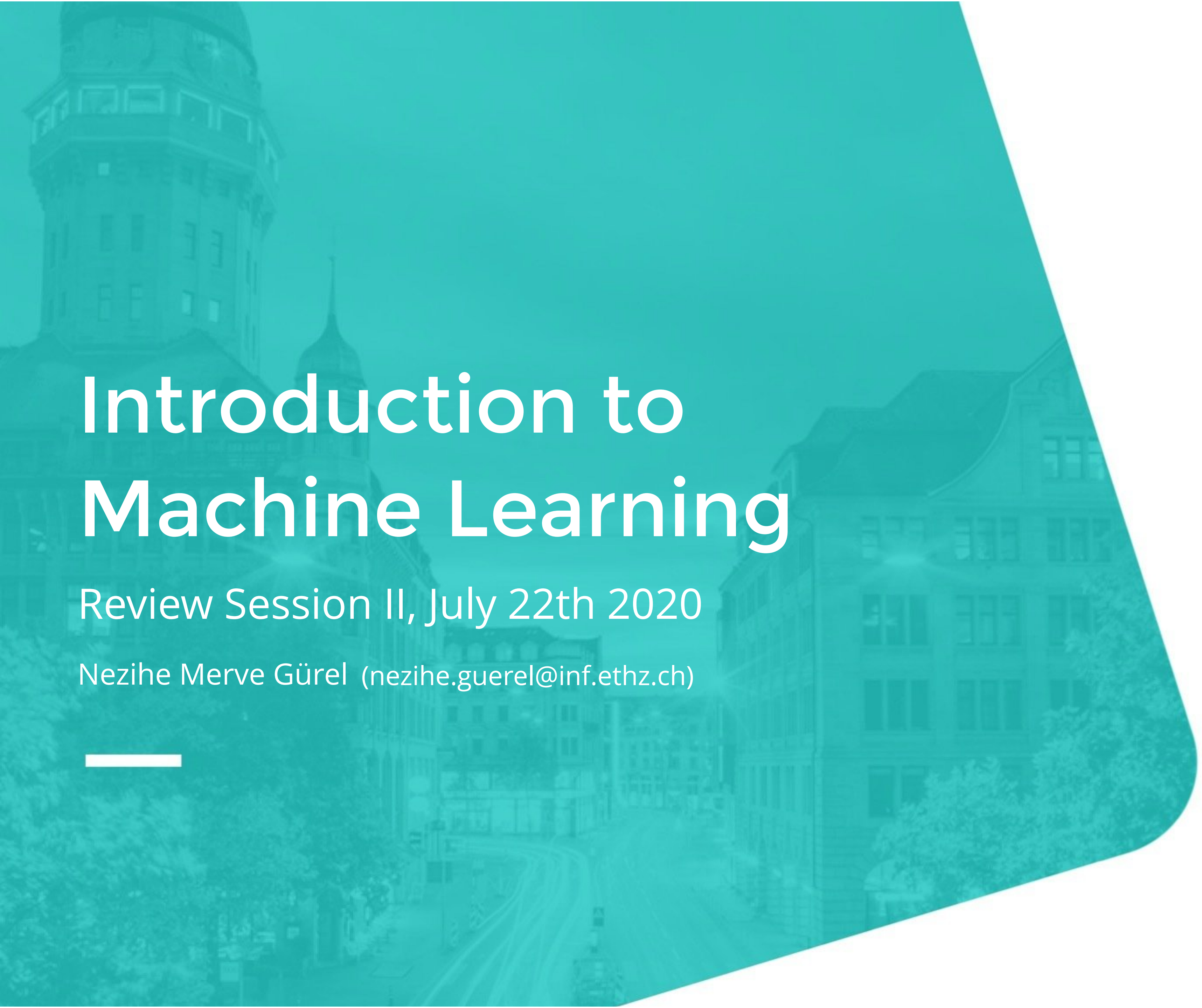


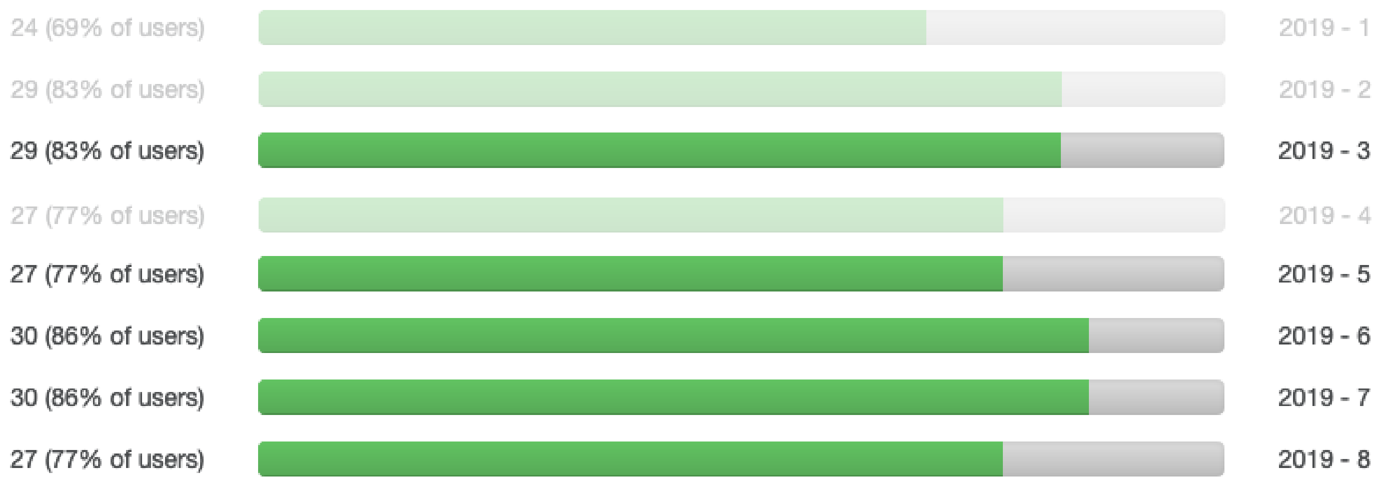
Introduction to Machine Learning

Review Session II, July 22th 2020

Nezihe Merve Gürel (nezihe.guerel@inf.ethz.ch)

Outline

Survey results on Piazza:



Agenda for today: Exam 2019

• Question 3	Convolutional Neural Networks
• Question 5	Clustering
• Question 6	Dimensionality Reduction
10 mins break	
• Question 7	Linear Discriminant Analysis
• Question 8	Gaussian Mixture Models and EM Algorithm

Next in Agenda

Exam 2019

- Question 3 Convolutional Neural Networks
- Question 5 Clustering
- Question 6 Dimensionality Reduction
- Question 7 Linear Discriminant Analysis
- Question 8 Gaussian Mixture Models and EM Algorithm

Recap: Convolutional Neural Networks

Key ideas:

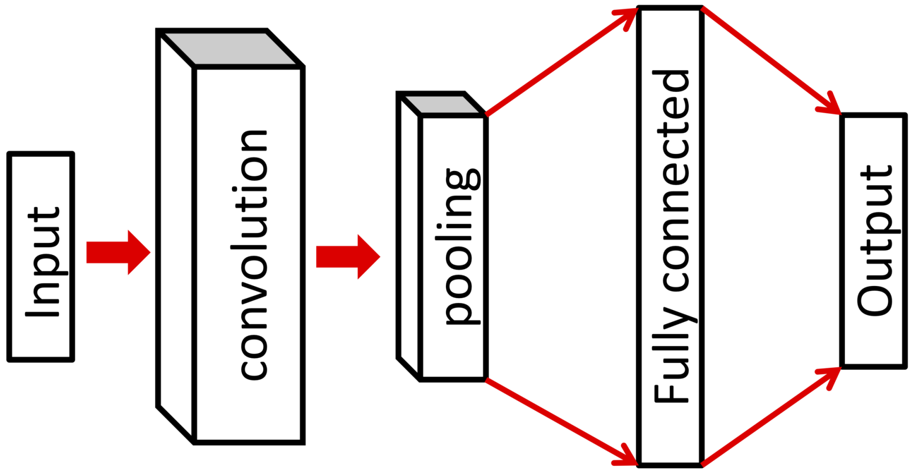
- Robust predictions under transformations of data
- Less parameters (scalability, overfitting)

Output dimensions determined by:

- Input of size: $n \times n$
- M filters of size: $f \times f$
- Stride: s
- Padding: p

$$\boxed{\begin{array}{l}\text{Output dimension: } L \times L \times M \\ \text{where } L = \frac{n-f+2p}{s} + 1\end{array}}$$

CNN architecture:



Training via Backpropagation!

Recap: Convolutional Neural Networks

Key ideas:

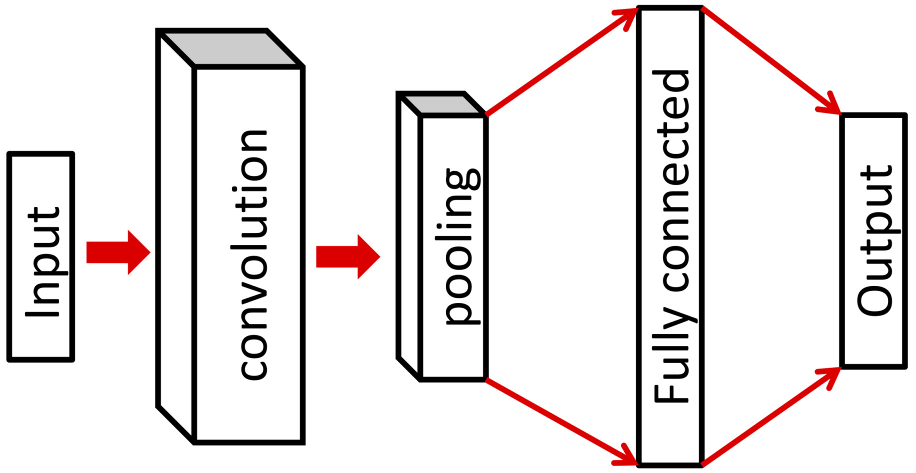
- Robust predictions under transformations of data
- Less parameters (scalability, overfitting)

Output dimensions determined by:

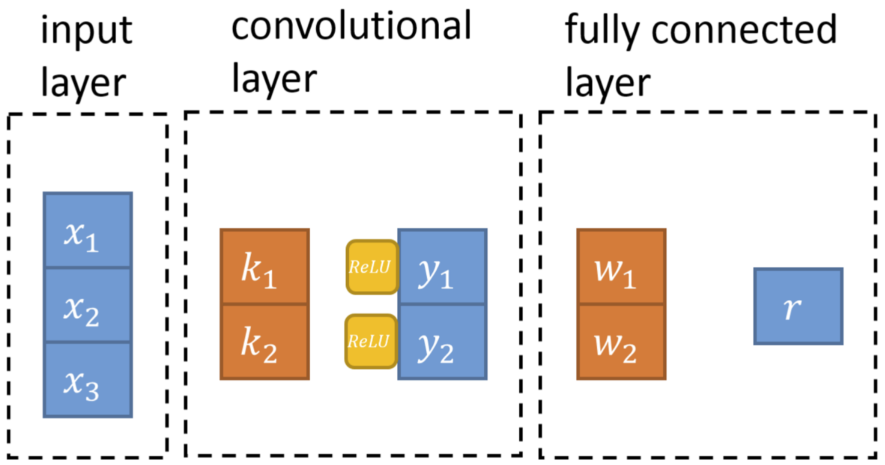
- Input of size: $n \times n$
- M filters of size: $f \times f$
- Stride: s
- Padding: p

Output dimension: $L \times L \times M$
where $L = \frac{n-f+2p}{s} + 1$

CNN architecture:



Training via Backpropagation!



Example: a simple convolutional network

- One dimensional input with 3 features,
- A single filter of size 2,
- ReLU activation function,
- No pooling, no padding, stride of 1

Exam 2019 - 3.i

Consider a one dimensional convolutional neural network given in the figure below. The input to the network has three features x_1 to x_3 each in $\mathbb{R}$. The network consists of the following layers,

1. a convolution layer using a kernel of size two with weights k_1 and k_2. The convolution uses no padding and stride of length one,
2. afterwards ReLU ($\text{ReLU}(x) = \max(x, 0)$) is applied to get outputs y_1 and y_2
3. a fully connected layer applied to y_1 and y_2 with the weights w_1 and w_2 to get a scalar output r.

