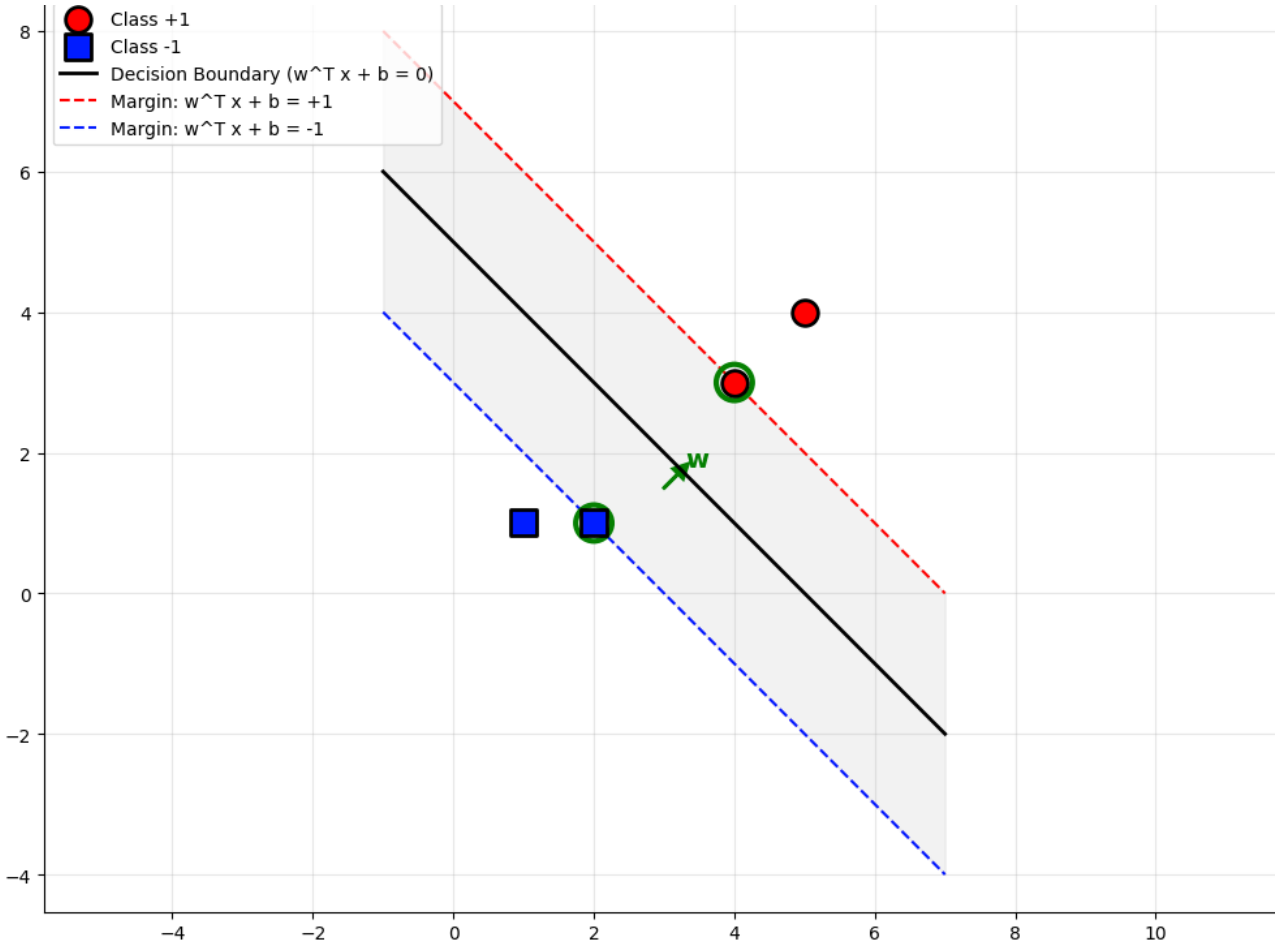
$$\boldsymbol{y} = \left(\sum_{\{i=1\}}^{N} \alpha_i^* y_i h(x_i)\right)^{\top} h(x) + b^*$$

$$= \sum_{\{i=1\}}^{N} \alpha_i^* y_i \phi(x_i, x) + b^*$$

Why do we use the sign() function? Because all we care about is whether its greater than or less than 0:

$$f(x) > 0 \rightarrow +1$$

$$f(x) < 0 \rightarrow -1$$



Assume we know the above and which points are support vectors. As well we know the labels for our two classes:

Red Positive:	**blue negative:**
Point [4 3]:	Point [1 1]:
Point [5 4]:	Point [2 1]:

How can we calculate the vector *w* and solve the optimization problem:

$$\min_{w,b} \frac{1}{2} w^T w$$

subject to:

$$y_i(w^T x_i + b) \geq 1, \forall i \in 1,2,\dots,N$$

Assume we know the above and which points are support vectors. As well we know the labels for our two classes:

Red Positive:	**blue negative:**
Point [4 3]:	Point [1 1]:
Point [5 4]:	Point [2 1]:

How can we calculate the vector *w* and solve the optimization problem:

$$\min_{w,b} \frac{1}{2} w^T w$$

subject to:

$$y_i(w^T x_i + b) \geq 1, \forall i \in 1,2,\dots,N$$

Solution:

Positive support vector where y = +1: $w^T x + b \geq 1$

$$w^1(4) + w^2(3) + b = 1$$

Negative support vector where y = -1: $w^T x + b \leq -1$

$$w_1(2) + w_2(1) + b = -1$$

subtract the second from the first:

$$4w_1 + 3w_2 + b - 2w_1 - w_2 - b = 1 - (-1)$$
$$2w_1 + 2w_2 = 2$$
$$w_1 + w_2 = 1$$

Consider: $w^T w$, this is equivalent to $||w||^2 = w_1^2 + w_2^2$ (as it is a two row vector in this example)

Apply Lagrange multipliers to $f(w_1, w_2) = w_1^2 + w_2^2$ and $g(w_1, w_2) = w_1 + w_2 - 1 = 0$

$$w_1^2 + w_2^2 = \lambda(w_1 + w_2 - 1)$$

$$\mathcal{L}(w_1, w_2, \lambda) = w_1^2 + w_2^2 - \lambda(w_1 + w_2 - 1)$$
$$= w_1^2 + w_2^2 - \lambda w_1 - \lambda w_2 + \lambda$$
$$\frac{\partial \mathcal{L}}{\partial w_1} = 2w_1 - \lambda = 0$$

Thus: $w_1 = \frac{\lambda}{2}$

$$\frac{\partial \mathcal{L}}{\partial w_2} = 2w_2 - \lambda = 0$$

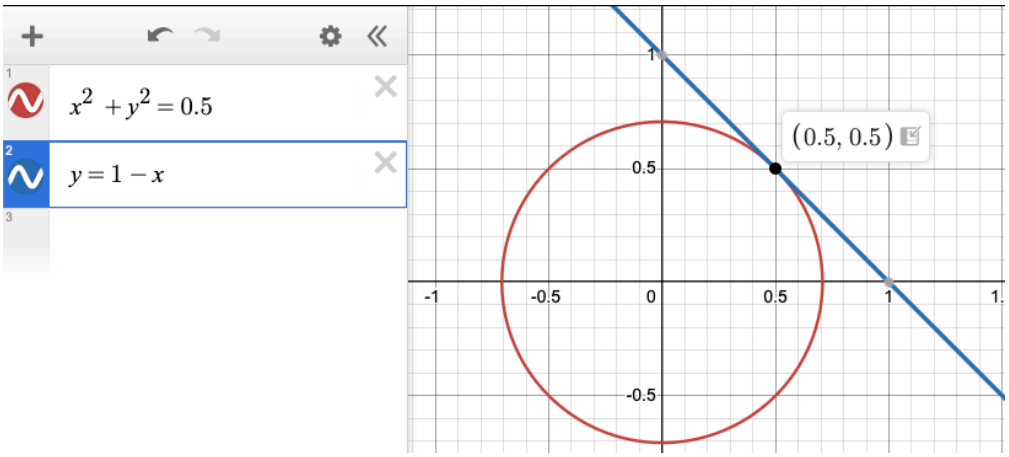
Thus: $w_2 = \frac{\lambda}{2}$

From the original equation $w_1 + w_2 = 1$, we can apply:

$$\frac{\lambda}{2} + \frac{\lambda}{2} = 1$$
$$2\lambda = 2$$
$$\lambda = 1$$

Furthermore, because both w equal $\frac{\lambda}{2}$: $w_1 = w_2 = \frac{1}{2}$

The vector *w*, with points w_1, w_2, is thus $\begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$



$$\nabla f(w_1, w_2) = \begin{bmatrix} 2w_1 \\ 2w_2 \end{bmatrix}$$
$$\nabla g(w_1, w_2) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

