Perceptrons

Classification

Naive Bayes → Perceptrons!

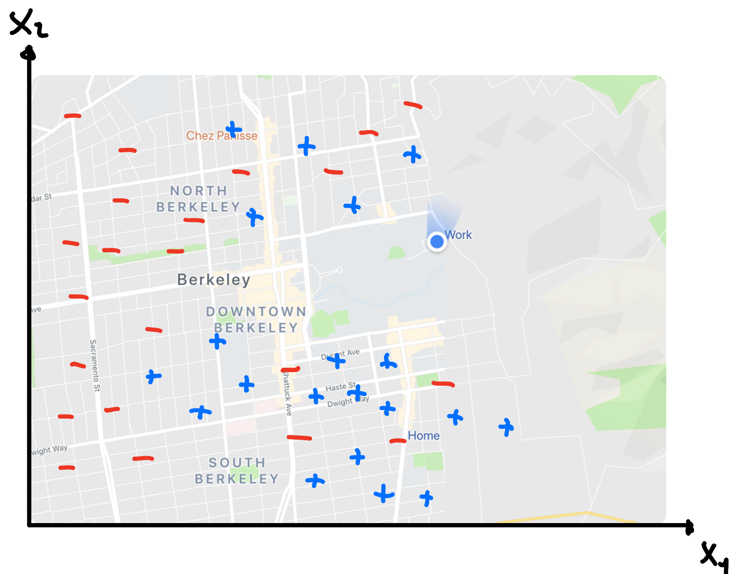
more general

(relies on indep. assumptions)

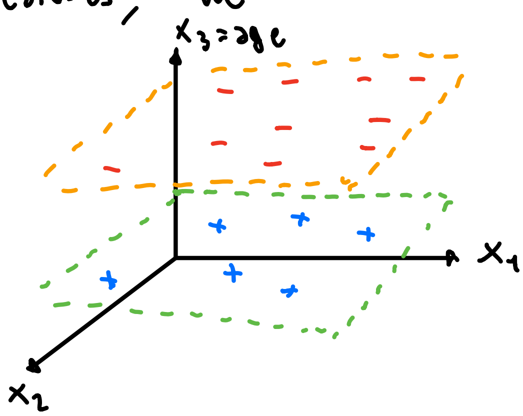
"Mark 1"

We have features → want classification

Assumption: our data is linearly separable



Idea: by adding features, we will be able to separate the data



The Algorithm

The perceptron algorithm works as follows:

1. Initialize all weights to 0: $\mathbf{w} = \mathbf{0}$
2. For each training sample, with features $\mathbf{f}(x)$ and true class label $y^* \in \{-1, +1\}$, do:
   (a) Classify the sample using the current weights, let y be the class predicted by your current $\mathbf{w}$:

$$y = \text{classify}(x) = \begin{cases} +1 & \text{if activation}_w(x) = \mathbf{w}^T\mathbf{f}(x) > 0 \\ -1 & \text{if activation}_w(x) = \mathbf{w}^T\mathbf{f}(x) < 0 \end{cases}$$

   (b) Compare the predicted label y to the true label y^*:
   - If $y = y^*$, do nothing
   - Otherwise, if $y \neq y^*$, then update your weights: $\mathbf{w} \leftarrow \mathbf{w} + y^*\mathbf{f}(x)$
3. If you went through **every** training sample without having to update your weights (all samples predicted correctly), then terminate. Else, repeat step 2

1 Perceptron

You want to predict if movies will be profitable based on their screenplays. You hire two critics A and B to read a script you have and rate it on a scale of 1 to 4. The critics are not perfect; here are five data points including the critics' scores and the performance of the movie:

#	Movie Name	A	B	Profit?
1	Pellet Power	1	1	-
2	Ghosts!	3	2	+
3	Pac is Bac	2	4	+
4	Not a Pizza	3	4	+
5	Endless Maze	2	3	-

1. First, you would like to examine the linear separability of the data. Plot the data on the 2D plane above; label profitable movies with + and non-profitable movies with − and determine if the data are linearly separable.

linearly seperable!

2. Now you decide to use a perceptron to classify your data. Suppose you directly use the scores given above as features, together with a bias feature. That is $f_0 = 1$, f_1 = score given by A and f_2 = score given by B.