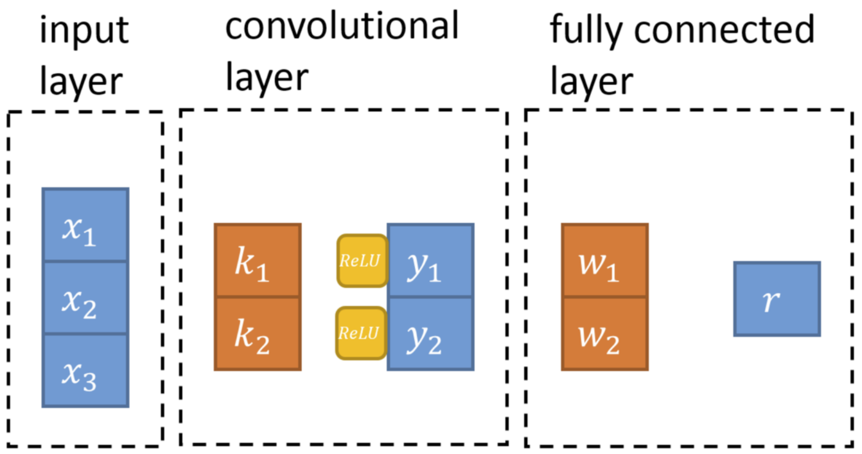


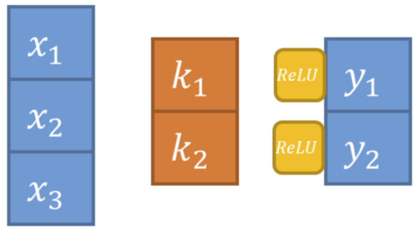
Figure 2: Simple convolutional neural network.

(i) *Forward Propagation (Num)*

Suppose all weights (k_1, k_2, w_1, w_2) of the network are initialized to be 0.5. Consider an input example $\mathbf{x} = ([1.0, 1.0, -1.0]^\top$. What is the value of r after forward propagating this input example?

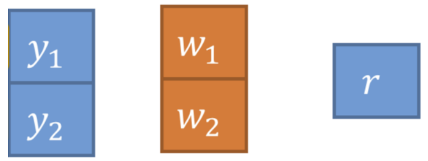
Solution to i

Let ψ denote ReLU activation function. At the convolutional layer, we have:



$$y_1 = \psi([x_1\ x_2] \cdot [k_1\ k_2]) = \psi(x_1 k_1 + x_2 k_2)$$
$$y_2 = \psi([x_2\ x_3] \cdot [k_1\ k_2]) = \psi(x_2 k_1 + x_3 k_2)$$

At the fully connected layer:



$$r = w_1 y_1 + w_2 y_2$$

Inserting y_1, y_2 into r gives us:

$$r = w_1 \psi(x_1 k_1 + x_2 k_2) + w_2 \psi(x_2 k_1 + x_3 k_2)$$
$$r\Big|_{\mathbf{x}=[1,1,-1], k_{1:2}=0.5, w_{1:2}=0.5} = 0.5\psi(1) + 0.5\psi(0) = 0.5$$

Exam 2019 - 3.ii

Consider a one dimensional convolutional neural network given in the figure below. The input to the network has three features x_1 to x_3 each in $\mathbb{R}$. The network consists of the following layers,

1. a convolution layer using a kernel of size two with weights k_1 and k_2. The convolution uses no padding and stride of length one,
2. afterwards ReLU ($\text{ReLU}(x) = \max(x, 0)$) is applied to get outputs y_1 and y_2
3. a fully connected layer applied to y_1 and y_2 with the weights w_1 and w_2 to get a scalar output r.

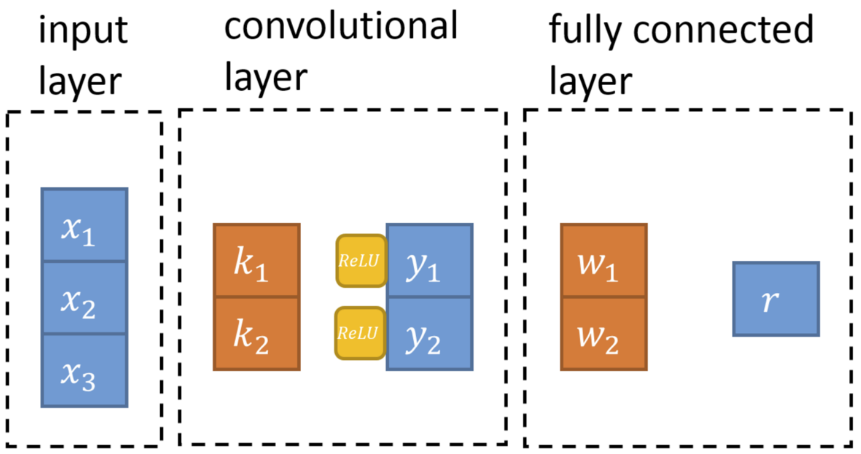


Figure 2: Simple convolutional neural network.

(ii) *Backward Propagation (Num)*

Suppose you are given, the same training example and the same initial weights as in the previous question. You want to backpropagate the loss to the input layer. Suppose $\frac{\partial L}{\partial r} = 2$ for the example point, what is $\frac{\partial L}{\partial x_1}$ for the input example.

Solution to ii

Let ψ denote ReLU activation function.

At the fully connected layer:

$$r = w_1 y_1 + w_2 y_2$$

At the convolutional layer, we have:

$$y_1 = \psi([x_1 \; x_2] \cdot [k_1 \; k_2]) = \psi(x_1 k_1 + x_2 k_2)$$
$$y_2 = \psi([x_2 \; x_3] \cdot [k_1 \; k_2]) = \psi(x_2 k_1 + x_3 k_2)$$

Then we use chain rule to express $\frac{\partial L}{\partial x_1}$ such that: $\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial r}\frac{\partial r}{\partial x_1} = \frac{\partial L}{\partial r}\frac{\partial r}{\partial y_1}\frac{\partial y_1}{\partial x_1}$

Given $\frac{\partial L}{\partial r} = 2$ we further compute: $\frac{\partial r}{\partial y_1} = w_1$ and $\frac{\partial y_1}{\partial x_1} = k_1$ if $x_1 k_1 + x_2 k_2 \geq 0$ and 0 otherwise. Noting $x_1 k_1 + x_2 k_2 = 1$ for the specified values, we further have:

$$\frac{\partial L}{\partial x_1}\Big|_{\mathbf{x}=[1,1,-1],k_{1:2}=0.5,w_{1:2}=0.5} = 2 \cdot 0.5 \cdot 0.5 = 0.5$$

Exam 2019 - 3.iii-iv

(iii) *Activation Functions (T/F)*
Let $x \in \mathbb{R}$, and $\sigma(x) = \frac{1}{1+\exp(-x)}$ be the sigmoid activation function. Determine which of the following statements are true or false:

True **False**

☒ ☐ $\mathrm{ReLU}(x)$ is a non-linear activation function

☐ ☒ The sigmoid activation can be represented as a finite combination of argument scaled $\mathrm{ReLU}(x)$. In other words, $\sigma(x) = \sum_{i=1}^{k} \mathrm{ReLU}(a_i x)$, where $a_i \in \mathbb{R}$.

☒ ☐ The absolute value function $|x|$ can be represented exactly as finite combination of argument scaled $\mathrm{ReLU}(x)$. In other words, $|x| = \sum_{i=1}^{k} \mathrm{ReLU}(a_i x)$, where $a_i \in \mathbb{R}$.

☐ ☒ $\lim_{y\to\infty} \frac{d\sigma(x)}{dx}|_{x=y} = \infty$, where the vertical rule implies that the function is evaluated at this point.

Solution to iii

1. ReLU $\psi(x) = \max(x, 0)$ is a nonlinear activation function (it does not preserve addition or scalar multiplication).

2. Hint: ReLU is not differentiable

3. First, note that $|x| = \; x \;\; \text{if } x \geq 0$ and $|x| = -x \;\; \text{otherwise}$

Second, look closer into ReLU:

$$\psi(x) = \begin{matrix} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{matrix} \qquad \psi(-x) = \begin{matrix} 0 & \text{if } x \geq 0 \\ -x & \text{if } x < 0 \end{matrix}$$

Hence $|x| = \;\; \psi(x) \;\; \text{if } x \geq 0$ and $|x| = \psi(-x) \;\; \text{otherwise}$

Finally, note that: $|x| = \psi(x) + \psi(-x)$

4. Note that $\frac{d}{dx}\sigma(x) = \sigma(x)(1 - \sigma(x))$

Therefore $\lim_{y\to\infty} \sigma(y)(1-\sigma(y)) = \lim_{y\to\infty} \frac{\exp(-y)}{(1+\exp(-y))^2} = 0$

Recall: vanishing gradients!

(iv) *Learning Rate (T/F)*
Decide which of the following choices for η_t defines a converging SGD method in general (assume the losses are bounded).

True **False**

☐ ☒ $\eta_t \propto \log(t)$

☒ ☐ $\eta_t \propto \frac{1}{t}$

☒ ☐ $\eta_t \propto \min(0.1, \frac{1}{t})$

☐ ☒ $\eta_t \propto e^t$

Solution to iv

Robbins-Monro conditions: Learning rate η_t guarantees convergence if $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$

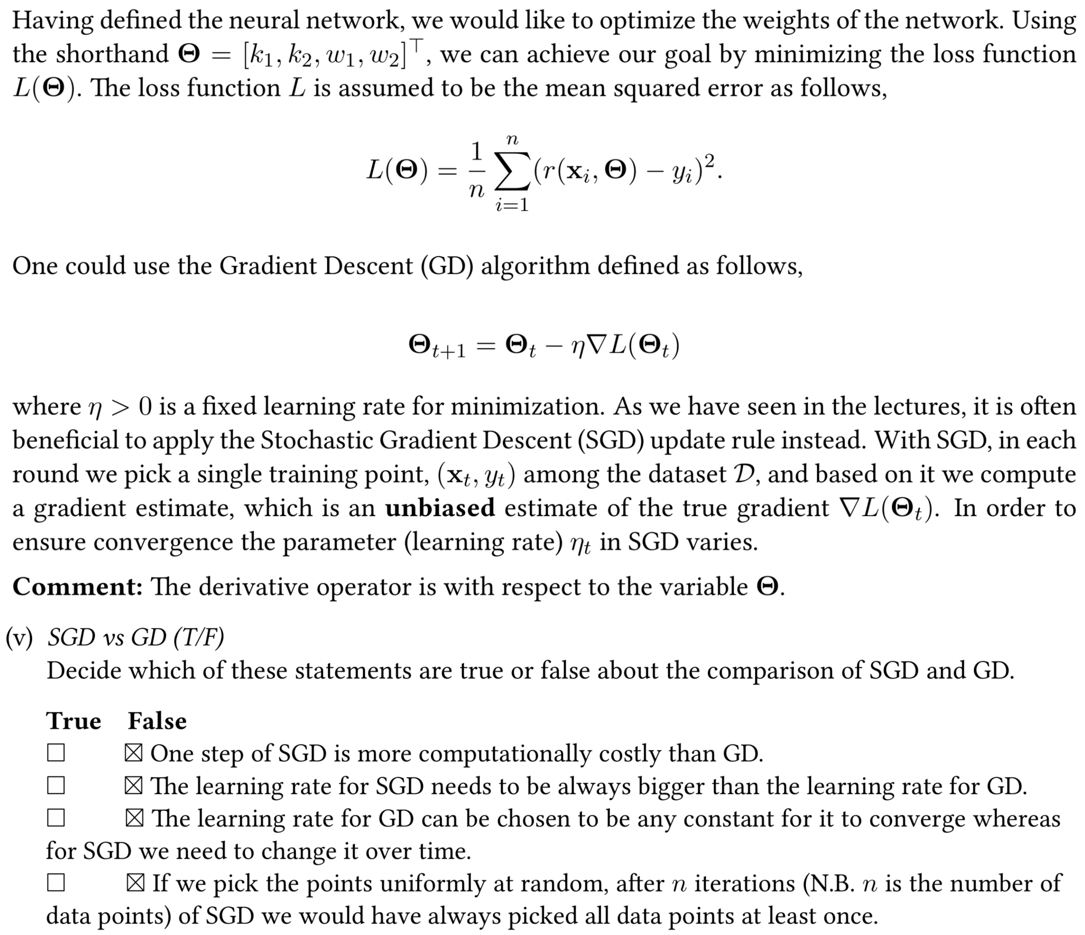
1. $\sum_t \eta_t \big|_{\eta_t=\log(t)} = \infty$ and $\sum_t \eta_t^2 \big|_{\eta_t=\log(t)} = \infty$

2. $\sum_t \eta_t \big|_{\eta_t=1/t} = \infty$ and $\sum_t \eta_t^2 \big|_{\eta_t=1/t} < \infty$

3. $\sum_t \eta_t \big|_{\eta_t=\min(0.1,1/t)} = \infty$ and $\sum_t \eta_t^2 \big|_{\eta_t=\min(0.1,1/t)} < \infty$

4. $\sum_t \eta_t \big|_{\eta_t=\exp(t)} = \infty$ and $\sum_t \eta_t^2 \big|_{\eta_t=\exp(t)} = \infty$

Exam 2019 - 3.v-vi



Having defined the neural network, we would like to optimize the weights of the network. Using the shorthand $\mathbf{\Theta} = [k_1, k_2, w_1, w_2]^\top$, we can achieve our goal by minimizing the loss function $L(\mathbf{\Theta})$. The loss function L is assumed to be the mean squared error as follows,

$$L(\mathbf{\Theta}) = \frac{1}{n}\sum_{i=1}^{n}(r(\mathbf{x}_i, \mathbf{\Theta}) - y_i)^2.$$

One could use the Gradient Descent (GD) algorithm defined as follows,

$$\mathbf{\Theta}_{t+1} = \mathbf{\Theta}_t - \eta\nabla L(\mathbf{\Theta}_t)$$

where $\eta > 0$ is a fixed learning rate for minimization. As we have seen in the lectures, it is often beneficial to apply the Stochastic Gradient Descent (SGD) update rule instead. With SGD, in each round we pick a single training point, $(\mathbf{x}_t, y_t)$ among the dataset $\mathcal{D}$, and based on it we compute a gradient estimate, which is an **unbiased** estimate of the true gradient $\nabla L(\mathbf{\Theta}_t)$. In order to ensure convergence the parameter (learning rate) η_t in SGD varies.