To find the line between the two classes we also need to find the bias b:

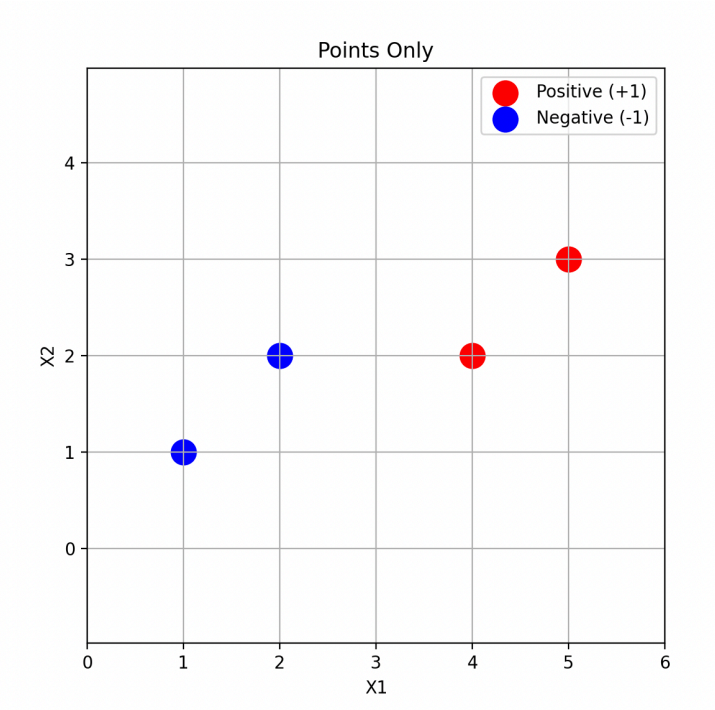
Using the negative class support vector where y=-1: point (2,1)

$$y(w^T x + b) = 1$$
$$-1\left(\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \end{bmatrix}\begin{bmatrix} 2 \\ 1 \end{bmatrix} + b\right) = 1$$
$$-1\left(\frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 1 + b\right) = 1$$
$$-1\left(\frac{3}{2} + b\right) = 1$$
$$-\frac{3}{2} - b = 1$$
$$-\frac{5}{2} = b = -2.5$$

Consider the negative support vector: $w^T x + b = -1$, and the positive: $w^T x + b = 1$
We need the distance from between those two points:

$$d = \frac{|1 - (-1)|}{||w||} = \frac{2}{||w||}$$
$$= \frac{2}{\sqrt{\frac{1}{2}^2 + \frac{1}{2}^2}}$$
$$= \frac{2}{\sqrt{\frac{1}{2}}} = \frac{2}{\frac{1}{\sqrt{2}}} \cdot \frac{\sqrt{2}}{\sqrt{2}} = 2\sqrt{2} \approx 2.828$$
$$\gamma = \sqrt{2}; margin = 2\sqrt{2}$$

Coincidentally, because this is a simple example [2,1] and [4,3] are on a line perpendicular to the decision boundary, but this is not always the case.



Consider a linear SVM with the following training data

Red Positive (+1):

- Point [4,2]
- Point [5,3]

Blue Negative (-1):

- Point [1,1]
- Point [2,2]

Task: Solve the SVM dual optimization problem to find the optimal Lagrange multipliers α_i:

$$\max_{\alpha_i \geq 0} [\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j]$$

subject to:

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

(Note: just set up the lagrange multiplier, beyond that will not be needed)

Solution:

Use matrix form for optimization:

$$\max_{\alpha} 1^{\top}\alpha \; -\frac{1}{2}\alpha^{\top}Q\alpha$$

subject to:

$$y^{T}\alpha \; = \; 0$$
$$\alpha \; \geq \; 0$$

$$X = \begin{bmatrix} 4 & 2 \\ 5 & 3 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}; \quad Y = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}; \; YY^{\top} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix} =$$

$$yy^{\top} = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix} \; ; \quad X^{\top}X \; = \begin{bmatrix} 20 & 26 & 6 & 12 \\ 26 & 34 & 8 & 16 \\ 6 & 8 & 2 & 4 \\ 12 & 16 & 4 & 8 \end{bmatrix}$$

Note: Calculating $X^{T}X$

$x_1^{\top}x_1 = \begin{bmatrix} 4 & 2 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 2 \end{bmatrix} = 16 + 4 = 20$

$x_1^{\top}x_2 = \begin{bmatrix} 4 & 2 \end{bmatrix} \cdot \begin{bmatrix} 5 \\ 3 \end{bmatrix} = 20 + 6 = 26$ etc.

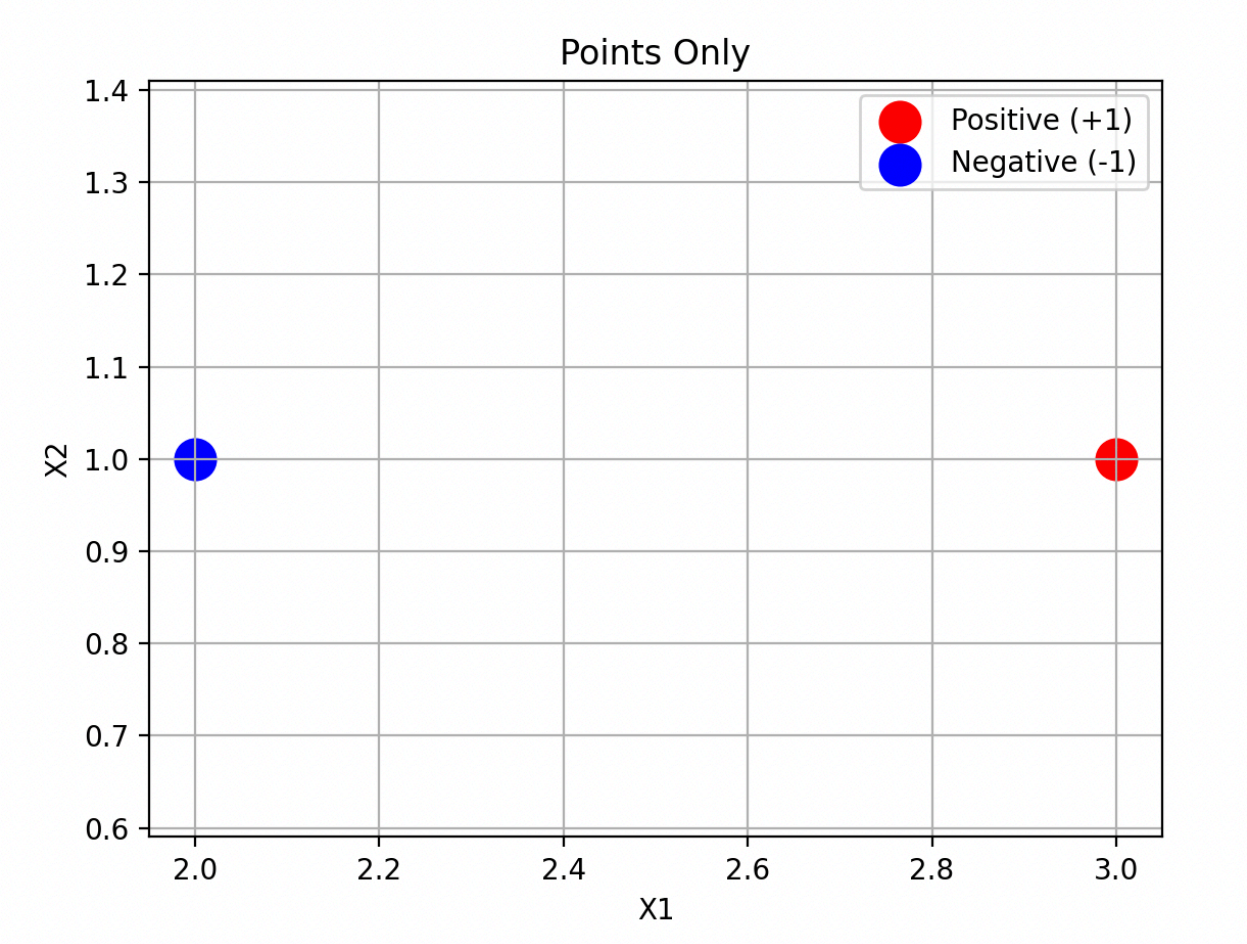
$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \end{bmatrix} \cdot \begin{bmatrix} 20 & 26 & -6 & -12 \\ 26 & 34 & -8 & -16 \\ -6 & -8 & 2 & 4 \\ -12 & -16 & 4 & 8 \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix}$$