   Run one pass through the data with the perceptron algorithm, filling out the table below. Go through the data points in order, e.g. using data point #1 at step 1.

step	Weights	Score	Correct?
1	[-1, 0, 0]	$-1 \cdot 1 + 0 \cdot 1 + 0 \cdot 1 = -1$	yes
2	[-1,0,0]	$-1 \cdot 1 + 0 + 0 = -1$	no
3	[0, 3, 2]	$0 \cdot 1 + 3 \cdot 2 + 2 \cdot 4 = 14$	yes
4	[0,3,2]	17	yes
5	[0,3,2]	12	no

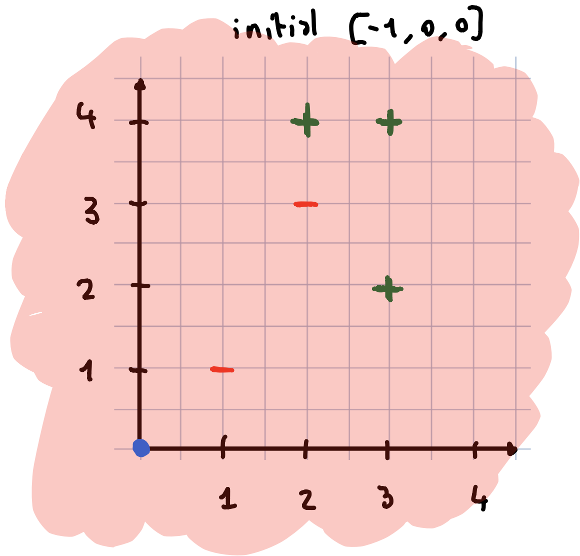
$$w \leftarrow w + y^* f(x)$$

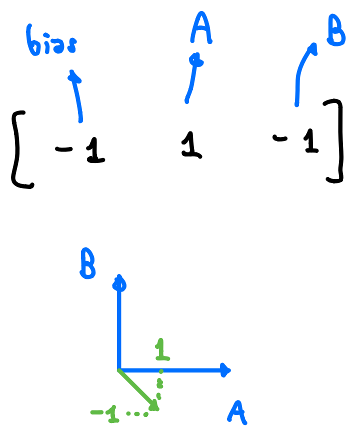
Step 2 update: $w \leftarrow [-1, 0, 0] + 1\,[1 \quad 3 \quad 2] = [0 \quad 3 \quad 2]$

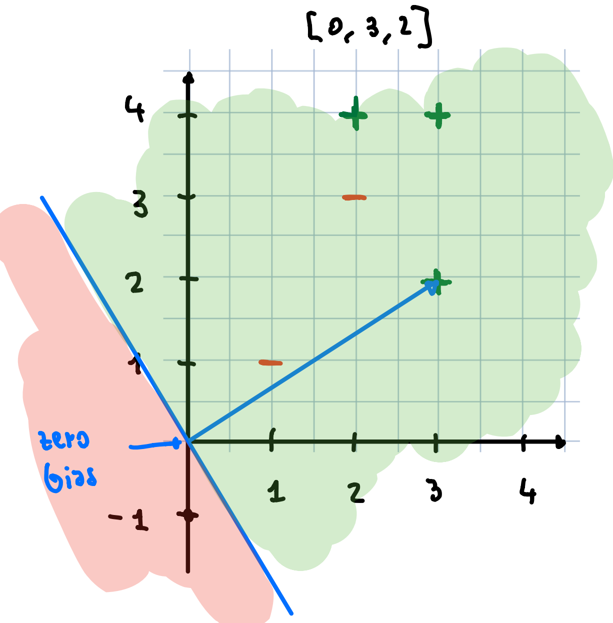
Step 5 update: $w \leftarrow [0, 3, 2] - [1 \quad 2 \quad 3] = [-1 \quad 1 \quad -1]$