Comment: The derivative operator is with respect to the variable $\mathbf{\Theta}$.

(v) *SGD vs GD (T/F)*
Decide which of these statements are true or false about the comparison of SGD and GD.

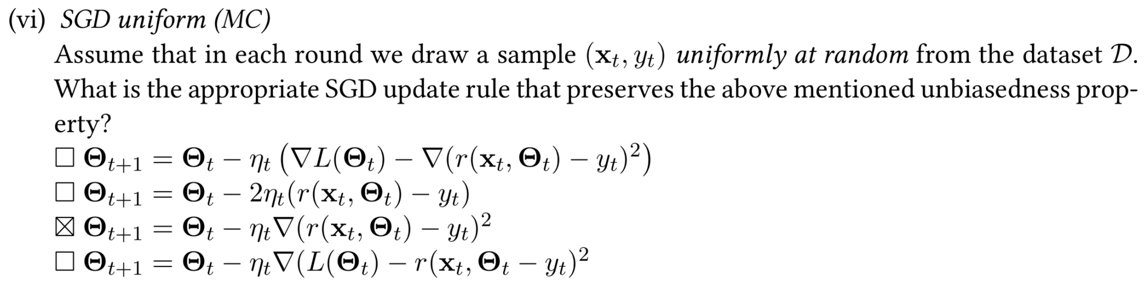
True **False**
☐ ☒ One step of SGD is more computationally costly than GD.
☐ ☒ The learning rate for SGD needs to be always bigger than the learning rate for GD.
☐ ☒ The learning rate for GD can be chosen to be any constant for it to converge whereas for SGD we need to change it over time.
☐ ☒ If we pick the points uniformly at random, after n iterations (N.B. n is the number of data points) of SGD we would have always picked all data points at least once.

Solution to v

1. Computing gradient requires summing over all data

2-3. Gradient Descent: Choose η_t sufficiently small (fixed or adaptive)
Stochastic Gradient Descent: η_t adaptive over time

4. Recall that we draw a data point uniformly at random with replacement



(vi) *SGD uniform (MC)*
Assume that in each round we draw a sample $(\mathbf{x}_t, y_t)$ *uniformly at random* from the dataset $\mathcal{D}$. What is the appropriate SGD update rule that preserves the above mentioned unbiasedness property?
☐ $\mathbf{\Theta}_{t+1} = \mathbf{\Theta}_t - \eta_t\left(\nabla L(\mathbf{\Theta}_t) - \nabla(r(\mathbf{x}_t, \mathbf{\Theta}_t) - y_t)^2\right)$
☐ $\mathbf{\Theta}_{t+1} = \mathbf{\Theta}_t - 2\eta_t(r(\mathbf{x}_t, \mathbf{\Theta}_t) - y_t)$
☒ $\mathbf{\Theta}_{t+1} = \mathbf{\Theta}_t - \eta_t\nabla(r(\mathbf{x}_t, \mathbf{\Theta}_t) - y_t)^2$
☐ $\mathbf{\Theta}_{t+1} = \mathbf{\Theta}_t - \eta_t\nabla(L(\mathbf{\Theta}_t) - r(\mathbf{x}_t, \mathbf{\Theta}_t - y_t)^2$

Solution to vi

Gradient at t: g_t

Unbiasness of gradient: Choose any g_t such that $\mathbb{E}g_t = \nabla L(\Theta_t)$

Let's analyze $\nabla L(\Theta_t)$ further:

$$\nabla L(\Theta_t) = \nabla\frac{1}{n}\sum_{i=1}^{n}(r(\mathbf{x}_i, \Theta_t) - y_i)^2 = \sum_{i=1}^{n}\frac{1}{n}\nabla(r(\mathbf{x}_i, \Theta_t) - y_i)^2$$

SGD computes gradient at a randomly sampled point $\mathbf{x}_t$:

$$g_t = \nabla(r(X, \Theta_t) - y_i)^2 \quad \text{where } X \sim \mathrm{Unif}(\mathcal{D}).$$

Hence $$\mathbb{E}g_t = \textstyle\sum_{t\in\mathcal{D}} P(X = \mathbf{x}_t)\nabla(r(\mathbf{x}_t, \Theta_t) - y_t)^2\Big|_{X\sim\mathrm{Unif}}$$
$$= \sum_{t=1}^{n}\frac{1}{n}\nabla(r(\mathbf{x}_t, \Theta_t) - y_t)^2 = \nabla L(\Theta_t)$$

Exam 2019 - 3.vii

Having defined the neural network, we would like to optimize the weights of the network. Using the shorthand $\mathbf{\Theta} = [k_1, k_2, w_1, w_2]^\top$, we can achieve our goal by minimizing the loss function $L(\mathbf{\Theta})$. The loss function L is assumed to be the mean squared error as follows,

$$L(\mathbf{\Theta}) = \frac{1}{n}\sum_{i=1}^{n}(r(\mathbf{x}_i, \mathbf{\Theta}) - y_i)^2.$$

One could use the Gradient Descent (GD) algorithm defined as follows,

$$\mathbf{\Theta}_{t+1} = \mathbf{\Theta}_t - \eta \nabla L(\mathbf{\Theta}_t)$$

where $\eta > 0$ is a fixed learning rate for minimization. As we have seen in the lectures, it is often beneficial to apply the Stochastic Gradient Descent (SGD) update rule instead. With SGD, in each round we pick a single training point, $(\mathbf{x}_t, y_t)$ among the dataset $\mathcal{D}$, and based on it we compute a gradient estimate, which is an **unbiased** estimate of the true gradient $\nabla L(\mathbf{\Theta}_t)$. In order to ensure convergence the parameter (learning rate) η_t in SGD varies.

Comment: The derivative operator is with respect to the variable $\mathbf{\Theta}$.

(vii) *SGD sampling (MC)*
Assume that in each round we draw a sample $(\mathbf{x}_t, y_t)$ such that

$$\mathrm{P}\left((\mathbf{x}_t, y_t) \text{ is sampled}\right) = \lambda_t , \qquad \forall t \in \{1 \dots , N\}$$

What is the appropriate SGD update rule that preserves the above mentioned unbiasedness property?

☐ $\mathbf{\Theta}_{t+1} = \mathbf{\Theta}_t - \eta_t \lambda_t \left(\nabla L(\mathbf{\Theta}_t) - \nabla(r(\mathbf{x}_t, \mathbf{\Theta}_t) - y_t)^2\right)$
☐ $\mathbf{\Theta}_{t+1} = \mathbf{\Theta}_t - \frac{2}{\lambda_t}\eta_t(r(\mathbf{x}_t, \mathbf{\Theta}_t) - y_t)$
[✓ $\mathbf{\Theta}_{t+1} = \mathbf{\Theta}_t - \frac{\eta_t}{\lambda_t}\nabla(r(\mathbf{x}_t, \mathbf{\Theta}_t) - y_t)^2$]
☐ $\mathbf{\Theta}_{t+1} = \mathbf{\Theta}_t - \frac{\eta_t}{\lambda_t}\nabla(L(\mathbf{\Theta}_t) - r(\mathbf{x}_t, \mathbf{\Theta}_t - y_t)^2$

Solution to vii

Gradient at t: g_t

Unbiasness of gradient: Choose any g_t such that $\mathbb{E}g_t = \nabla L(\Theta_t)$

Let's analyze $\nabla L(\Theta_t)$ further:

$$\nabla L(\Theta_t) = \nabla \frac{1}{n}\sum_{i=1}^{n}(r(\mathbf{x}_i, \Theta_t) - y_i)^2 = \sum_{i=1}^{n}\frac{1}{n}\nabla(r(\mathbf{x}_i, \Theta_t) - y_i)^2$$

SGD computes gradient at a randomly sampled point $\mathbf{x}_t$:

$$g_t = \gamma_t \nabla(r(X, \Theta_t) - y_i)^2 \text{ where } P(X = \mathbf{x}_t) = \lambda_t .$$

Hence $\mathbb{E}g_t = \sum_{t\in\mathcal{D}} P(X = \mathbf{x}_t)\gamma_t \nabla(r(\mathbf{x}_t, \Theta_t) - y_t)^2\Big|_{P(X=\mathbf{x}_t)=\lambda_t}$

$$= \sum_{t=1}^{n}\lambda_t\gamma_t\nabla(r(\mathbf{x}_t, \Theta_t) - y_t)^2$$

If $\gamma_t \propto \frac{1}{\lambda_t}$ then $\mathbb{E}g_t \propto \nabla L(\Theta_t)$ (Precisely, $\gamma_t = \frac{1}{n\lambda_t}$ recovers the uniform sampling)

Next in Agenda

Exam 2019

- Question 3 Convolutional Neural Networks
- Question 5 Clustering
- Question 6 Dimensionality Reduction
- Question 7 Linear Discriminant Analysis
- Question 8 Gaussian Mixture Models and EM Algorithm

Recap: k-Means Clustering

Basics:

- Data points are in Euclidean space $\mathbf{x}_i \in \mathbb{R}^d$
- Cluster centers given by $\mu_j \in \mathbb{R}^d$ and each data point is assigned to the closest center

$$z_i \rightarrow \arg\min_j \|\mathbf{x}_i - \mu_j\|_2^2 \quad \text{where } z_i = [1:k] \text{ is the cluster of } \mathbf{x}_i$$

- Choose the centers that minimizes the average square distance $\hat{R}(\mu)$:

$$\hat{\mu} = \arg\min_\mu \hat{R}(\mu) \quad \text{where } \hat{R}(\mu) = \hat{R}(\mu_1, \ldots, \mu_k) = \sum_{i=1}^{N} \min_{j=[1:k]} \|\mathbf{x}_i - \mu_j\|_2^2$$

The Algorithm:

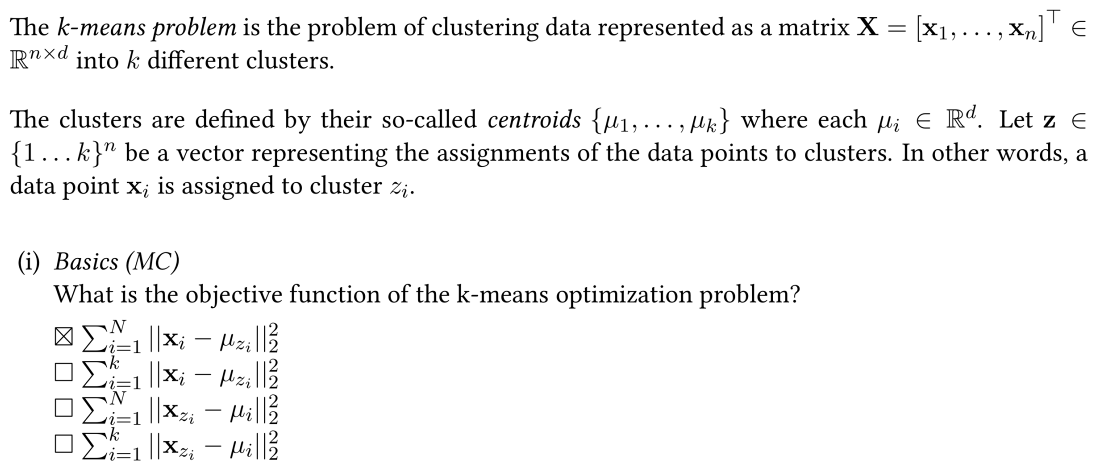
Initialize $\mu^{(0)} = \mu^{(0)}_{1:k}$,

Assignment of data points to clusters: $z_i^{(t)} \rightarrow \arg\min_j \|\mathbf{x}_i - \mu_j^{(t-1)}\|_2^2$

Update cluster centers: $\mu_j^{(t)} \leftarrow \frac{1}{n_j} \sum_{i|z_i^{(t)}=j} \mathbf{x}_i$

Initialization of centroids
How to choose k?

Exam 2019 - 5.i-ii


The *k-means problem* is the problem of clustering data represented as a matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top \in \mathbb{R}^{n\times d}$ into k different clusters.

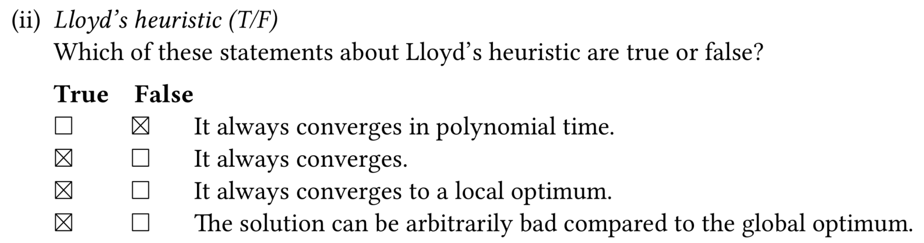
The clusters are defined by their so-called *centroids* $\{\mu_1, \ldots, \mu_k\}$ where each $\mu_i \in \mathbb{R}^d$. Let $\mathbf{z} \in \{1\ldots k\}^n$ be a vector representing the assignments of the data points to clusters. In other words, a data point $\mathbf{x}_i$ is assigned to cluster z_i.