***Constraint $y^{T}\alpha \; = \; 0$*:**

$$y^{T}\alpha = \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = a_1 + a_2 - a_3 - a_4 = \; 0$$

Thus: $a_4 = a_1 + a_2 - a_3$

$$\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\lambda}) = 1^{\top}\alpha \; -\frac{1}{2}\alpha^{\top}Q\alpha = \lambda(a_1 + a_2 - a_3 - a_4) = 0$$
$$\Rightarrow 1^{\top}\alpha \; -\frac{1}{2}\alpha^{\top}Q\alpha - \lambda(a_1 + a_2 - a_3 - a_4)$$



Consider a linear SVM with the following training data

Red Positive (+1):
• Point [3, 1]

Blue Negative (-1):
• Point [2, 1]

Task: Solve the SVM dual optimization problem to find the optimal Lagrange multipliers α_i:

$$\max_{\alpha_i \geq 0} [\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j]$$

subject to:

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

Solution:

Red Positive (+1): Point [3, 1] **Blue Negative (-1):** Point [2, 1]

Use matrix form for optimization:

$$\max_{\alpha} 1^{\top}\alpha \; -\frac{1}{2}\alpha^{\top}Q\alpha$$

subject to:

$$y^{T}\alpha \; = \; 0$$

$$\alpha \; \geq \; 0$$

$$X = \begin{bmatrix} 3 & 1 \\ 2 & 1 \end{bmatrix}; \;\; Y = \begin{bmatrix} 1 \\ -1 \end{bmatrix} ; \; YY^{\top} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} [1 \quad -1] =$$

$$yy^{\top} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \; ; \;\; X^{\top}X \; = \begin{bmatrix} 10 & 7 \\ 7 & 5 \end{bmatrix}$$

Note: Calculating $X^{T}X$

$x_1^{\top}x_1 = \; [3 \quad 1] \cdot \begin{bmatrix} 3 \\ 1 \end{bmatrix} = \; 9 + 1 = 10$

$x_1^{\top}x_2 = [3 \quad 1] \cdot \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \; 6 + 1 = 7$ etc.

$$[1 \quad 1] \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} - \frac{1}{2}[\alpha_1 \quad \alpha_2] \cdot \begin{bmatrix} 10 & -7 \\ -7 & 5 \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}$$

***Constraint $y^{T}\alpha \; = \; 0$*:**

$$y^{T}\alpha = [1 \quad -1] \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = a_1 - a_2 = \; 0$$

Thus: $a_1 = a_2$

$$\frac{1}{2}\alpha^{\top}Q\alpha = \frac{1}{2}\begin{bmatrix} 10\alpha_1 & -7\alpha_1 \\ -7\alpha_2 & 5\alpha_2 \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}$$

$$= \frac{1}{2}(10\alpha_1^2 - 7\alpha_1\alpha_2 - 7\alpha_1\alpha_2 + 5\alpha_2^2)$$

$$= 5\alpha_1^2 - 7\alpha_1\alpha_2 + \frac{5}{2}\alpha_2^2$$

$$\mathcal{L}(\boldsymbol{\alpha_1}, \boldsymbol{\alpha_2}, \boldsymbol{\lambda}) = 1^{\top}\alpha \; -\frac{1}{2}\alpha^{\top}Q\alpha = \lambda(a_1 - a_2) = 0$$

$$\Rightarrow 1^{\top}\alpha \; -\frac{1}{2}\alpha^{\top}Q\alpha - \lambda(a_1 - a_2)$$

$$\Rightarrow \alpha_1 + \alpha_2 \; - 5\alpha_1^2 + 7\alpha_1\alpha_2 - \frac{5}{2}\alpha_2^2 - \lambda(a_1 - a_2)$$

Use equality: $a_1 = a_2$

$$\Rightarrow \alpha_1 + \alpha_1 \; - 5\alpha_1^2 + 7\alpha_1^2 - \frac{5}{2}\alpha_1^2 - \lambda(a_1 - \alpha_1)$$

$$= 2\alpha_1 \; - \frac{1}{2}\alpha_1^2$$

$$\frac{\partial}{\partial \alpha_1} = 2 - \alpha_1$$

Set to zero:

$$\frac{\partial}{\partial \alpha_1} = 2 - \alpha_1 = 0$$

$$2 = \alpha_1 = \alpha_2$$

Finding w* and b*: $X = \begin{bmatrix} 3 & 1 \\ 2 & 1 \end{bmatrix}$; $Y = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

$$w^* = \sum_{i=1}^{N} a_i^* y_i x_i = 2 \cdot 1 \cdot [3 \quad 1] + 2 \cdot -1 \cdot [2 \quad 1]$$

$$= 2 \cdot [3 \quad 1] - 2[2 \quad 1]$$

$$= [6 \quad 2] - [4 \quad 2]$$

$$w^* = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

$$\boldsymbol{b^* = y_i - w^{*\top} x_i = 1 - [2 \quad 0] \begin{bmatrix} 3 \\ 1 \end{bmatrix}}$$

$$= 1 - 6 = -5$$

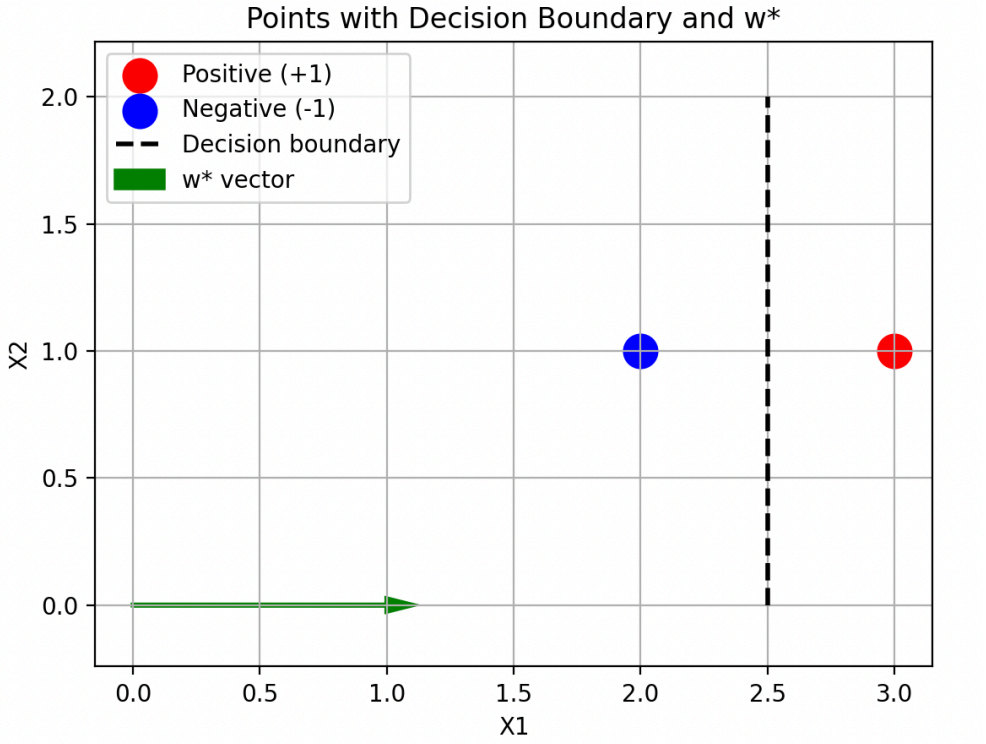
$$w^\top x + b = 0$$

$$[2 \quad 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - 5 = 0$$

$$2x_1 - 5 = 0$$

$$x_1 = 2.5$$

Note: In cases where boundary is not vertical : $w_1 x_1 + w_2 x_2 + b = 0$



Deriving non-linear:

$$w^* = \sum_{i=1}^{N} a_i^* y_i x_i$$

$$y = \left(\sum_{i=1}^{N} a_i^* y_i \vec{x}_i \right)^{\top} \vec{x} + b$$

$$= \sum_{i=1}^{N} a_i^* y_i \vec{x}_i^{\top} \vec{x} + b$$

Results in scalar (prediction)