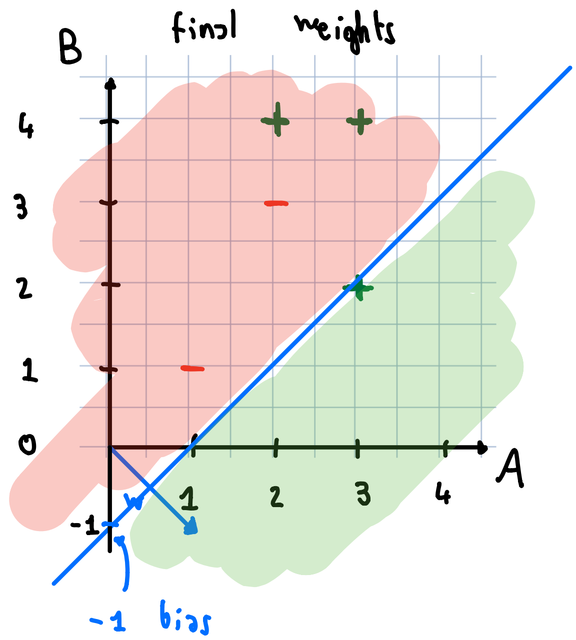
Final weights: $[-1 \quad 1 \quad -1]$

3. Have weights been learned that separate the data?









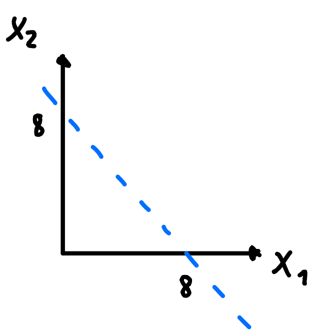
4. More generally, irrespective of the training data, you want to know if your features are powerful enough to allow you to handle a range of scenarios. Circle the scenarios for which a perceptron using the features above can indeed perfectly classify movies which are profitable according to the given rules:

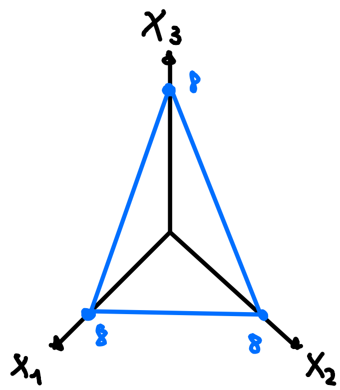
   (a) Your reviewers are awesome: if the total of their scores is more than 8, then the movie will definitely be profitable, and otherwise it won't be.

   (b) Your reviewers are art critics. Your movie will be profitable if and only if each reviewer gives either a score of 2 or a score of 3.

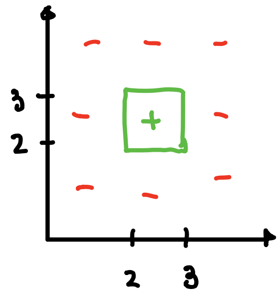
   (c) Your reviewers have weird but different tastes. Your movie will be profitable if and only if both reviewers agree.

a) $x_1 + x_2 + \ldots + x_n > 8$

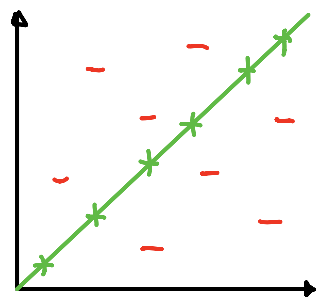




b)



c)



3 More Perceptron

We would like to use a perceptron to train a classifier with 2 features per point and labels +1 or −1. Consider the following labeled training data:

Features (x_1, x_2)	Label y^*
$(-1, 2)$	1
$(3, -1)$	-1
$(1, 2)$	-1
$(3, 1)$	1

1. Our two perceptron weights have been initialized to $w_1 = 2$ and $w_2 = -2$. After processing the first point with the perceptron algorithm, what will be the updated values for these weights?