(i) *Basics (MC)*
What is the objective function of the k-means optimization problem?

☒ $\sum_{i=1}^N ||\mathbf{x}_i - \mu_{z_i}||_2^2$
☐ $\sum_{i=1}^k ||\mathbf{x}_i - \mu_{z_i}||_2^2$
☐ $\sum_{i=1}^N ||\mathbf{x}_{z_i} - \mu_i||_2^2$
☐ $\sum_{i=1}^k ||\mathbf{x}_{z_i} - \mu_i||_2^2$

Solution to i

k-Means objective is the average squared distance between the data points and their respective centroids


(ii) *Lloyd's heuristic (T/F)*
Which of these statements about Lloyd's heuristic are true or false?

True	False	
☐	☒	It always converges in polynomial time.
☒	☐	It always converges.
☒	☐	It always converges to a local optimum.
☒	☐	The solution can be arbitrarily bad compared to the global optimum.

Solution to ii

1. It can take exponentially many steps to converge! (in practice it converges very fast)

2-3. We will show that the loss function is guaranteed to decrease monotonically

(a) Assignment step: $z_i^* \leftarrow \underset{j=[1:k]}{\arg\min} \|\mathbf{x}_i - \mu_j\|_2^2$

The change in the loss function is given by:

$$L(\mu, z^*) - L(\mu, z) = \sum_{i=1}^{N}(\|\mathbf{x}_i - \mu_{z_i^*}\|_2^2 - \|\mathbf{x}_i - \mu_{z_i}\|_2^2) \leq 0$$

(b) Refitting step: We can re-write the loss function as: $L(\mu, z) = \sum_{j=1}^{k} \sum_{i|z_i=j} \|\mathbf{x}_i - \mu_j\|_2^2$

After the assignment step, the change in the loss function becomes:

$$L(\mu^*, z^*) - L(\mu, z^*) = \sum_{j=1}^{k} \Big(\sum_{i=1}^{N} \|\mathbf{x}_i - \mu_j^*\|_2^2 - \sum_{i=1}^{N} \|\mathbf{x}_i - \mu_j\|_2^2 \Big) \leq 0$$

Hence, we can infer from above that $L(\mu^*, z^*) \leq L(\mu, z)$ and loss function is monotonically decreasing.

4. Due to its non-convex nature, the solution (a local minima) could be arbitrarily bad.

Exam 2019 - 5.iii-iv

The *k-means problem* is the problem of clustering data represented as a matrix $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d}$ into k different clusters.

The clusters are defined by their so-called *centroids* $\{\mu_1, \dots, \mu_k\}$ where each $\mu_i \in \mathbb{R}^d$. Let $\mathbf{z} \in \{1 \dots k\}^n$ be a vector representing the assignments of the data points to clusters. In other words, a data point $\mathbf{x}_i$ is assigned to cluster z_i.

(iii) *k-means++ (T/F)*

Which of these statements about the `k-means++` initialization are true or false?

True	False	
☐	☒	It chooses the initial centroids deterministically.
☐	☒	It chooses the initial centroids all at the same time but refines them sequentially if they are the same.
☒	☐	It chooses the initial centroids sequentially, where every new centroid is dependent on all the other already chosen ones.
☐	☒	The solution can be arbitrarily bad compared to the optimal centroids in expectation.

Solution to iii

1-3. k-means++ is a centroid initialization technique where centroids are selected sequentially such that they are likely to be in distinct clusters. The principle is to use importance sampling where sampling probabilities are updated adaptively (adaptive seeding).

4. The expected cost is **O(log k)** times the cost of optimal k-Means solution

(iv) *Criteria (T/F)*

Which of these methods can be used to choose the number of clusters k?

True	False	
☒	☐	Prior knowledge
☒	☐	The "elbow criterion"
☐	☒	Minimizing the k-means objective w.r.t. k on the training set
☐	☒	Minimizing the k-means objective w.r.t. k on a held-out validation set

Solution to iv

1-2. Strategies to determine *k* includes

- Exploratory analysis
- "elbow criterion": Choose a *k* such that a small decrease in loss is started to be observed (diminishing returns)
- Regularization: jointly minimize over *k* and centroids with a penalty on the number of clusters *k*

3-4. Why not cross validation?

As number of clusters increase, both training and generalization loss decrease as the average distance between the data points and their centroids decrease!

Exam 2019 - 5(2).i

Assume we have five one-dimensional points: $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 3, x_5 = 5$. We want to cluster these points using the k-means algorithm (Lloyd's heuristic).

(i) For $k = 2$, if we initialize the centroids as $\mu_1 = 0$ and $\mu_2 = 5$, which solution (in terms of μ_1, μ_2, and $\mathbf{z}$) will the algorithm converge to?

Hint: For example, $z_1 = 1$. Use the decimal point format. (Tolerance ± 0.01, i.e. you can round up or down.)

Solution

Initialization: $\mu_1^{(0)} = 0$ and $\mu_1^{(0)} = 5$

$t = 1$:

Assign clusters: $z_1^{(1)} \leftarrow 1, z_2^{(1)} \leftarrow 1, z_3^{(1)} \leftarrow 1, z_4^{(1)} \leftarrow 2, z_5^{(1)} \leftarrow 2$

Update centroids: $\mu_1^{(1)} = \frac{1}{3}(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3) = \frac{2}{3}, \;\; \mu_2^{(1)} = \frac{1}{2}(\mathbf{x}_4 + \mathbf{x}_5) = 4$

Convergence takes place

$t = 2$:

Assign clusters: $z_1^{(2)} \leftarrow 1, z_2^{(2)} \leftarrow 1, z_3^{(2)} \leftarrow 1, z_4^{(2)} \leftarrow 2, z_5^{(2)} \leftarrow 2$

Update centroids: $\mu_1^{(2)} = \frac{1}{3}(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3) = \frac{2}{3}, \;\; \mu_2^{(2)} = \frac{1}{2}(\mathbf{x}_4 + \mathbf{x}_5) = 4$

Centroids: $\mu_1 = \frac{2}{3} \approx 0.66, \;\; \mu_2 = 4$

Assigned clusters: $z_1 = 1, z_2 = 1, z_3 = 1, z_4 = 2, z_5 = 2$

Next in Agenda

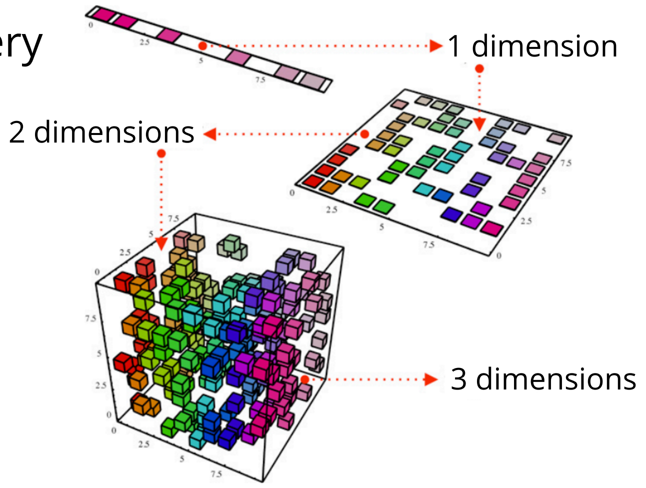
Exam 2019

Question 3	Convolutional Neural Networks
Question 5	Clustering
Question 6	Dimensionality Reduction
Question 7	Linear Discriminant Analysis
Question 8	Gaussian Mixture Models and EM Algorithm

Recap: Dimensionality Reduction

Suppose $x_i \in \mathbb{R}^d, i \in \{1, \cdots, n\}$ and we want to learn a mapping $f : \mathbb{R}^d \to \mathbb{R}^k$ with $k << d$ where we can reconstruct the data with little loss of information

Motivation: Visualization, compression, regularization, unsupervised feature discovery



Key question: How to choose the mapping f?

You have seen so far:

- Principal Component Analysis
- Kernel PCA
- Neural Network Encoders

image credit: https://bigsnarf.wordpress.com/

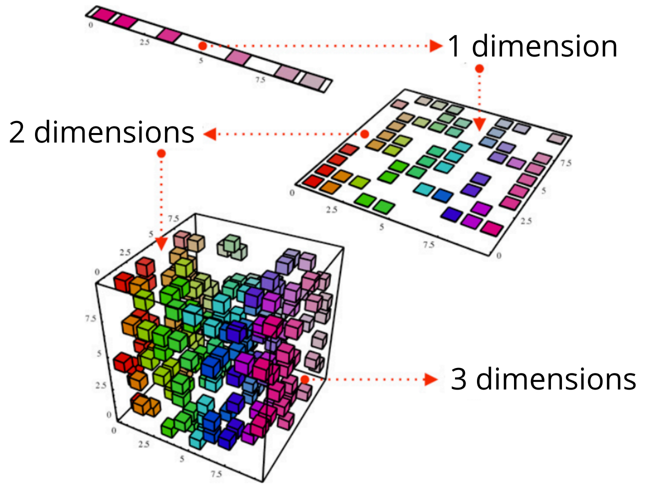
Recap: Principle Component Analysis

Suppose $x_i \in \mathbb{R}^d, i \in \{1, \cdots, n\}$ and we want to learn a mapping $f : \mathbb{R}^d \to \mathbb{R}^k$ with $k << d$ where we can reconstruct the data with little loss of information

Motivation: Visualization, compression, unsupervised feature discovery

Key question: How to choose the mapping f?

Linear dimensionality reduction: Principal Component Analysis (PCA)



Recall from the lecture that PCA is a linear dimensionality reduction technique

$$\mathbf{z}_i = \mathbf{W}^T \mathbf{x}_i, \quad \mathbf{W} \in \mathbb{R}^{d \times k}$$

which minimizes the reconstruction error $\min_{\mathbf{W}, \mathbf{z}_i} \sum_i \|\mathbf{W}\mathbf{z}_i - \mathbf{x}_i\|_2^2$ for orthogonal **W**

Solution to PCA. For centered data: $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$

$$\mathbf{W}^* = (\mathbf{v}_1 | \cdots | \mathbf{v}_k) \text{ and } \mathbf{z}_i = (\mathbf{W}^*)^T \mathbf{x}_i \text{ where } \Sigma = \sum_{i=1}^{d} \lambda_i \mathbf{v}_i \mathbf{v}_i^T, \ \lambda_1 \geq \cdots \geq \lambda_d \geq 0$$

image credit: https://bigsnarf.wordpress.com/

Exam 2019 - 6.i-iii

In dimension reduction, we want to embed high dimensional data from a space $\mathbb{R}^d$ to a space $\mathbb{R}^k$ with $k \ll d$. We therefore want to represent the data $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d}$ with the respective lower-dimensional embeddings $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_n]^\top \in \mathbb{R}^{n \times k}$. One particular method for dimensionality reduction is the *Principal Component Analysis* (PCA), where the embeddings are given by $\mathbf{z}_i = \mathbf{W}^\top \mathbf{x}_i$, where $\mathbf{W} \in \mathbb{R}^{d \times k}$.

(i) *PCA general (T/F)*
Determine whether the following statements about PCA are true or false.

True	**False**	
☒	☐	PCA is a linear dimension reduction method.
☐	☒	The first principal component is the eigenvector of the data covariance matrix with the smallest eigenvalue.
☒	☐	All principal components are mutually orthogonal.
☐	☒	PCA is regarded as a supervised learning technique.