Kernel function: $h^{\top}(\vec{x}_i)h(x)$

$$y = \sum_{i=1}^{N} a_i^* y_i h^{\top}(\vec{x}_i)h(x) + b$$

$$= \sum_{i=1}^{N} a_i^* y_i \phi(x_i, x) + b$$

For any support vector: $\{x_i, y_i\}$:

$$\boldsymbol{y_i}(\boldsymbol{w}^{*\top}\boldsymbol{x} + \boldsymbol{b}^*) = \mathbf{1}$$

$$\boldsymbol{b}^* = \boldsymbol{y_i} - \boldsymbol{w}^{*\top}\boldsymbol{x_i}$$

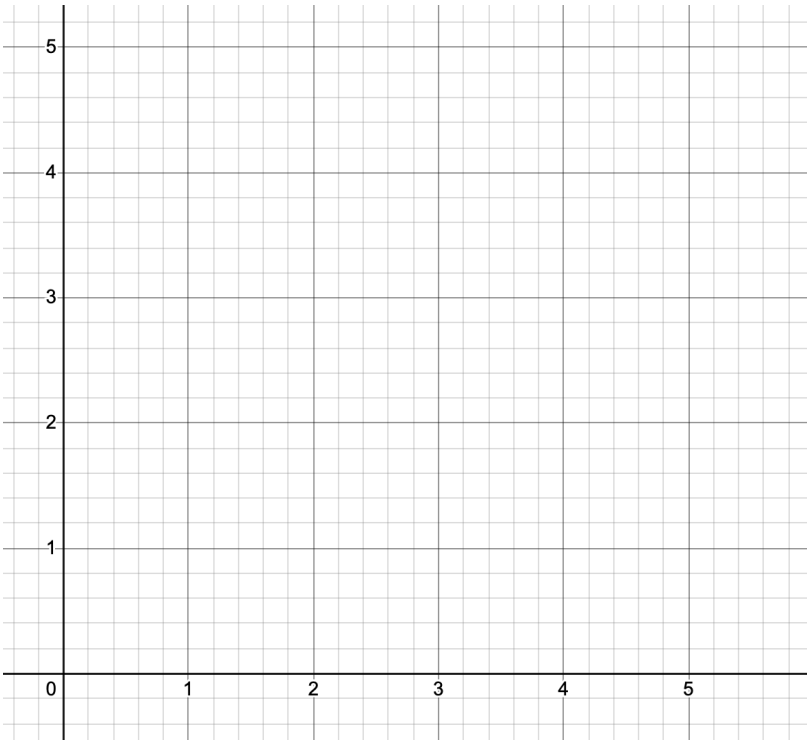
$$\boldsymbol{y} = \boldsymbol{w}^{*\top}\boldsymbol{x} + \boldsymbol{b}^*$$

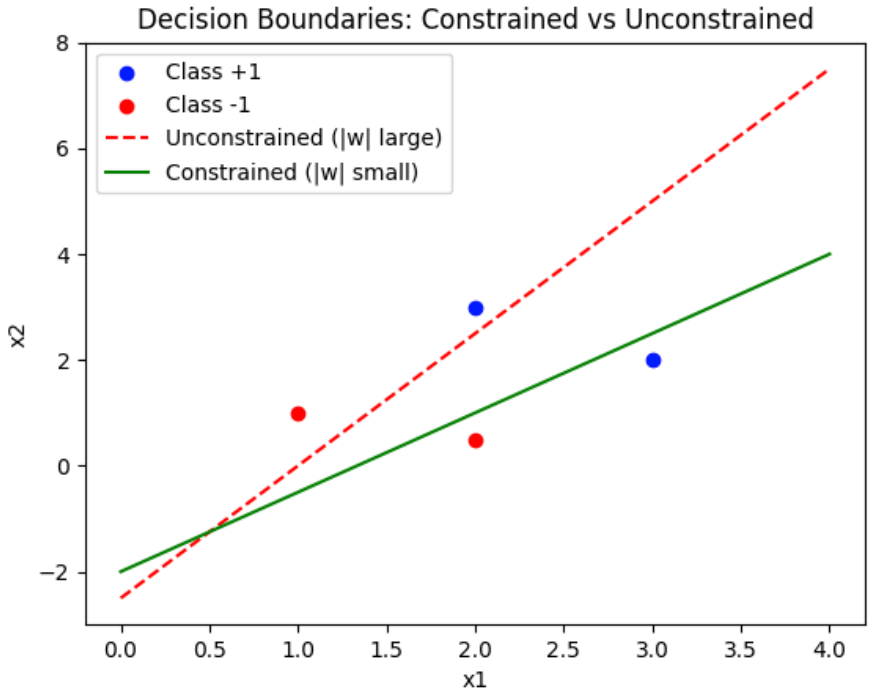
Plot these points below
pos = np.array([[2, 3], [3, 2]])

neg = np.array([[1, 1], [2, 0.5]])

How would constraining the decision boundary between them to $||w||^2 \leq 1$ affect the model *w*?

Consider an unconstrained boundary vector = (5, -2) which does effectively divide the classes, does it violate the constraint?





Consider an unconstrained boundary vector = [5, -2]; $5x_1 - 2x_2 + b = 0$

$$||w||^2 = \left(\sqrt{5^2 + (-2)^2}\right)^2$$

$$= 5^2 + (-2)^2$$

$$= 25 + 4 = 29 > 1$$

This violates the constraint $||w||^2 < 1$

That gives a relatively steep slope (since the ratio $-w_1/w_2 = 2.5$).

Now consider : (0.6, -0.4)

$$||w||^2 = \left(\sqrt{0.6^2 + (-0.4)^2}\right)^2$$

$= 0.36 + 0.16 = 0.52 < 1$

This has a similar direction but a less steep slope: $-\frac{w_1}{w_2} = \frac{-0.6}{-0.4} = 1.5$

Given the following m**atrix** X, find the SVD given that $X = U\Sigma V^{\top}$

$$X = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

Given the following m**atrix** X, find the SVD given that $X = U\Sigma V^\top$

$$X = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

Solution

$$\text{SVD is: } X = U\Sigma V^\top$$

Thus:

$$X^\top X = (V\Sigma U^\top)(U\Sigma V^\top) = V\Sigma^2 V^\top$$

Step 1: Compute $X^\top X$

$$X^\top X = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}^\top \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} \rightarrow A - \lambda I = \begin{bmatrix} 4-\lambda & 0 \\ 0 & 1-\lambda \end{bmatrix}$$

$$(4-\lambda)(1-\lambda) = 0$$

Solving for lambda, we get the following Eigenvalues: $\lambda_1 = 4,\ \lambda_2 = 1$
These eigen values are the squares of the individual diagonal elements of the matrix **Σ:**

$$\begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_m \end{bmatrix}$$

All sigma values below the rank "r" are zeros:

$$\begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Calculate Σ Singular values:
If: $\lambda_1 = 4,\ \lambda_2 = 1$, then:

$$\sigma_1 = \sqrt{4} = 2, \qquad \sigma_2 = \sqrt{1} = 1$$

Step 2: Compute V (Right Singular Vectors) where

$$\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots) \; such \; that \; \sigma_1 \geq \sigma_2 \geq \cdots \geq 0.$$