first datapoint

$$y^1 = w^T f(x^1) = \begin{bmatrix} 2 & -2 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \end{bmatrix} = -2 - 4 = -6$$

$$\therefore \quad y \neq y^* \Rightarrow w \leftarrow w + y^* f(x)$$

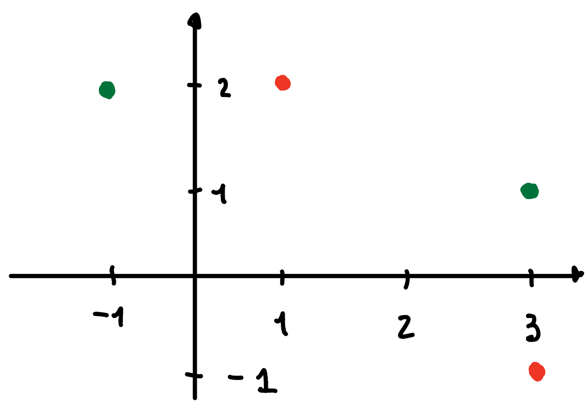
$$\leftarrow \begin{bmatrix} 2 \\ -2 \end{bmatrix} + \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

2. After how many steps will the perceptron algorithm converge? Write "never" if it will never converge.

   Note: one step means processing one point. Points are processed in order and then repeated, until convergence.

Never: not linearly separable!

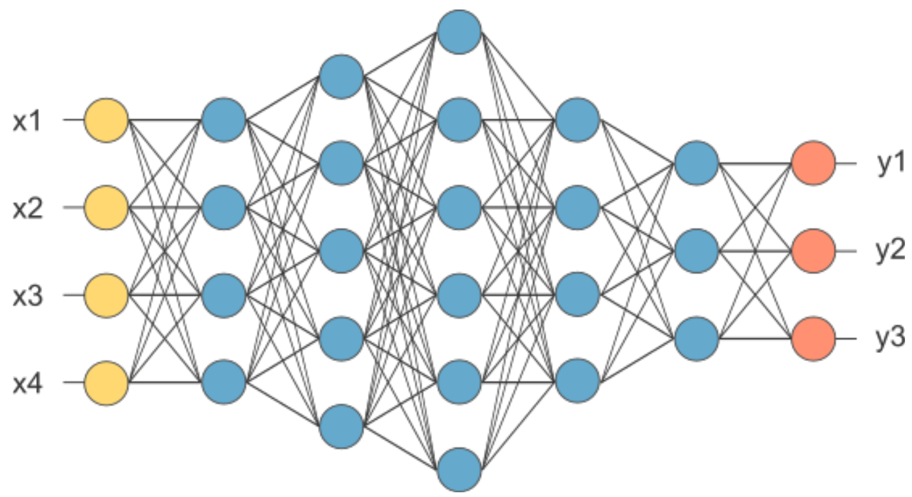
Features (x_1, x_2)	Label y^*
$(-1, 2)$	1
$(3, -1)$	-1
$(1, 2)$	-1
$(3, 1)$	1



Perceptrons / Neural Nets

feature extraction

Structure



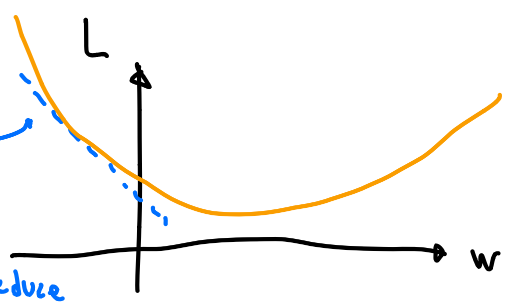
→ Predicting Disease

→ Best stock market action

→ Image recognition

How well are we doing? To assess this we use a

Loss function



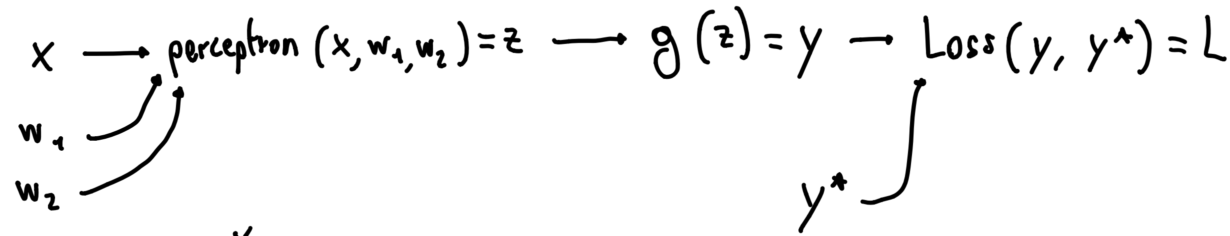
derivative tells us what direction to go in to reduce loss