Solution to i

1. We want $\mathbf{W}\mathbf{z}_i \approx \mathbf{x}_i \quad \mathbf{W} \in \mathbb{R}^{d \times k}$

2. That does not minimize $\sum_i \|\mathbf{W}\mathbf{z}_i - \mathbf{x}_i\|_2^2$

3. Due to orthonormality of $\mathbf{v}_k$'s, we have $\mathbf{v}_k^T \mathbf{v}_l = 1[k = l]$

4. Unsupervised: no labels

(ii) *PCA general II (T/F)*
Suppose $k = 1$, and hence $\mathbf{W}$ becomes a column vector $\mathbf{w}$. Which of these objective functions is a valid PCA objective:

True	**False**	
☒	☐	$\arg\min_{\mathbf{w}, \|\mathbf{w}\|_2^2 = 1, \{z_i\}_{i=1}^n} \sum_{i=1}^n \|\mathbf{w} z_i - \mathbf{x}_i\|_2^2$
☒	☐	$\arg\min_{\mathbf{w}, \|\mathbf{w}\|_2^2 = 1} \sum_{i=1}^n \left\|\mathbf{w}\mathbf{w}^\top \mathbf{x}_i - \mathbf{x}_i\right\|_2^2$
☐	☒	$\arg\min_{\mathbf{w}, \|\mathbf{w}\|_2^2 = 1} \sum_{i=1}^n \left\|\mathbf{w}^\top \mathbf{x}_i - \mathbf{w}^\top \left(\frac{1}{n^2} \sum_{i=1}^n \mathbf{x}_i\right)\right\|_2^2$
☐	☒	$\arg\min_{\mathbf{w}, \{z_i\}_{i=1}^n} \frac{1}{n} \sum_{i=1}^n \|\mathbf{w} z_i - \mathbf{x}_i\|_2^2$

Solution to ii

For $k = 1$ PCA optimizes for $\min_{\mathbf{w}, \|\mathbf{w}\|_2 = 1, z_{i \in [n]}} \sum_i \|\mathbf{w} z_i - \mathbf{x}_i\|_2^2$

Towards solving, we jointly optimize for $(\mathbf{w}^*, \mathbf{z}^*) = \arg\min_{\mathbf{w}, \|\mathbf{w}\|_2 = 1, \mathbf{z}} \sum_i \|\mathbf{w} z_i - \mathbf{x}_i\|_2^2$

For a fix $\mathbf{w}$, the optimal $\mathbf{z} : \mathbf{z}^*$ is given by $\mathbf{z}_i^* = \mathbf{w}^T \mathbf{x}_i$. Hence

$$\mathbf{w}^* = \arg\min_{\mathbf{w}: \|\mathbf{w}\|_2 = 1} \sum_i \|\mathbf{w}\mathbf{w}^T \mathbf{x}_i - \mathbf{x}_i\|_2^2$$

(iii) *PCA general III (T/F)*
When $k > 1$, the matrix $\mathbf{W}$

True	**False**	
☒	☐	$\mathbf{W}^\top \mathbf{W}$ is the identity
☐	☒	$\mathbf{W}$ is symmetric
☐	☒	$\mathbf{W}$ is diagonal
☐	☒	$\mathbf{W}\mathbf{W}^\top$ is the identity

Solution to iii

Recall that $\mathbf{W} = (\mathbf{v}_1 | \cdots | \mathbf{v}_k)$ where $\Sigma = \sum_{i=1}^d \lambda_i \mathbf{v}_i \mathbf{v}_i^T, \quad \lambda_1 \geq \cdots \geq \lambda_d \geq 0$

We have the followings:

$(\mathbf{W}\mathbf{W}^T)_{i,j} = \sum_{m=[k]} (\mathbf{v}_m)_i (\mathbf{v}_m)_j$ and $(\mathbf{W}^T\mathbf{W})_{i,j} = \mathbf{v}_i^T \mathbf{v}_j$

Therefore it only holds that $\mathbf{W}^T\mathbf{W}$ is identity.
(Orthonormality of eigenvectors implies the identity matrix)

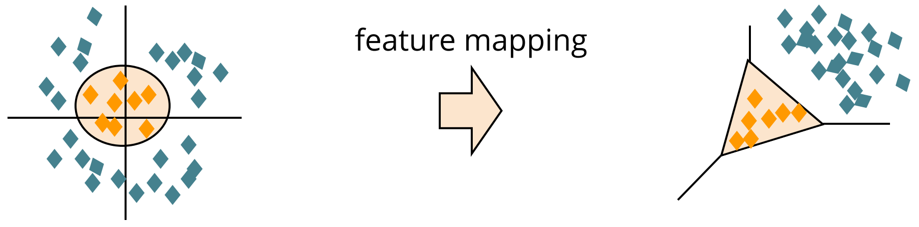
Recap: Kernel PCA

Motivation. How to capture non-linear manifold structures?

Kernel PCA. Apply Kernel method to PCA! $\boxed{k(\mathbf{x},\mathbf{z}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T(z_1^2, \sqrt{2}z_1z_2, z_2^2) = (\mathbf{x}^T\mathbf{z})^2}$

Map data to higher dimensions where contain linear patterns: Data becomes linearly separable in the new feature space

Example. Feature mapping function $\phi : \mathbb{R}^2 \to \mathbb{R}^3 \quad (x_1, x_2) \to (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$



Data in low dimensional space

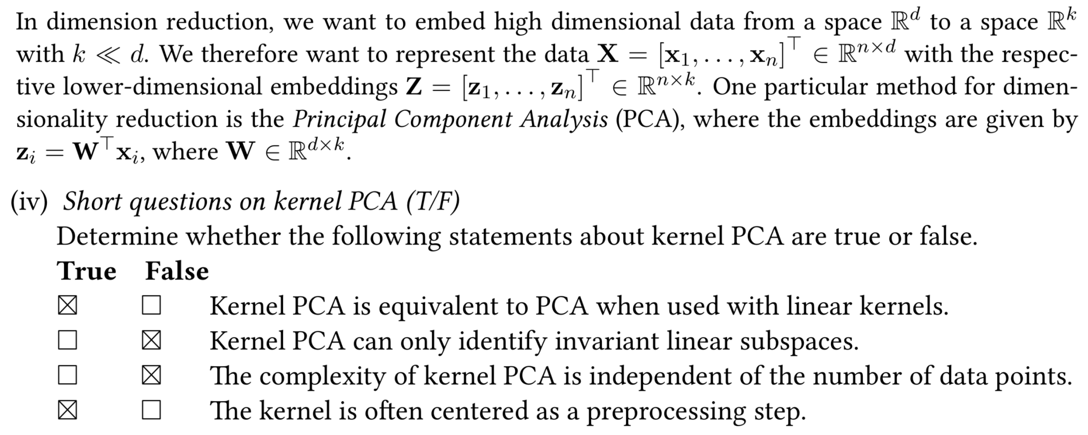
Data in high dimensional space

The feature mapping ϕ is not necessary to know! We deal with kernel functions instead

Recall from the class that kernel principal components $\alpha^{(1)}, \cdots, \alpha^{(k)} \in \mathbb{R}^n$ are given by $\alpha^{(i)} = \frac{1}{\sqrt{\lambda_i}}\mathbf{v}_i$ where $\lambda_i, \mathbf{v}_i, i = \{1, \cdots, n\}$ are obtained by eigendecomposition of $\mathbf{K} = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \mathbf{v}_i^T$

A new point x is projected as $\boxed{z_i = \sum_{j=1}^{n} \alpha_j^{(i)} k(x, x_j)}$

Exam 2019 - 6.iv



In dimension reduction, we want to embed high dimensional data from a space $\mathbb{R}^d$ to a space $\mathbb{R}^k$ with $k \ll d$. We therefore want to represent the data $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d}$ with the respective lower-dimensional embeddings $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_n]^\top \in \mathbb{R}^{n \times k}$. One particular method for dimensionality reduction is the *Principal Component Analysis* (PCA), where the embeddings are given by $\mathbf{z}_i = \mathbf{W}^\top \mathbf{x}_i$, where $\mathbf{W} \in \mathbb{R}^{d \times k}$.

(iv) *Short questions on kernel PCA (T/F)*

Determine whether the following statements about kernel PCA are true or false.

True	**False**	
☒	☐	Kernel PCA is equivalent to PCA when used with linear kernels.
☐	☒	Kernel PCA can only identify invariant linear subspaces.
☐	☒	The complexity of kernel PCA is independent of the number of data points.
☒	☐	The kernel is often centered as a preprocessing step.

Solution to iv

1. $\mathbf{K} = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \mathbf{v}_i^T$

2. Kernel PCA can $\underline{\text{also}}$ be used to identify invariant linear subspaces with the use of nonlinear mappings ϕ

3. Taking eigenvalue decomposition of the kernel matrix $\mathbf{K}$ and computing $\alpha^{(i)}$ leads to a complexity that grows with number of points

4. Analogue linear PCA to Kernel one by setting empirical mean to 0

Recap: Autoencoders

Use neural network autoencoders to learn the nonlinear mapping for dimensionality reduction through an identity function

$$x \approx f(x;\theta)$$

Properties of $f(\cdot)$: approximates the identitity function & performs compression

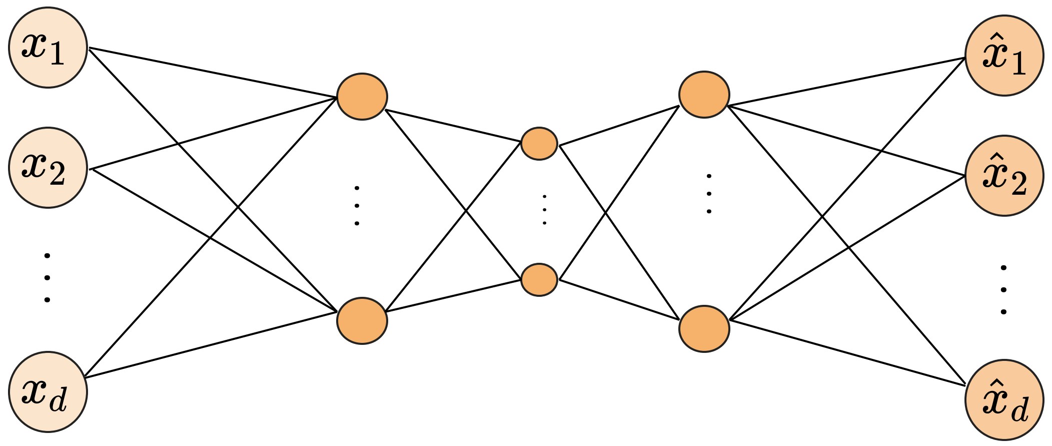
How to pick $f(\cdot)$: Composition of two nonlinear functions $f_1(\cdot)$ and $f_2(\cdot)$ such that

$\boxed{f(x;\theta) = f_2(f_1(x;\theta_1);\theta_2)}$ where $\underbrace{f_1(\cdot) : \mathbb{R}^d \to \mathbb{R}^k}_{\text{encoder}}$ and $\underbrace{f_2(\cdot) : \mathbb{R}^k \to \mathbb{R}^d}_{\text{decoder}}$

How to learn $f_1(\cdot)$ and $f_2(\cdot)$? Use Neural Networks!

Non-linear generalization of PCA.

Recap: Autoencoders



$f_1 = F_1 \circ \cdots \circ F_l : \mathbb{R}^d \to \mathbb{R}^k, \mathbf{x} \to \mathbf{z}$

encoder

$f_2 = F_L \circ \cdots \circ F_{l+1} : \mathbb{R}^k \to \mathbb{R}^d, \mathbf{z} \to \hat{\mathbf{x}}$

decoder

How to train autoencoders?

Optimize the weights such that $\boxed{\hat{\mathbf{x}} = f(\mathbf{x}; \mathbf{w}^{(1)}, \mathbf{w}^{(2)}) = f_2(f_1(\mathbf{x}; \mathbf{w}^{(1)}); \mathbf{w}^{(2)}) \approx \mathbf{x}}$ e.g., $\boxed{\min_{\mathbf{W}} \sum_{i=1}^{n} \|\mathbf{x}_i - f(\mathbf{x}_i; \mathbf{W})\|_2^2}$ via backpropagation.

Autoencoders vs. PCA

original autoencoder PCA

image credit: http://nghiaho.com

See js demo for digit images:

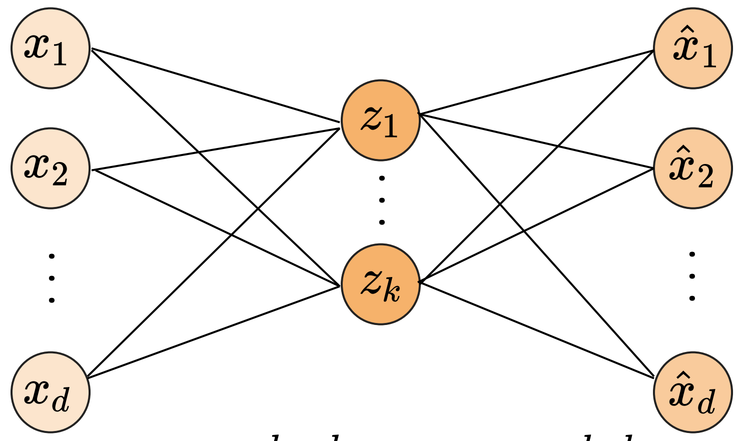
https://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencod

.html

Recap: Autoencoders

Given data points $\mathbf{x}_i \in \mathbb{R}^d, i = 1, \cdots, n$ compress data into k-dimensional representation $k \leq d$.

Linear auto-encoding with a single hidden layer



How to choose E and D?