Eigenvectors of $X^\top X$:
Considering $\lambda_1 = 4$:

$$A - \lambda I = \begin{bmatrix} 4-4 & 0 \\ 0 & 1-4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ -3y \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$\lambda_2 = 1$:

$$A - \lambda I = \begin{bmatrix} 4-1 & 0 \\ 0 & 1-1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} -3x \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$v_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, v_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

So:

$$V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Step 3: Compute U (Left Singular Vectors)

$$u_i = \frac{1}{\sigma_i} X v_i$$

sigma values: $\sigma_1 = 2$; $\sigma_2 = 1$

$$u_1 = \frac{1}{\sigma_1} X v_1 = \frac{1}{2} \cdot \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{2} \cdot \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$u_2 = \frac{1}{\sigma_2} X v_2 = \frac{1}{1} \cdot \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{1} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Thus:

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Step 4: Σ (Diagonal Matrix of Singular Values)

$$\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

Step 5: Verify Reconstruction $X = U\Sigma V^\top$

$$X = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \; \Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} ; \; V^\top = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$U\Sigma V^\top = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^\top = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} = \text{X}$$

SVD Example 2: How to decompose matrix A, $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$, by just finding its Left singular values (S_L) and right singular values (S_R) ? (find the eigenvalues for each using numerical methods)

SVD Example 2: How to decompose matrix A, $\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$, by just finding its Left singular values (S_L) and right singular values (S_R) ? (find the eigenvalues for each using numerical methods)

Solution:

Left singular values (S_L) : $AA^\top$

$$AA^\top = \begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} = \begin{bmatrix}14 & 32\\ 32 & 77\end{bmatrix}$$

Eigenvalues: [0.59732747 90.40267253]

Right singular values (S_R) : $A^\top A$

$$A^\top A = \begin{bmatrix}1 & 4\\ 2 & 5\\ 3 & 6\end{bmatrix}\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} = \begin{bmatrix}17 & 22 & 27\\ 22 & 29 & 36\\ 27 & 36 & 45\end{bmatrix}$$

Eigenvalues: [90.40267253 , 0.59732747 , -4.88052448e-15]

Note that both resulting matrices are symmetric. Sort the eigen values for both in descending order:

	$\boldsymbol{\lambda_1}$	$\boldsymbol{\lambda_2}$	$\boldsymbol{\lambda_3}$
(S_L) : $AA^\top$	90.40267253	0.59732747	
(S_R) : $A^\top A$	90.40267253	0.59732747	-4.88052448e-15

$$\sigma_1 = \sqrt{\lambda_1} \quad ; \quad \sigma_2 = \sqrt{\lambda_2}$$

Note that the first two for both are equal.

Let $\vec{u}_1, \vec{u}_2$ ... be the eigenvectors of (S_L) : $AA^\top$

Let $\vec{v}_1, \vec{v}_2$ ... be the eigenvectors of (S_R) : $A^\top A$

$$A = U\Sigma V^\top$$

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} = \begin{bmatrix}| & |\\ \vec{u}_1 & \vec{u}_2\\ | & |\end{bmatrix}\begin{bmatrix}\sigma_1 & 0 & 0\\ 0 & \sigma_2 & 0\end{bmatrix}\begin{bmatrix}- & \vec{v}_1^\top & -\\ - & \vec{v}_2^\top & -\\ - & \vec{v}_3^\top & -\end{bmatrix}$$

Considering SVD ($\boldsymbol{X = U\Sigma V^{\top}}$), are there situations when this decomposition cannot occur?
No any matrix X can be unconditionally decomposed such that $\boldsymbol{X = U\Sigma V^{\top}}$

When is SVD not appropriate?
SVD is not efficient for large sparse matrices, not suitable for partially observed matrices

What is the advantage of the ReLU activation function
The advantage of the ReLU activation function is that it normally leads to much larger gradients than other activation functions.

Why do we need to specify the structure of an ANN?
we have not found any effective machine learning methods that can automatically learn a network structure from data. When we use ANNs, we have to first predetermine the network structure based on the nature of the data, as well as our domain knowledge.

How does sign classification in Linear Regression work?

When applying the model onto a data point ($w^\top x$), if it is greater then 0 assign y = +1, otherwise: y=-1.

$$y = \text{sign}(\mathbf{w}^{*\top}\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{w}^{*\top}\mathbf{x} > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Question: why does this decision rule make sense?
$w^\top x$ tells you on which side of the hyperplane defined by w the point x lies.

Show the procedure to vectorize the square error objective function in linear regression methods:

$$\sum_{i=1}^{N} (w^\top x_i - y_i)^2 = \left|\left|Xw - y\right|\right|^2$$

Solution:

$$X = \begin{bmatrix} x_1^\top \\ x_2^\top \\ \vdots \\ x_N^\top \end{bmatrix}_{N\times d} \longrightarrow y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}_{N\times 1}$$

The **least square error**: *we want* $\mathbf{w}^\top x_i$ *to be the smallest possible*

$$E(\mathbf{w}) = \sum_{i=1}^{N} (\mathbf{w}^\top x_i - y_i)^2 = \| X\mathbf{w} - y \|^2$$

Result is a column vector:

$$X\mathbf{w} - y = \begin{bmatrix} x_1^\top \\ x_2^\top \\ \vdots \\ x_N^\top \end{bmatrix} [W] - \begin{bmatrix} y_1^\top \\ y_2^\top \\ \vdots \\ y_N^\top \end{bmatrix}$$

$$\begin{bmatrix} x_1^\top w \\ x_2^\top w \\ \vdots \\ x_N^\top w \end{bmatrix} - \begin{bmatrix} y_1^\top \\ y_2^\top \\ \vdots \\ y_N^\top \end{bmatrix} = \begin{bmatrix} x_1^\top w - \mathrm{y}_1 \\ x_2^\top w - y_2 \\ \vdots \\ x_N^\top w - y_N \end{bmatrix}$$

$$\begin{aligned} \mathrm{E(w)} = \| X\mathbf{w} - y \|^2 &= (X\mathrm{w} - y)^\top (X\mathrm{w} - y) \\ &= ((\mathrm{Xw})^\top - y^T)(X\mathrm{w} - y) \\ &= (\mathrm{w}^\top X^T - y^T)(X\mathrm{w} - y) \\ &= \mathrm{w}^\top X^T X w - y^\top X w - \mathrm{w}^\top X^T y + y^\top y \\ &= \mathrm{w}^\top X^T X w - \left(y^\top (Xw)\right)^\top - \mathrm{w}^\top X^T y + y^\top y \\ &= \mathrm{w}^\top X^T X w - (Xw)^\top y^{\top\top} - \mathrm{w}^\top X^T y + y^\top y \end{aligned}$$

apply ⊤ property to second term:

$$\begin{aligned} &= \mathrm{w}^\top X^T X w - \mathrm{w}^\top X^T y - \mathrm{w}^\top X^T y + y^\top y \\ &= w^T X^T X w - 2 w^T X^T y + y^T y \end{aligned}$$

How to find the gradient of the square error objective function in linear regression?

Using this we then find the gradient of E(w) w.r.t. w:

$$\frac{\partial E(w)}{\partial w} = 2X^{\top}XW - 2X^{\top}y + 0$$

Doing this ($w^TX^TXw - 2w^TX^Ty + y^Ty$), term by term wrt w:

Consider X^TX symmetric, thus:

$$w^TX^TXw \rightarrow w^TAw = 2Aw \rightarrow 2X^TXw$$
$$-2w^TX^Ty \rightarrow -2X^Ty$$
$$y^Ty \rightarrow 0$$

Thus:

$$\frac{\partial E(w)}{\partial w} = 2X^{\top}XW - 2X^{\top}y + 0$$

Setting the gradient equal to 0:

$$\frac{\partial E(w)}{\partial w} = 0 \rightarrow 2X^{\top}XW - 2X^{\top}y = 0$$
$$2X^{\top}XW = 2X^{\top}y$$
$$X^{\top}XW = X^{\top}y$$
$$(X^{\top}X)^{-1}X^{\top}XW = (X^{\top}X)^{-1}X^{\top}y$$
$$W^* = (X^{\top}X)^{-1}X^{\top}y$$

Find the distance from a Point to a Hyperplane given:

- Weight vector: $\mathbf{w} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$
- Bias: $b = -10$
- Point: $\mathbf{x}_0 = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$

Solution:

Step 1: Compute $w^\top x_0$

$$\mathbf{w}^\top \mathbf{x}_0 = 3 \cdot 2 + 4 \cdot 3 = 6 + 12 = 18$$

Step 2: Add bias

$$\mathbf{w}^\top \mathbf{x}_0 + b = 18 + (-10) = 8$$

Step 3: Take absolute value

$$| \mathbf{w}^\top \mathbf{x}_0 + b | = | 8 | = 8$$

Step 4: Compute norm of w

$$\| \mathbf{w} \| = \sqrt{3^2 + 4^2} = \sqrt{9 + 16} = \sqrt{25} = 5$$

Step 5: Compute distance from point to hyperplane

$$\text{Distance} = \frac{| \mathbf{w}^\top \mathbf{x}_0 + b |}{\| \mathbf{w} \|} = \frac{8}{5} = 1.6$$

Concerning the formation of SVMO

max$_{\varepsilon,w,b}\ \gamma$

$$\frac{y_i(w^T x_i + b)}{||w||} \geq \gamma, \forall_i \in \{1, 2, \ldots, N\}$$

Why is there this inequality?