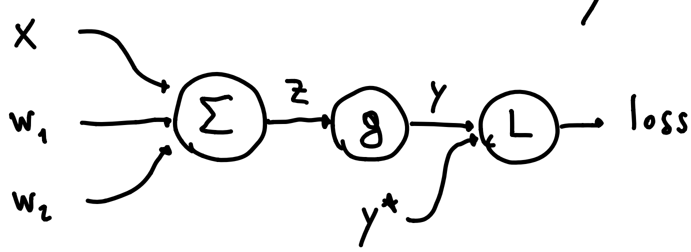
Perceptron → Neural Nets

Instead of the standard perceptron algorithm, we decide to treat the perceptron as a single node neural network and update the weights using gradient descent on the loss function.

The loss function for one data point is $Loss(y, y^*) = \frac{1}{2}(y - y^*)^2$, where y^* is the training label for a given point and y is the output of our single node network for that point. We will compute a score $z = w_1x_1 + w_2x_2$, and then predict the output using an activation function g: $y = g(z)$.

1. Given a general activation function $g(z)$ and its derivative $g'(z)$, what is the derivative of the loss function with respect to w_1 in terms of $g, g', y^*, x_1, x_2, w_1$, and w_2?





$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_1} = \underbrace{\frac{\partial L}{\partial y}}_{1}\ \underbrace{\frac{\partial y}{\partial z}}_{2}\ \underbrace{\frac{\partial z}{\partial w_1}}_{3}$$

1 $\quad \frac{\partial L}{\partial y} = \frac{\partial}{\partial y}\frac{1}{2}(y - y^*)^2 = \frac{1}{2}\,2\,(y - y^*) = y - y^*$

2 $\quad \frac{\partial y}{\partial z} = \frac{\partial}{\partial z} g(z) = g'(z)$

3 $\quad \frac{\partial z}{\partial w_1} = \frac{\partial}{\partial w_1}\text{perceptron}(x, w_1, w_2) = \frac{\partial}{\partial w_1}\sum_i^2 x_i w_i$

$$= \frac{\partial}{\partial w_1}(x_1 w_1 + x_2 w_2) = x_1$$

→ note that this is equal to weight update in perceptron alg.

$$\therefore \frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial w_1} = (y - y^*)\,g'(z)\,x_1$$

$$= (g(z) - y^*)\,g'(z)\,x_1$$

$$= \left(g\left(\sum_i^2 x_i w_i\right) - y^* \right) g'\left(\sum_i^2 x_i w_i\right) x_1$$

Without explicitly using computational graph:

$$\frac{\partial Loss}{\partial w_1} = \frac{\partial}{\partial w_1} \frac{1}{2} \left(y - y^* \right)^2$$

The loss function for one data point is $Loss(y, y^*) = \frac{1}{2}(y - y^*)^2$, where y^* is the training label for a given point and y is the output of our single node network for that point. We will compute a score $z = w_1x_1 + w_2x_2$, and then predict the output using an activation function g: $y = g(z)$.

$$= \frac{\partial}{\partial w_1} \frac{1}{2} \left(g(z) - y^* \right)^2$$

$$= \frac{\partial}{\partial w_1} \frac{1}{2} \left(g\left(x_1 w_1 + x_2 w_2 \right) - y^* \right)^2$$

$$= \left(g\left(x_1 w_1 + x_2 w_2 \right) - y^* \right) \frac{\partial}{\partial w_1} \left(g\left(x_1 w_1 + x_2 w_2 \right) - y^* \right)$$

$$= \left(g\left(x_1 w_1 + x_2 w_2 \right) - y^* \right) \underbrace{g'\left(x_1 w_1 + x_2 w_2 \right)} x_1$$

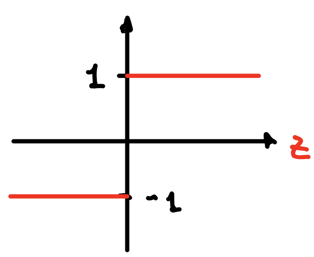
2. For this question, the specific activation function that we will use is

$$g(z) = 1 \text{ if } z \geq 0 \text{ , or } -1 \text{ if } z < 0$$

Given the gradient descent equation $w_i \leftarrow w_i - \alpha \frac{\partial Loss}{\partial w_i}$, update the weights for a single data point. With initial weights of $w_1 = 2$ and $w_2 = -2$, what are the updated weights after processing the first point?