Optimal solution satisfies: $\boxed{\min \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{D}\mathbf{E}\mathbf{x}_i\|_2^2}$

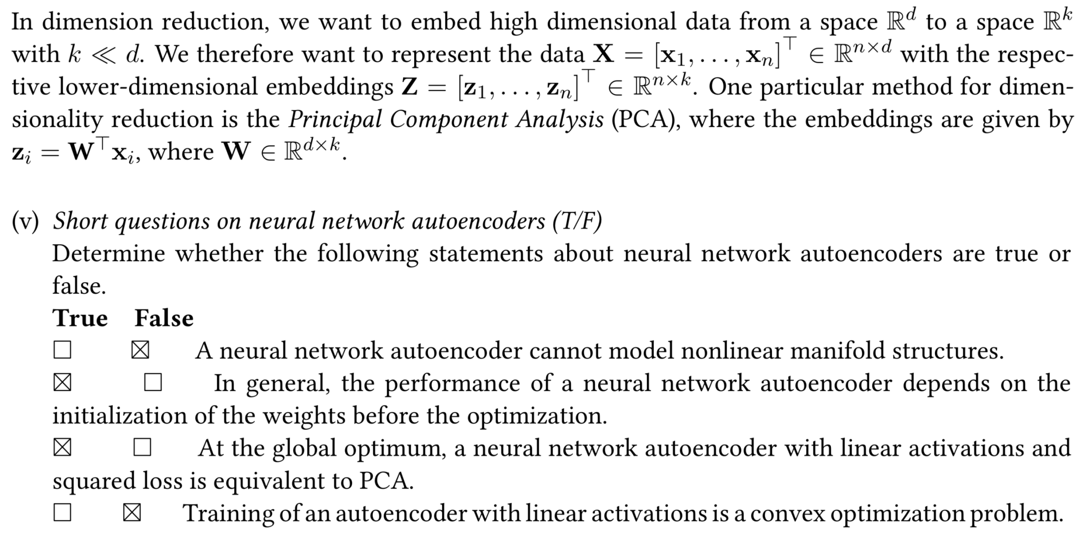
$\boxed{\mathbf{E} = \mathbf{U}_k^T}$ $E \in \mathbb{R}^{k \times d}$ ***encoder*** $D \in \mathbb{R}^{d \times k}$ ***decoder*** $\boxed{\mathbf{D} = \mathbf{U}_k}$ $\longrightarrow$ $\boxed{\mathbf{D}\mathbf{E}\mathbf{X} = \mathbf{U}_k \Lambda_k \mathbf{V}_k^T}$

Eckart-Young theorem: Let $\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ and SVD of $\mathbf{X} = \mathbf{U}\Lambda\mathbf{V}$. For $k \leq \min(n, d)$

$$\underset{\hat{\mathbf{X}}:\text{rank}(\hat{\mathbf{X}})=\mathbf{k}}{\arg\min} \|\mathbf{X} - \hat{\mathbf{X}}\|_F^2 = \mathbf{U}_k \Lambda_k \mathbf{V}_k^T$$

PCA

Exam 2019 - 6.v


In dimension reduction, we want to embed high dimensional data from a space $\mathbb{R}^d$ to a space $\mathbb{R}^k$ with $k \ll d$. We therefore want to represent the data $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d}$ with the respective lower-dimensional embeddings $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_n]^\top \in \mathbb{R}^{n \times k}$. One particular method for dimensionality reduction is the *Principal Component Analysis* (PCA), where the embeddings are given by $\mathbf{z}_i = \mathbf{W}^\top \mathbf{x}_i$, where $\mathbf{W} \in \mathbb{R}^{d \times k}$.

(v) *Short questions on neural network autoencoders (T/F)*
Determine whether the following statements about neural network autoencoders are true or false.

True	False	
☐	☒	A neural network autoencoder cannot model nonlinear manifold structures.
☒	☐	In general, the performance of a neural network autoencoder depends on the initialization of the weights before the optimization.
☒	☐	At the global optimum, a neural network autoencoder with linear activations and squared loss is equivalent to PCA.
☐	☒	Training of an autoencoder with linear activations is a convex optimization problem.

Solution to v

1. A neural network autoencoder can model nonlinear manifold structures with use of nonlinear activation functions

2. Due to the non-convex objective, initialization matters

3. Eckart-Young Theorem

4. Non-convex due to the dimensionality reduction constraint

Next in Agenda

Exam 2019

Question 3	Convolutional Neural Networks
Question 5	Clustering
Question 6	Dimensionality Reduction
Question 7	Linear Discriminant Analysis
Question 8	Gaussian Mixture Models and EM Algorithm

Recap: Discriminative vs. Generative Modeling

- Discriminative models estimate class conditional probabilities $P(y|\mathbf{x}) = \frac{P(y, \mathbf{x})}{P(x)}$
- Generative models estimate joint distribution $P(y, \mathbf{x}) = P(\mathbf{x})P(y|\mathbf{x})$

Approach to generative modeling

- Estimate the distribution of class labels $P(y)$
- Estimate the conditional distribution for each class y $P(\mathbf{x}|y)$
- Obtain predictive distribution using Bayes' rule

$$P(y|\mathbf{x}) = \frac{1}{P(x)} P(y)P(\mathbf{x}|y)$$

Example Naive Bayes Classifier

- Form distribution on class labels from categorical variables $P(Y = y) = p_y$
- Features are conditionally independent given class label

$$P(X_1 = x_1, \ldots, X_n = x_n | Y = y) = \prod_{i=1}^{n} P(X_I = x_i | Y = y)$$

Exam 2019 - 7.i-ii

In this problem we would like to perform classification using linear discriminant analysis (LDA). We assume the following simplified model,

$$P(X|Y=j) = \mathcal{N}(\mu_j, \mathbf{I})$$

where $\mathbf{I} \in \mathbb{R}^{d\times d}$ is an identity matrix and $j \in \{1,2\}$. Furthermore, we parametrize $P(Y=1)=p$, and naturally $P(Y=2)=1-p$.

Consider a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where n_j represents the number of samples such that $y=j$.

(i) *MLE Probability (Num)*
Assuming that $n_1 = 1$ and $n_2 = 3$ calculate the MLE estimate of p, $\hat{p}$.

Solution to i

Class labels $\mathcal{Y} = \{1,2\}$ with probabilities

$$P(Y=1)=p \quad \text{and} \quad P(Y=2)=1-p$$

Conditional distribution of X given a class label $P(X|Y=j) = \mathcal{N}(\mu_j, \mathbf{I})$

Gaussian Naive Bayes classifier

Estimate parameters of $P(Y)$, i.e., p, using $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^4$ via Maximum Likelihood Estimation (MLE):

$$\hat{p} = \arg\max_{p'} P(\mathcal{D}|p')$$

$P(\mathcal{D}|p') = \prod_{i=1}^4 (p')^{1\{y_i=1\}}(1-p')^{1\{y_i=2\}} = (p')^{n_1}(1-p')^{n_2} = p'(1-p')^3$

$P(\mathcal{D}|p')$ is maximized when derivative is 0: $(1-p')^3 - 3p'(1-p')^2 = 0$

Note that this happens at $p' = 0.25$. Hence the estimate of *p* is given by 0.25

Summary: $\hat{P}(Y=y) = \frac{\text{Count}(Y=y)}{n}$

(ii) *MAP Probability (Num)*
An expert on the problem at hand has told us that he is sure that $p=0.5$ or $p=0.25$ with equal probability. Assuming this as a prior belief, calculate the maximum a-posteriori estimate of p, $\hat{p}$:

Solution to ii

- Calculate two posteriors for p=0.25 and p=0.5
- Then choose the one that maximizes $P(p'|\mathcal{D}) \propto P(p')P(\mathcal{D}|p')$
- Given $P(p') = 1/2$ for both p', $\hat{p}_{\text{MAP}} = 0.25$

Final Exam 2019 - 7.iii

In this problem we would like to perform classification using linear discriminant analysis (LDA). We assume the following simplified model,

$$P(X|Y=j) = \mathcal{N}(\mu_j, \mathbf{I})$$

where $\mathbf{I} \in \mathbb{R}^{d\times d}$ is an identity matrix and $j \in \{1,2\}$. Furthermore, we parametrize $P(Y=1)=p$, and naturally $P(Y=2)=1-p$.

Consider a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where n_j represents the number of samples such that $y=j$.

(iii) *MLE II (MC)*

Which of the following expresses the MLE estimate $\hat{\mu}_1$ of μ_1?

☒ $\hat{\mu}_1 = \frac{1}{n_1}\sum_{i,\text{s.t.}y_i=1} \mathbf{x}_i$
☐ $\hat{\mu}_1 = \frac{n_1}{n_1+n_2}\sum_{i,\text{s.t.}y_i=1} \mathbf{x}_i$
☐ $\hat{\mu}_1 = \frac{n_1}{n_2}\sum_{i,\text{s.t.}y_i=1} \mathbf{x}_i$
☐ Cannot be derived from the information given.

Solution to iii

Class labels $\mathcal{Y} = \{1,2\}$ with probabilities

$$P(Y=1)=p \quad \text{and} \quad P(Y=2)=1-p$$

Conditional distribution of X given a class label $P(X|Y=j) = \mathcal{N}(\mu_j, \mathbf{I})$

Estimate parameters of $P(X|Y)$ using $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ via Maximum Likelihood Estimation (MLE):

$$\begin{aligned}
\hat{\mu}_j &= \arg\max_{\mu'_j} P(\mathcal{D}|Y=j) = \arg\min_{\mu'_j} -\log P(\mathcal{D}|Y=j) \\
&= \arg\min_{\mu'_j} -\sum_{i:Y_i=j} \log P(\mathbf{x}_i|Y=j) \\
&= \arg\min_{\mu'_j} \sum_{i:Y_i=j} -\log \frac{1}{(2\pi)^{d/2}\sqrt{\det(\Sigma)}} \exp\left(-\tfrac{1}{2}(\mathbf{x}_i-\mu'_j)^T\Sigma^{-1}(\mathbf{x}_i-\mu'_j)\right) \\
&= \arg\min_{\mu'_j} \sum_{i:Y_i=j} (\mathbf{x}_i-\mu'_j)^T(\mathbf{x}_i-\mu'_j)
\end{aligned}$$

Remember from the class that $\hat{\mu}_j = \frac{1}{\text{Count}(Y_i=j)}\sum_{i:Y_i=j} \mathbf{x}_i$

Summary: $\hat{\mu} = \frac{1}{\text{Count}(Y=y)}\sum_{i:Y_i=y} \mathbf{x}_i$

$$\hat{\Sigma} = \frac{1}{\text{Count}(Y=y)}\sum_{i:Y_i=y} (\mathbf{x}_i-\mu_y)^T(\mathbf{x}_i-\mu_y)$$

Exam 2019 - 7.iv

In this problem we would like to perform classification using linear discriminant analysis (LDA). We assume the following simplified model,

$$P(X|Y=j) = \mathcal{N}(\mu_j, \mathbf{I})$$

where $\mathbf{I} \in \mathbb{R}^{d \times d}$ is an identity matrix and $j \in \{1, 2\}$. Furthermore, we parametrize $P(Y = 1) = p$, and naturally $P(Y = 2) = 1 - p$.

Consider a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ where n_j represents the number of samples such that $y = j$.

(iv) *Decision rule (MC)*
Suppose that we have the estimates of $\hat{p}_j$ and $\hat{\mu}_j$ for all j. The LDA provides a decision rule that given an $\mathbf{x}$, it can predict its class. Which of the following is a valid procedure to predict the class label of $\mathbf{x}$ using LDA.

Comment: $\hat{p}_1 = \hat{p}$ and $\hat{p}_2 = 1 - \hat{p}$.

☐ $\hat{y} = \arg\max_{j \in \{1,2\}} \exp(-\frac{\hat{p}_j ||\mathbf{x} - \hat{\mu}_j||_2^2}{2})$
☐ $\hat{y} = \arg\max_{j \in \{1,2\}} (\mathbf{x} - \hat{\mu}_j)^\top \hat{\mu}_j + \hat{p}_j$
☒ $\hat{y} = \arg\max_{j \in \{1,2\}} (2\mathbf{x} - \hat{\mu}_j)^\top \hat{\mu}_j + 2\log(\hat{p}_j)$
☐ $\hat{y} = \arg\max_{j \in \{1,2\}} \exp(-\frac{||\mathbf{x} - \hat{\mu}_j||_2^2}{2\hat{p}_j})$
☐ $\hat{y} = \arg\max_{j \in \{1,2\}} \exp(-\frac{||\mathbf{x} - \hat{\mu}_j||_2^2}{\hat{p}_j})$

Approach to generative modeling

- Estimate the distribution of class labels $P(y)$
- Estimate the conditional distribution for each class y $P(\mathbf{x}|y)$
- Obtain predictive distribution using Bayes' rule

$$P(y|\mathbf{x}) = \frac{1}{P(x)} P(y) P(\mathbf{x}|y)$$

$y = \arg\max_{y'} P(y'|\mathbf{x})$ minimizes the misclassification error

$y = \arg\max_{j} \log P(Y=j|\mathbf{x})$ also minimizes the misclassification error

$$\begin{aligned}
\log P(Y=j|\mathbf{x}) &= \log\left(\tfrac{1}{P(\mathbf{x})} P(Y=j) \prod_{i=1}^{d} P(x_i|Y=j)\right) \\
&= \log \tfrac{1}{P(\mathbf{x})} + \log P(Y=j) + \sum_{i=1}^{d} \log P(x_i|Y=j) \\
&= \log \tfrac{1}{P(\mathbf{x})} + \log P(Y=j) + \sum_{i=1}^{d} \log \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp(-\tfrac{1}{2\sigma_i^2}(x_i - \mu_{j,i})^2)
\end{aligned}$$

$$\begin{aligned}
y = \arg\max_{j} \log P(Y=j|\mathbf{x}) &= \arg\max_{j}\ \log p_j - \tfrac{1}{2}\sum_{i=1}^{d}(x_i - \mu_{j,i})^2 \\
&= \arg\max_{j}\ \log p_j + \tfrac{1}{2}\sum_{i=1}^{d}(2x_i\mu_{j,i} - \mu_{j,i}^2) \\
&= \arg\max_{j}\ 2\log p_j + (2\mathbf{x} - \hat{\mu}_j)^T \hat{\mu}_j
\end{aligned}$$

Exam 2019 - 7.v

In this problem we would like to perform classification using linear discriminant analysis (LDA). We assume the following simplified model,

$$P(X|Y=j) = \mathcal{N}(\mu_j, \mathbf{I})$$

where $\mathbf{I} \in \mathbb{R}^{d\times d}$ is an identity matrix and $j \in \{1,2\}$. Furthermore, we parametrize $P(Y=1)=p$, and naturally $P(Y=2)=1-p$.