Because the distance from any point to the hyperplane will be at least gamma (the hyperplane being 2 times gamma, the exact midpoint from the support vectors for each class).

When solving for the maximum-margin hyperplane in a linear SVM, explain why the optimization problem SVM0 can be equivalently reformulated as the problem SVM1.

In the first one the distance is gamma, but now we just rescale to be +1 or -1:

$$\gamma = \frac{|w^T x + b|}{||w||} = \frac{|1|}{||w||}$$

Maximizing 2γ is equivalent to minimizing $\frac{1}{2}||w||^2 = \frac{1}{2} w^t w$

What is the Kernel Trick?

Calculating high-dimensional relationships without transforming the data into the higher dimension

Derive the closed-form solution for the ridge regression method?

$$\sum_{i=1}^{N}(w^T x_i - y_i)^2 + \lambda \cdot ||w||_2^2$$

Solution:

$$\sum_{i=1}^{N}(w^T x_i - y_i)^2 + \lambda \cdot ||w||_2^2$$

$$L_{ridge}(w) = ||Xw - y||^2 + \lambda w^\top w$$
$$= (Xw - y)^\top (Xw - y) + \lambda w^\top w$$
$$= (Xw - y)^\top (Xw - y) + \lambda w^\top w$$
$$= w^\top X^\top Xw - 2Xwy^\top + y^\top y + \lambda w^\top w$$
$$\nabla L_{ridge}(w) = 2X^\top Xw - 2\mathrm{X^T y} + 2\lambda w$$

Set gradient to 0:

$$2X^\top Xw - 2X^\top y + 2\lambda w = 0$$
$$X^\top Xw - X^\top y + \lambda w = 0$$
$$(X^\top X + \lambda \mathrm{I})w = X^\top y$$
$$w^*_{ridge} = (\mathrm{X^T X} + \lambda \mathrm{I})^{-1}\mathrm{X^T y}$$

What is the general idea behind a lambda value as a regularization?

Penalizing a particular value

For a real number $p \geq 1$, how is the ℓ_p-norm of a vector $x \in \mathbb{R}^n$ defined?

Solution:

For a real number $p \geq 1$, :

$$\| x \|_p = \left(\sum_{i=1}^{n} |w_n|^p \right)^{\frac{1}{p}}$$

Given the vector $x = (5,2,-3)$, find the L1, L2 and L-infinity norm of x:

Solution:

ℓ_1-norm (Manhattan norm):

$$\| x \|_1 = | 5 | + | 2 | + | -3 | = 5 + 2 + 3 = \mathbf{10}$$

ℓ_2-norm (Euclidean norm):

$$\| x \|_2 = \sqrt{5^2 + 2^2 + (-3)^2} = \sqrt{25 + 4 + 9} = \sqrt{38} \approx \mathbf{6.16}$$

$\ell\infty$-norm (Maximum norm):

$$\| x \|_\infty = \max(| 5 |, | 2 |, | -3 |) = \max(5,2,3) = \mathbf{5}$$

Essential differences btw Ridge and Lasso?
Ridge = shrinkage ; Lasso = shrinkage + sparsity
Ridge regression is useful when most (or all) variables are useful, whereas LASSO is useful when we want to eliminate some features to 0. Lasso is suitable for feature selection and interpretation. Ridge regression *does* have a closed form solution whereas LASSO does not (must be done with gradient descent).

Explain why $\mathcal{L}1$ norm regularization lead to sparse solutions.
With LASSO (L1), some parameters can be shrunk all the way to zero, which Ridge (L2) cannot do

Given the SVD methods available in linear algebra to factorize matrices, why do we still need to use machine learning methods in collaborative filtering?

1. The computational complexity would be extremely high (Alternating Algorithm, SGD can be parallelizable)
2. SVD requires a complete matrix (in reality often not available)
3. SVD has no built-in regularization, thus for sparse matrices overfitting would be prevalent

Ridge Regression Example:

You have two training points:

$$(x_1, y_1) = (1,2) \ ; \ (x_2, y_2) = (2,2)$$

We fit the linear model $\hat{y} = w\,x$ (no intercept) with ridge penalty $\lambda > 0$

1. Write the closed-form ridge solution for w.
2. Compute w for $\lambda = 0$ (OLS) and for $\lambda = 1$
3. For each case compute the predictions, residuals and training sum of squared errors (SSE). Interpret.

Solution:

There are only 2 data points so N = 2:

$$E(w) = \sum_{i=1}^{2} (w^T x_i - y_i)^2 + \lambda \cdot w^2$$

$$E(w) = (w * 1 - 2)^2 + (w * 2 - 2)^2 + \lambda \cdot w^2$$

1. Finding the closed-form ridge solution:

$$\frac{\partial E(w)}{\partial w} = \frac{\partial}{\partial w}\left(\sum_{i=1}^{N} (w^T x_i - y_i)^2 + \lambda \cdot w^2\right)$$

$$= \sum_{i=1}^{N} 2(w^T x_i - y_i)x_i + 2\lambda w$$

Setting gradient to 0 (finding minimum):

$$0 = \sum_{i=1}^{N} 2(w^T x_i - y_i)x_i + 2\lambda w$$

$$0 = \sum_{i=1}^{N} (w^T x_i - y_i)x_i + \lambda w$$

$$0 = w\sum_{i=1}^{N} x_i^2 - \sum_{i=1}^{N} y_i x_i + \lambda w$$

$$0 = w\left(\left(\sum_{i=1}^{N} x_i^2\right) + \lambda\right) - \sum_{i=1}^{N} y_i x_i$$

$$w\left(\left(\sum_{i=1}^{N} x_i^2\right) + \lambda\right) = \sum_{i=1}^{N} y_i x_i$$

$$w^* = \frac{\sum_{i=1}^{N} y_i x_i}{(\sum_{i=1}^{N} x_i^2) + \lambda}$$

Solving with Data points: $(x_1, y_1) = (1,2)$; $(x_2, y_2) = (2,2)$

$$\sum_{i=1}^{N} y_i x_i = 1 \cdot 2 + 2 \cdot 2 = 6 \qquad and \qquad \sum_{i=1}^{N} x_i^2 = 1^2 + 2^2 = 5$$

$$w^* = \frac{\sum_{i=1}^{N} y_i x_i}{(\sum_{i=1}^{N} x_i^2) + \lambda}$$

$$= \frac{6}{5 + \lambda}$$

2: Compute w for $\lambda = 0$ and for $\lambda = 1$

$$\frac{6}{5 + 0} = \frac{6}{5} \qquad and \qquad \frac{6}{5 + 1} = \frac{6}{6} = 1$$

3. For each case compute the predictions, residuals and training sum of Ridge Loss:

$\lambda = 0$:

$$E(w) = \left(\frac{6}{5} * 1 - 2\right)^2 + \left(\frac{6}{5} * 2 - 2\right)^2 + 0 \cdot \frac{6}{5}^2$$

$$= \left(-\frac{4}{5}\right)^2 + \left(\frac{2}{5}\right)^2$$

$$= \frac{16}{25} + \frac{4}{25} = \frac{20}{25} = 0.8$$

$\lambda = 1$:

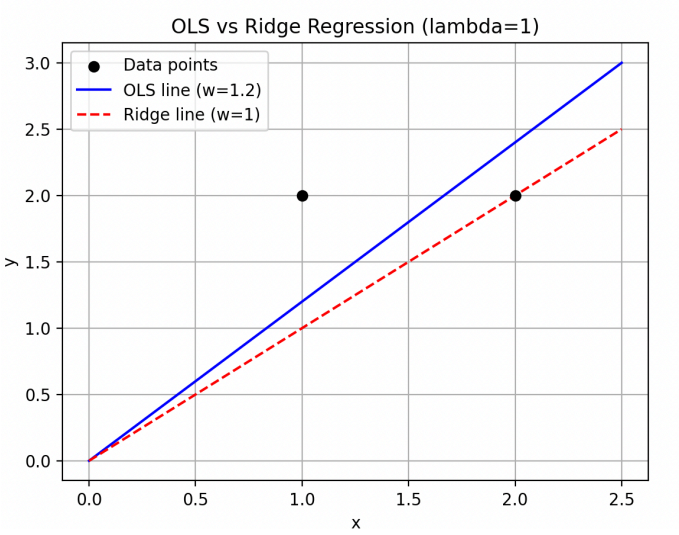
$$E(w) = (1 * 1 - 2)^2 + (1 * 2 - 2)^2 + 1 \cdot 1^2$$

$$= (-1)^2 + (0)^2 + 1$$

$$= 1 + 1 = 2$$

when lambda is 1 how is that different from lambda = 0?

The ridge line becomes less steep, the larger lambda is the less sensitive it is to the data points during training



Calculate the Softmax Output

Compute $\mathbf{y} = \text{softmax}(\mathbf{x})$ for the following input vectors:

- $\mathbf{x}_1 = [-2,\ 0,\ 1]$
- $\mathbf{x}_2 = [10,\ 4,\ -5]$

Solution:

$$using \quad y_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}:$$

$\mathbf{x}_1 = [-2,\ 0,\ 1]$

$$y_1 = \frac{e^{-2}}{e^{-2} + e^{0} + e^{1}} \quad ; \quad y_2 = \frac{e^{0}}{e^{-2} + e^{0} + e^{1}} \quad ; \quad y_3 = \frac{e^{1}}{e^{-2} + e^{0} + e^{1}}$$

$\mathbf{x}_2 = [10,\ 4,\ -5]$

$$y_1 = \frac{e^{10}}{e^{10} + e^{4} + e^{-5}} \quad ; \quad y_2 = \frac{e^{4}}{e^{10} + e^{4} + e^{-5}} \quad ; \ y_3 = \frac{e^{-5}}{e^{10} + e^{4} + e^{-5}}$$

If Stride is 1 and no padding, what is the output length of a vector (length n) and convolutional kernel (length k)?

Answer:

n-k+1

Assume we have an input vector:

$$\mathbf{x} = [\mathbf{2},\ \mathbf{2},\ \mathbf{3},\ \mathbf{1},\ \mathbf{31},\ \mathbf{33},\ \mathbf{29},\ \mathbf{36}]$$

If we apply a convolution kernel:

$$\mathbf{w_1} = [\mathbf{1},\ -\mathbf{1},\ \mathbf{1}]$$

show the procedure how to calculate the output from the convolutional sum: $\mathbf{x} * \mathbf{w1}$

Consider another kernel $w_2 = [1 \quad 1]$, repeat the above and show that $\mathbf{x} * \mathbf{w2}$

Solution:

w1: 1*2 + (-1)*2 + 1*3 = 2 - 2 + 3 = 3 1*2 + (-1)*3 + 1*1 = 2 - 3 + 1 = 0 1*3 + (-1)*1 + 1*31 = 3 - 1 + 31 = 33 1*1 + (-1)*31 + 1*33 = 1 - 31 + 33 = 3 1*31 + (-1)*33 + 1*29 = 31 - 33 + 29 = 27 1*33 + (-1)*29 + 1*36 = 33 - 29 + 36 = 40 x * w1 = [3, 0, 33, 3, 27, 40]	w2: 1*2 + 1*2 =4 1*2 + 1*3=5 1*3 + 1*1=4 1*1 + 1*31=32 1*31 + 1*33=64 1*33 + 1*29=62 1*29 + 1*36=65 x * w2 = [4, 5, 4, 32, 64, 62, 65]

Why are w1 and w2 called local feature extractor?
It looks for groups of high values which are grouped together, highlights changes in features (such as edges)

Comparing w1 with w2, explain why w1 is sometimes called an edge detector ?
w_1 = [1, -1, 1]: It emphasizes differences between neighboring values: Positive * Negative → highlights where the signal changes direction or jumps
w_2 = [1, 1]: Just sums neighboring values → highlights high intensity regions

Other Examples:
Edge/Gradient detectors: w = [1, -1] in 1D → detects upward/downward jumps
Smoothing: w = [1/3, 1/3, 1/3] → averages local region → removes noise, smooths signal
Sharpening / Highlight details: w = [-1, 2, -1] → emphasizes differences → makes peaks more visible
Pattern / Texture detection: w = [1, 0, -1, 0, 1] → detects repeating patterns
emphasizes rising slope : w = [1, 2, 1]
local maxima: w = [-1, 2, -1]

Explain how zero padding will affect the convolution output.
Example, padding the ends with zeros:

$$\boldsymbol{x' = [0, 2, 2, 3, 1, 31, 33, 29, 36, 0]}$$

And applying w1:$\mathbf{w_1 = [1, -1, 1]}$ What is the effect?
w1:
1*0 + (-1)*2 + 1*2 = 0 - 2 + 2 = 0
1*2 + (-1)*2 + 1*3 = 2 - 2 + 3 = 3
1*2 + (-1)*3 + 1*1 = 2 - 3 + 1 = 0
1*3 + (-1)*1 + 1*31 = 3 - 1 + 31 = 33
1*1 + (-1)*31 + 1*33 = 1 - 31 + 33 = 3
1*31 + (-1)*33 + 1*29 = 31 - 33 + 29 = 27
1*33 + (-1)*29 + 1*36 = 33 - 29 + 36 = 40
1*29 + (-1)*36 + 1*0 = 29 - 36 +0 = -7

X' * w1 = [0, 3, 0, 33, 3, 27, 40,-7]

The edge boundaries come up with strange values on the edges by introducing the 0s. Useful when you want to keep the output the same side and always include edge data (important info at the boundaries), lets the edge contribute.

Explain how using stride s= 2 will affect the convolution output., what is the output length of a vector (length n) and convolutional kernel (length k) when the stride is 2?

Answer:
Instead of n-k+1, it is $\left\lfloor \frac{n-k}{s} \right\rfloor + 1$ // we need the floor to avoid partial numbers

Computing Y1: (stride =1) for X= 1 2 3 4 5 and W = -1 2

Solution:
Computing Y1: (*stride =1*)

X=	**1**	**2**	3	4	5

W =	**-1**	**2**

$$1 \times (-1) + 2 \times 2 = 3$$

Y=	3				

Computing Y2:

X=	1	**2**	**3**	4	5

W =	**-1**	**2**

$$2 \times (-1) + 3 \times 2 = 4$$

Y=	3	4			

Computing Y3:

X=	1	2	**3**	**4**	5

W =	**-1**	**2**

$$3 \times (-1) + 4 \times 2 = 5$$

Y=	3	4	5		

Computing Y4:

X=	1	2	3	**4**	**5**

W =	**-1**	**2**

$$4 \times (-1) + 5 \times 2 = 6$$

Y=	3	4	5	6

Output neurons: n = d - f + 1
D = 5, f = 2 thus : n = 5 - 2 + 1 = 4

Consider a One-Hidden-Layer Neural Network with Softmax Output:

Input:

$$\mathbf{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Hidden Layer:

$$\mathbf{W}^{(1)} = \begin{bmatrix} 1 & 2 \\ 0 & -1 \\ 2 & 1 \end{bmatrix}, \quad \mathbf{b}^{(1)} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

Activation Function: ReLU

Output Layer:

$$\mathbf{W}^{(2)} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}, \qquad \mathbf{b}^{(2)} = [0 \quad 0]$$

Final Output: Softmax Output

Show the procedure to compute the output for the above setting.