For this question

$$g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases} \Rightarrow g'(z) = 0 \quad \forall z$$



$$\Rightarrow \frac{\partial Loss}{\partial w_1} = 0$$

Therefore our weight update:

$$w_i \leftarrow w_i - \alpha \underbrace{\frac{\partial Loss}{\partial w_1}}_{0} = w_i - \alpha \cdot 0 = w_i$$

Our weights will remain unchanged at every iteration

3. What is the most critical problem with this gradient descent training process with that activation function?

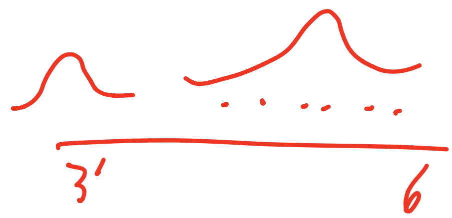
Derivative is zero!

2 Maximum Likelihood

In this problem, we're going to estimate the parameters of a normal distribution using maximum likelihood. Note that continuous distributions are not in scope for exams/homeworks, but this is good practice for probability, maximum likelihood, and calculus. A normal distribution is a continuous probability distribution over real numbers X with mean μ and standard deviation σ. The probabilitiy is given as follows:

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

Given a dataset $D = \{x_1, \ldots, x_n\}$, find the maximum likelihood estimate of μ, σ.



Likelihood of dataset:

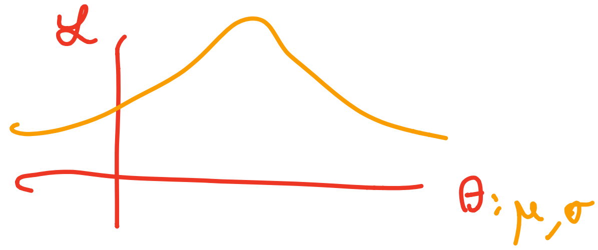
R.V. for ith datapoint

actual value of ith datapoint

$$\mathcal{L}(x_1, \ldots, x_n) = P(X_1 = x_1, \ldots, X_n = x_n)$$

$$= \prod_i^n P(X_i = x_i)$$ as we assume each datapoint to be independent

$$= \prod_i^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$



$$\mu_{MLE}, \sigma_{MLE} = \arg\max_{\mu,\sigma} \mathcal{L}(x_1, \ldots, x_n) = \arg\max_{\mu,\sigma} \prod_i^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

$\Updownarrow$ as we are taking argmax, this is the same

$$\arg\max_{\mu,\sigma} \log \mathcal{L}(x_1, \ldots, x_n) = \arg\max_{\mu,\sigma} \log \prod_i^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

denotes log-likelihood

$$\Rightarrow \mu_{MLE}, \sigma_{MLE} = \arg\max_{\mu,\sigma} \ell(x_1, \ldots, x_n) = \arg\max_{\mu,\sigma} \sum_i^n \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \right)$$

$$= \arg\max_{\mu,\sigma} -\sum_i^n \frac{1}{2} \log 2\pi\sigma^2 + \frac{(x_i-\mu)^2}{2\sigma^2}$$

We want to find input σ, μ that maximize the function above: take derivative and set to 0!

$$\frac{\partial \ell}{\partial \mu} = \frac{\partial}{\partial \mu}\left(-\sum_i^n \frac{1}{2}\log 2\pi\sigma^2 + \frac{(x_i-\mu)^2}{2\sigma^2}\right)$$

$$= \sum_i^n \frac{\cancel{2}(x_i-\mu)}{\cancel{2}\sigma^2} = 0$$

$$\Rightarrow \quad \sum_i^n (x_i-\mu) = 0 \quad \Leftrightarrow \left(\sum_i^n x_i\right) - n\mu = 0$$

$$\Leftrightarrow \mu_{MLE} = \frac{\sum_i^n x_i}{n}$$

Same can be done to find σ_{MLE} (see solutions)