Consider a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where n_j represents the number of samples such that $y=j$.

(v) *Decision rule II (MC)*
If $\mathbf{x}^\top(\hat\mu_1 - \hat\mu_2) > \frac{1}{2}\left(\hat\mu_1{}^\top\hat\mu_1 - \hat\mu_2^\top\hat\mu_2\right)$, $\mathbf{x}$ is classified as
☐ 1 for any $\hat p$
☐ 2 for any $\hat p$
☐ 2 if $\hat p = 0.5$
☒ 1 if $\hat p = 0.5$

Solution to v

$y = \arg\max_j \log P(Y=j|\mathbf{x})$ minimizes the misclassification error

$$\log P(Y=j|\mathbf{x}) = \log\left(\tfrac{1}{P(\mathbf{x})} P(Y=j) \prod_{i=1}^d P(x_i|Y=j)\right)$$
$$= \log\tfrac{1}{P(\mathbf{x})} + \log P(Y=j) + \sum_{i=1}^d \log P(x_i|Y=j)$$
$$= \log\tfrac{1}{P(\mathbf{x})} + \log P(Y=j) + \sum_{i=1}^d \log \tfrac{1}{\sqrt{2\pi\sigma_i^2}} \exp(-\tfrac{1}{2\sigma_i^2}(x_i - \mu_{j,i})^2)$$

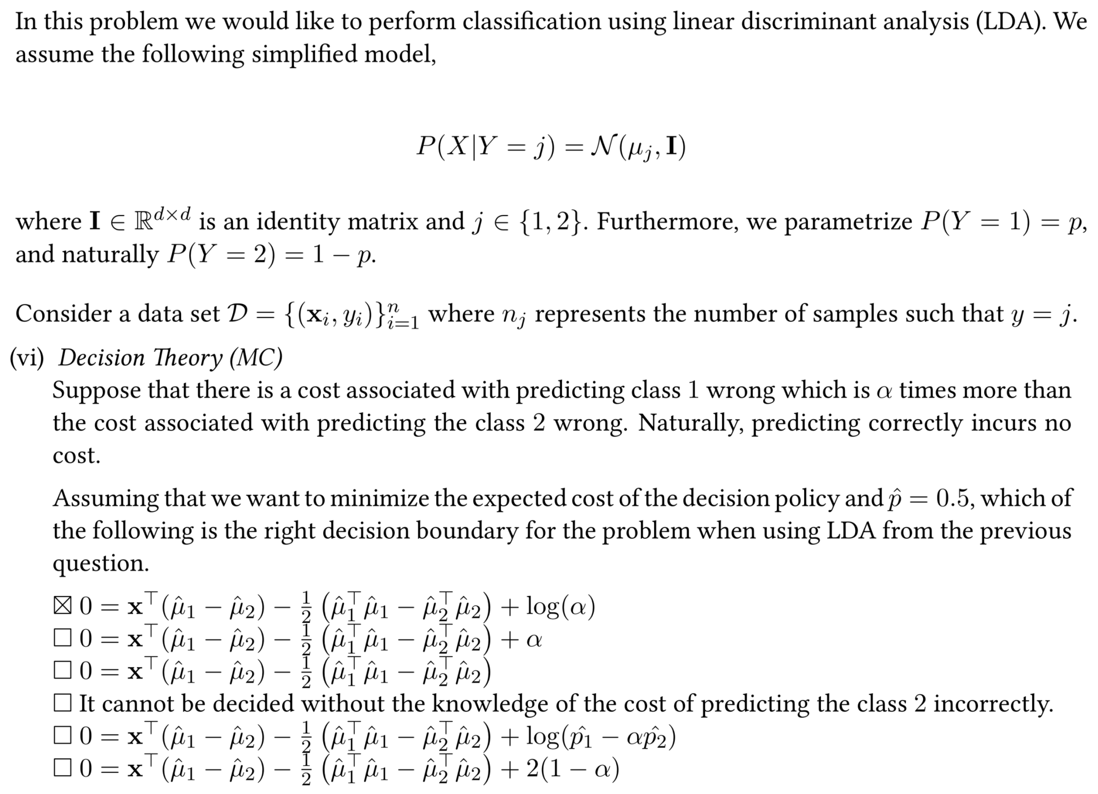
$$y = \arg\max_j \log P(Y=j|\mathbf{x}) = \arg\max_j \ \log p_j - \tfrac{1}{2}\sum_{i=1}^d (x_i - \mu_{j,i})^2$$
$$= \arg\max_j \ \log p_j + \tfrac{1}{2}\sum_{i=1}^d (2x_i\mu_{j,i} - \mu_{j,i}^2)$$
$$= \arg\max_j \ 2\log p_j + (2\mathbf{x} - \hat\mu_j)^T\hat\mu_j$$

It is clear that:

If $2\log p_1 + (2\mathbf{x} - \hat\mu_1)^T\hat\mu_1 > 2\log p_2 + (2\mathbf{x} - \hat\mu_2)^T\hat\mu_2$ then $\mathbf{x}$ is classified as 1
else it is classified as 0.

For $\hat p = 0.5$, the decision rule can be re-written as:
If $2\mathbf{x}^T\hat\mu_1 - \hat\mu_1^T\hat\mu_1 > 2\mathbf{x}^T\hat\mu_2 - \hat\mu_2^T\hat\mu_2$ then $\mathbf{x}$ is classified as 1, else as 0.
In other words, if $2\mathbf{x}^T(\hat\mu_1 - \hat\mu_2) > \hat\mu_1^T\hat\mu_1 - \hat\mu_2^T\hat\mu_2$ then $\mathbf{x}$ is classified as 1, else as 0.
We finally arrive at the solution by dividing each side by 2.

Exam 2019 - 7.vi



In this problem we would like to perform classification using linear discriminant analysis (LDA). We assume the following simplified model,

$$P(X|Y=j) = \mathcal{N}(\mu_j, \mathbf{I})$$

where $\mathbf{I} \in \mathbb{R}^{d\times d}$ is an identity matrix and $j \in \{1,2\}$. Furthermore, we parametrize $P(Y=1)=p$, and naturally $P(Y=2)=1-p$.

Consider a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where n_j represents the number of samples such that $y=j$.

(vi) *Decision Theory (MC)*

Suppose that there is a cost associated with predicting class 1 wrong which is α times more than the cost associated with predicting the class 2 wrong. Naturally, predicting correctly incurs no cost.

Assuming that we want to minimize the expected cost of the decision policy and $\hat{p}=0.5$, which of the following is the right decision boundary for the problem when using LDA from the previous question.

☒ $0 = \mathbf{x}^\top(\hat{\mu}_1 - \hat{\mu}_2) - \frac{1}{2}\left(\hat{\mu}_1^\top\hat{\mu}_1 - \hat{\mu}_2^\top\hat{\mu}_2\right) + \log(\alpha)$
☐ $0 = \mathbf{x}^\top(\hat{\mu}_1 - \hat{\mu}_2) - \frac{1}{2}\left(\hat{\mu}_1^\top\hat{\mu}_1 - \hat{\mu}_2^\top\hat{\mu}_2\right) + \alpha$
☐ $0 = \mathbf{x}^\top(\hat{\mu}_1 - \hat{\mu}_2) - \frac{1}{2}\left(\hat{\mu}_1^\top\hat{\mu}_1 - \hat{\mu}_2^\top\hat{\mu}_2\right)$
☐ It cannot be decided without the knowledge of the cost of predicting the class 2 incorrectly.
☐ $0 = \mathbf{x}^\top(\hat{\mu}_1 - \hat{\mu}_2) - \frac{1}{2}\left(\hat{\mu}_1^\top\hat{\mu}_1 - \hat{\mu}_2^\top\hat{\mu}_2\right) + \log(\hat{p_1} - \alpha\hat{p_2})$
☐ $0 = \mathbf{x}^\top(\hat{\mu}_1 - \hat{\mu}_2) - \frac{1}{2}\left(\hat{\mu}_1^\top\hat{\mu}_1 - \hat{\mu}_2^\top\hat{\mu}_2\right) + 2(1-\alpha)$

Solution to vi

We are given:

- Predictive distribution $P(Y=y|\mathbf{x})$
- Set of actions $\mathcal{A}$
- Cost function to penalize our actions $C: \mathcal{Y}\times\mathcal{A} \to \mathbb{R}$

Task: Predict the label given x where cost of actions are different, formally

$$C(a=2, Y=1) = \alpha k \text{ and } C(a=1, Y=2) = k$$

Minimize the expected cost $a^* = \arg\min_{a\in\mathcal{A}} \mathbb{E}[C(y,a)|\mathbf{x}]$
where $\mathbb{E}[C(y,a)|\mathbf{x}] = \sum_{y\in\mathcal{Y}} P(Y=y|\mathbf{x})C(y,a)$

If $a=1$ then $\mathbb{E}[C(y,a=1)|\mathbf{x}] = P(Y=1|\mathbf{x})C(y=1,a=1) + P(Y=2|\mathbf{x})C(y=2,a=1)$
$= P(Y=2|\mathbf{x})k$

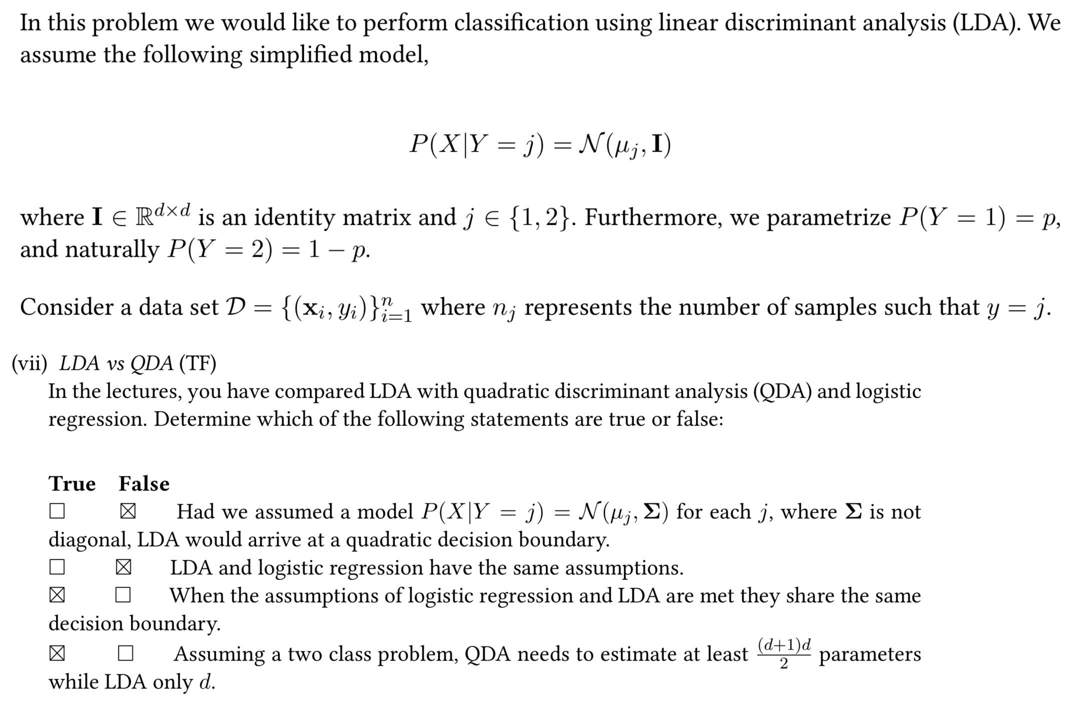
Else ($a=2$), $\mathbb{E}[C(y,a=2)|\mathbf{x}] = P(Y=1|\mathbf{x})C(y=1,a=2) + P(Y=2|\mathbf{x})C(y=2,a=2)$
$= P(Y=1|\mathbf{x})\alpha k$

We can write down the decision rule as follows:

If $P(Y=1|\mathbf{x})\alpha k > P(Y=2|\mathbf{x})k$ then choose action 1, else choose action 2.
Taking logarithm of each side and incorporating the derivation of $P(Y=y|\mathbf{x}),\ y=1,2$ from the previous question, we can write down the decision boundary as:

$$\mathbf{x}^T(\hat{\mu_1} - \hat{\mu_2}) - \frac{1}{2}(\hat{\mu}_1^T\hat{\mu}_1 - \hat{\mu}_2^T\hat{\mu}_2) + \log\alpha = 0$$

Exam 2019 - 7.vii


In this problem we would like to perform classification using linear discriminant analysis (LDA). We assume the following simplified model,

$$P(X|Y=j) = \mathcal{N}(\mu_j, \mathbf{I})$$

where $\mathbf{I} \in \mathbb{R}^{d\times d}$ is an identity matrix and $j \in \{1, 2\}$. Furthermore, we parametrize $P(Y=1) = p$, and naturally $P(Y=2) = 1-p$.