Solution:

$$using \quad z = \phi(w^{\top}x + b):$$

$$z = \begin{bmatrix} 1 & 2 \\ 0 & -1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \cdot 1 + 2 \cdot 1 \\ 0 \cdot 1 - 1 \cdot 1 \\ 2 \cdot 1 + 1 \cdot 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

$$= \begin{bmatrix} 3 \\ -1 \\ 3 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

$$= \begin{bmatrix} 4 \\ -1 \\ 2 \end{bmatrix}$$

ReLU: $\boldsymbol{z = max(0, x)} \rightarrow \begin{bmatrix} 4 \\ 0 \\ 2 \end{bmatrix}$

$$z = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 0 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 4 \cdot 1 + 0 \cdot 0 - 1 \cdot 2 \\ 2 \cdot 4 + 1 \cdot 0 + 0 \cdot 2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 2 \\ 8 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 8 \end{bmatrix}$$

$$using \quad y_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}:$$

$$y_1 = \frac{e^2}{e^2 + e^8} \quad ; \quad y_2 = \frac{e^8}{e^2 + e^8}$$

MCE - Given:

- **x = [2, 1]**
- **w = [1, 1]**
- **True label: y = +1**

Is this correct or incorrect?

Solution

Compute:

$$w^T x = 1 \cdot 2 + 1 \cdot 1 = 3$$
$$-y(w^T x) = (+1)(3) = -3$$

Since -y(w^T x) = -3 < 0, this is a correct classification.

$$-y_i \mathbf{w}^\intercal \mathbf{x}_i = \begin{cases} > 0 & \Longrightarrow \text{mis-classification} \\ < 0 & \Longrightarrow \text{correct classification} \end{cases}$$

MCE - Given:

- **x = [2, 1]**
- **w = [-1, -1]**
- **True label: y = +1**

Is this correct or incorrect?

Compute:

$$w^T x = (-1) \cdot 2 + (-1) \cdot 1 = -3$$
$$-y(w^T x) = (+1)(-3) = +3$$

Since -y(w^T x) = 3 > 0, this is an incorrect classification.

$$-y_i \mathbf{w}^\intercal \mathbf{x}_i = \begin{cases} > 0 & \Longrightarrow \text{mis-classification} \\ < 0 & \Longrightarrow \text{correct classification} \end{cases}$$

Concerning the empirical risk (average loss) for logistic regression:

$$Q(w) = -lnL(w) = -\frac{1}{N}\sum_{i=1}^{N} l(y_i w^T x_i) \qquad and \qquad -\ln\left(l\left(y \odot (Xw)\right)\right.$$

Show that both forms are equivalent

(From Oct 28)

$$\frac{\partial Q(w;B)}{\partial B} = \frac{1}{B}\sum_{i=1}^{B} \mathrm{y_i}\left(1 - \sigma(y_i w^T x_i)\right)x_i$$

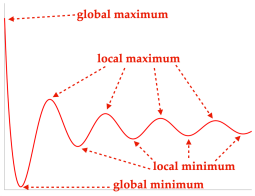
$$= \frac{1}{B} X^\intercal \left[y \odot l\left(y \odot (Xw)\right) - y\right]$$

Show that both forms are equivalent

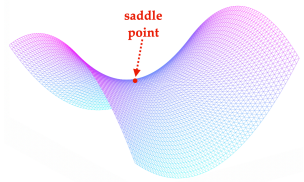
What is the difference between a stationary point, critical point, and saddle point?

stationary point: where the gradient is zero

critical point: a stationary point **OR** an undifferentiable point



saddle point: a stationary point but is Not a local minimum (maximum)



Not a local min, but to decrease means to change directions on a different dimension

Optimization:

What are the three cases of optimization:

- 1.unconstrained optimization: *any x that meets optimality conditions will suffice*
- 2.optimization under *equality* constraints
- 3.general optimization subject to both *inequality* and *equality* constraints

What types of numerical optimization methods are there?

- **zero-order methods**: using the function values alone, such as grid search
- **first-order methods**: using the function values and gradients, such as gradient descent method
- **second-order methods**: using the function values, gradients and Hessians, such as Newton method

extra matrix problem:

Given:

- $\mathbf{K} \in \mathbb{R}^{n \times n}$ (symmetric)
- $\mathbf{z} \in \mathbb{R}^{n}$
- $\boldsymbol{\gamma > 0}$

Solve:

$\mathbf{min}_x \, (1/2)x^\top Kx - z^\top x + (\gamma/2)\|Kx\|^2$

Hint: $\|Kx\|^2 = x^\top K^\top Kx = x^\top K^2 x$ (since K is symmetric)

Numerical Optimization Methods

- **Gradient Descent:** $x^{(n+1)} = x^{(n)} - \eta \nabla f(x^{(n)})$
- **SGD:** $x^{(n+1)} = x^{(n)} - \eta \nabla f_k(x^{(n)})$
- **Newton's Method:** $x^{(n+1)} = x^{(n)} - H^{-1}(x^{(n)}) \nabla f(x^{(n)})$

Problem 3.1: Gradient Descent Implementation

Given the objective function:

$f(x_1, x_2) = x_1^2 + 2x_2^2 - 4x_1 - 8x_2 + 10$ // normally you would minimize the error function

(a) Compute the gradient $\nabla f(x_1, x_2)$

(b) Write the pseudocode for gradient descent to minimize this function

(c) Given initial point $x^{(0)} = (0, 0)$ and learning rate $\eta = 0.1$, perform **two iterations** by hand and show:

- $x^{(1)} = ?$
- $x^{(2)} = ?$

Solution:

$$\nabla f(x_1, x_2) = \begin{bmatrix} 2x_1 - 4 \\ 4x_2 - 8 \end{bmatrix}$$

$x^{(n+1)} = x^{(n)} - \eta \nabla f(x^{(n)})$

$$\begin{aligned} x^{(1)} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 0.1 \begin{bmatrix} 2 \cdot 0 - 4 \\ 4 \cdot 0 - 8 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} -0.1 \\ -0.8 \end{bmatrix} \\ &= \begin{bmatrix} 0.4 \\ 0.8 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} x^{(2)} &= \begin{bmatrix} 0.4 \\ 0.8 \end{bmatrix} - 0.1 \begin{bmatrix} 2 \cdot 0.4 - 4 \\ 4 \cdot 0.8 - 8 \end{bmatrix} \\ &= \begin{bmatrix} 0.4 \\ 0.8 \end{bmatrix} - \begin{bmatrix} -0.32 \\ -0.48 \end{bmatrix} \\ &= \begin{bmatrix} 0.72 \\ 1.28 \end{bmatrix} \end{aligned}$$

Problem 3.2: Stochastic Gradient Descent

Consider a training dataset with N = 4 samples, where the loss function is:

$f(w) = (1/4)\sum_{i=1}^{4} f_i(w)$

with:

- $f_1(w) = (w - 3)^2$
- $f_2(w) = (w - 1)^2$
- $f_3(w) = (w + 2)^2$
- $f_4(w) = (w - 2)^2$

(a) Write the full gradient $\nabla f(w) = (1/4)\sum_{i=1}^{4} \nabla f_i(w)$

(b) Compute each individual gradient: $\nabla f_1(w)$, $\nabla f_2(w)$, $\nabla f_3(w)$, $\nabla f_4(w)$

(c) Write the pseudocode for **stochastic gradient descent (SGD)** that randomly selects one sample at each iteration

(d) Starting at $w^{(0)} = 0$ with $\eta = 0.2$, if sample k=2 is randomly selected, compute $w^{(1)}$

Problem 3.3: Mini-batch SGD

Given the same setup as Problem 3.2 with 4 samples.

(a) Write pseudocode for **mini-batch SGD** with batch size $B = 2$

(b) If the samples are shuffled into batches: $B_1 = \{1, 3\}$ and $B_2 = \{2, 4\}$, perform **one complete epoch** starting from $w^{(0)} = 0$ with $\eta = 0.2$. Show:

- $w^{(1)}$ after processing batch B_1
- $w^{(2)}$ after processing batch B_2