Consider a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ where n_j represents the number of samples such that $y = j$.

(vii) *LDA vs QDA* (TF)
In the lectures, you have compared LDA with quadratic discriminant analysis (QDA) and logistic regression. Determine which of the following statements are true or false:

True False
☐ ☒ Had we assumed a model $P(X|Y=j) = \mathcal{N}(\mu_j, \mathbf{\Sigma})$ for each j, where $\mathbf{\Sigma}$ is not diagonal, LDA would arrive at a quadratic decision boundary.
☐ ☒ LDA and logistic regression have the same assumptions.
☒ ☐ When the assumptions of logistic regression and LDA are met they share the same decision boundary.
☒ ☐ Assuming a two class problem, QDA needs to estimate at least $\frac{(d+1)d}{2}$ parameters while LDA only d.

Solution to vii

1. Shared variance leads to a linear decision boundary

2. LDA assumes: Shared variance, density is Gaussian, balanced classes

3. Gaussian Naive Bayes model with constant variance uses the discriminant:

$$f(\mathbf{x}) = \log \frac{P(Y=1|\mathbf{x})}{P(Y=-1|\mathbf{x})} = \mathbf{w}^T\mathbf{x} + w_0$$

where $\mathbf{w} = \hat{\mathbf{\Sigma}}^{-1}((\hat{\mu}_+ - \hat{\mu}_-)$ and $w_0 = \frac{1}{2}(\hat{\mu}_-^T\hat{\Sigma}^{-1}\hat{\mu}_- - \hat{\mu}_+^T\hat{\Sigma}^{-1}\hat{\mu}_+)$

Hence the class distribution:
$P(Y=1|\mathbf{x}) = \frac{1}{1+\exp(-f(\mathbf{x}))} = \sigma(\mathbf{w^T x + w_0})$

4. LDA is linear in *d* while QDA requires $\frac{(d+1)d}{2}$ parameters to estimate upper (lower) triangle of covariance matrix, in addition to the complexity for estimation of mean and another parameter for the prior.

Next in Agenda

Exam 2019

- Question 3 Convolutional Neural Networks
- Question 5 Clustering
- Question 6 Dimensionality Reduction
- Question 7 Linear Discriminant Analysis
- Question 8 Gaussian Mixture Models and EM Algorithm

Recap: Gaussian Mixture Models and EM Algorithm

Gaussian mixtures: $P(\mathbf{x}|\theta) = P(\mathbf{x}|\mu, \Sigma, \mathbf{w}) = \sum_i \pi_i \mathcal{N}(\mathbf{x}; \mu_i, \Sigma_i)$

where $\pi_i \geq 0 \quad \text{s.t.} \sum_i \pi_i = 1$

EM Algorithm:

- ☐ Initialize the parameters $\theta^{(0)}$
- ☐ For $t = 1, 2, \ldots$until convergence:
  - ○ **E-step:** Predict most likely class for each data point

$$\pi_{j,i} = P(z_i^{(t)} = j|\mathbf{x}_i, \theta^{(t-1)}) = \frac{P(z_i^{(t)}=j|\theta^{(t-1)})P(\mathbf{x}_i|z_i^{(t)}=j,\theta^{(t-1)})}{P(\mathbf{x}_i|\theta^{(t-1)})} = \frac{\pi_j P(\mathbf{x}_i|\mu_j^{(t-1)},\Sigma_j^{(t-1)})}{\sum_l \pi_j P(\mathbf{x}_i|\mu_l^{(t-1)},\Sigma_l^{(t-1)})}$$

  - ○ **M-step:** Maximize the likelihood function

$$\theta^{(t)} = \arg\max_\theta P(\mathbf{x}_{1:N}|\theta) = \arg\max_\theta \prod_i P(\mathbf{x}_i|\theta) = \arg\max_\theta \prod_i \sum_j \pi_j \mathcal{N}(\mathbf{x}_i; \mu_i, \Sigma_i)$$

Maximum Likelihood Estimation:

$$\mu_j^* = \frac{\sum_i \pi_{j,i}\mathbf{x}_i}{\sum_i \pi_{j,i}} \qquad \Sigma_j^* = \frac{\sum_i \pi_{j,i}(\mathbf{x}_i - \mu_j^*)(\mathbf{x}_i - \mu_j^*)^T}{\sum_i \pi_{j,i}} \qquad \pi_j^* = \frac{1}{N}\sum_{i=1}^N \pi_{j,i}$$

Exam 2019 - 8.i-iv

Suppose you have a dataset $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, where each $\mathbf{x}_i \in \mathbb{R}^d$ is generated with a Gaussian mixture model (GMM) with k components with weights $\{w_1 \dots w_k\}$, means $\{\mu_1, \dots, \mu_k\}$, and covariance matrices $\{\mathbf{\Sigma}_1, \dots, \mathbf{\Sigma}_k\}$.

(i) *Basics (MC)*
Which of the following independence assumptions is encoded in the Gaussian mixture model?
☒ The points $\mathbf{x}_i$ are independent of each other.
☐ The points $\mathbf{x}_i$ are dependent on each other when conditioned that they were sampled from the same mixture component.
☐ The points $\mathbf{x}_i$ need to be dependent of each other.
☐ The points $\mathbf{x}_i$ can be but not need to be dependent of each other.

(ii) *Basics II (Form)*
What is the a priori probability that a point from the dataset belongs to the i-th center.
Hint: The answer is a symbol not a text and has already been introduced.

$w_i = P(z = i)$ where $i \in \{1, 2, \dots, k\}$

(iii) *E-step (MC)*
Let

$$\theta = (w_1 \dots w_k, \mu_1, \dots, \mu_k, \mathbf{\Sigma}_1, \dots, \mathbf{\Sigma}_k)$$

be the parameters from the previous M-step and z_i be the assignment of point $\mathbf{x}_i$ to a mixture component taking values in $\{1 \dots k\}$. Capital letters denote random variables. In the E-step of the algorithm you compute:
☐ $P(X = x_i | Z_i = j, \theta)$ for all $i \in \{1 \dots n\}$ and $j \in \{1 \dots k\}$.
☒ $P(Z_i = j | X = x_i, \theta)$ for all $i \in \{1 \dots n\}$ and $j \in \{1 \dots k\}$.
☐ $P(Z_i = j, \theta)$ for all $i \in \{1 \dots k\}$ and $j \in \{1 \dots k\}$.
☐ $P(X = x_i, \theta)$ for all $i \in \{1 \dots n\}$.

○ **E-step:** Predict most likely class for each data point

$$\pi_{j,i} = P(z_i^{(t)} = j | \mathbf{x}_i, \theta^{(t-1)}) = \frac{P(z_i^{(t)}=j|\theta^{(t-1)})P(\mathbf{x}_i|z_i^{(t)}=j,\theta^{(t-1)})}{P(\mathbf{x}_i|\theta^{(t-1)})} = \frac{\pi_j P(\mathbf{x}_i|\mu_j^{(t-1)},\Sigma_j^{(t-1)})}{\sum_l \pi_j P(\mathbf{x}_i|\mu_l^{(t-1)},\Sigma_l^{(t-1)})}$$

(iv) *M-step (MC)*
In the M step of the EM algorithm applied to GMMs you:
☐ maximize the marginal data log-likelihood, where the marginalization is taken with respect to the distribution calculated in the E-step.
☒ maximize the expected joint data log-likelihood, where the expectation is taken with respect to the distribution calculated in the E-step.
☐ maximize the expected joint data log-likelihood, where the expectation is taken with respect to the true distribution $P(w|\mathbf{x})$.
☐ maximize the marginal data log-likelihood, where the marginalization is taken with respect to the true distribution $P(w|\mathbf{x})$.

○ **M-step:** Maximize the likelihood function

$$\theta^{(t)} = \arg\max_{\theta} P(\mathbf{x}_{1:N}|\theta) = \arg\max_{\theta} \prod_i P(\mathbf{x}_i|\theta) = \arg\max_{\theta} \prod_i \sum_j \pi_j \mathcal{N}(\mathbf{x}_i; \mu_i, \Sigma_i)$$

Recap: EM Algorithm

General Procedure:

☐ Initialize the parameters $\theta^{(0)}$

☐ For $t = 1, 2, \ldots$until convergence:

- **E-step:** Calculate expected complete data log-likelihood (function of θ)

$$Q(\theta|\theta^{(t-1)}) = \mathbb{E}_{\mathbf{z}_{1:N}}[\log P(\mathbf{x}_{1:N}, \mathbf{z}_{1:N}|\theta)|\mathbf{x}_{1:N}, \theta^{(t-1)}]$$

$$\mathbf{x}_i \text{ : observed values} \qquad \mathbf{z}_i \text{ : missing values}$$

- **M-step:** Maximize the likelihood function

$$\theta^{(t)} = \arg\max_{\theta} Q(\theta|\theta^{(t-1)})$$

Exam 2019 - 8(2).i

Your friend comes up with an unfair four-sided die which takes values in $\{1, 2, 3, 4\}$ and asks your help to estimate its properties.

She knows that the die takes value 1 with probability $\frac{1}{4}$, 2 with probability ε, 3 with probability 2ε and 4 with probability $\frac{3}{4} - 3\varepsilon$. She already tried to estimate the probabilities and she recorded the outcomes of rolling the die. She observed that the side 4 appeared x_4 times and the side 3 appeared x_3 times. She also observed that the side 1 and side 2 appeared x_{12} times in total, unfortunately she did not note how many times 1 and 2 appeared separately. In other words, x_1 and x_2 are unknown values with $x_1 + x_2 = x_{12}$ and she only knows x_{12}. We can model the unknown occurrences of side 1 and 2 by two random variables X_1 and X_2, respectively. These two random variables then satisfy the constraint $X_1 + X_2 = x_{12}$. You immediately realize that the EM algorithm can be used to get a MLE for ε and you decide to calculate it.

Comment: Simple MLE estimate can be used to solve this problem by ignoring some information, but due to data efficiency we choose to perform EM. Also, note that capital letters denote random variables.

(i) *EM on a tetrahedron die - E-Step (MC)*
What is $r_1 = \mathbb{E}[X_1|\varepsilon]$ and $r_2 = \mathbb{E}[X_2|\varepsilon]$ respectively?

[✓ $x_{12}\frac{1/4}{1/4+\varepsilon}$, $x_{12}\frac{\varepsilon}{1/4+\varepsilon}$]

☐ $x_{12}\frac{\varepsilon}{1/2+\varepsilon}$, $x_{12}\frac{\varepsilon}{1/2+\varepsilon}$

☐ $x_{12}\frac{\varepsilon}{1+\varepsilon}$, $x_{12}\frac{\varepsilon/4}{1+\varepsilon}$

☐ $\varepsilon\frac{x_{12}}{1/4+x_1}$, $\varepsilon\frac{x_{12}}{1/4+x_2}$

☐ $\varepsilon\frac{x_{12}+\varepsilon}{1/4+\varepsilon}$, $\varepsilon\frac{x_{12}+\varepsilon}{1/4+\varepsilon}$

☐ $\varepsilon\frac{x_{12}}{1/2+\varepsilon}$, $\varepsilon\frac{x_{12}}{1/4+\varepsilon}$

☐ $\varepsilon\frac{x_{12}+1/4}{1/2+\varepsilon}$, $\varepsilon\frac{x_{12}+1/4}{1/4+\varepsilon}$

Solution to (2).i

(A Multinomial Example)
$P(Y=1) = \frac{1}{4},\ P(Y=2) = \varepsilon,\ P(Y=3) = 2\varepsilon$ and $P(Y=4) = \frac{3}{4} - 3\varepsilon$ where $\{Y = j\}$ is observed $x_j, j = [1:4]$ times.
The density of data: $f_{X|\theta} = \frac{N!}{x_1!x_2!x_3!x_4!}(\frac{1}{4})^{x_1}(\varepsilon)^{x_2}(2\varepsilon)^{x_3}(\frac{3}{4} - 3\varepsilon)^{x_4}$
Log likelihood is given by: $\log f_{X|\theta} = c + x_1 \log\frac{1}{4} + x_2\log\varepsilon + x_3(\log 2\varepsilon) + x_4\log(\frac{3}{4} - 3\varepsilon)$

Note that x_1 and x_2 are not observed (are missing).
Denote them by random variables X_1 and X_2 such that $X_1 + X_2 = x_{12}$
Log likelihood can be re-written as:

$$\log f_{Y|\theta} = c + (x_{12} - X_2)\log\frac{1}{4} + X_2\log\varepsilon + x_3(\log 2\varepsilon) + x_4\log(\frac{3}{4} - 3\varepsilon)$$

We write first **E-step** as:

$$\begin{aligned} Q(\theta|\theta^{(0)}) &= \mathbb{E}_{X_2}[\log f_{Y|\theta}|x_{12}, x_3, x_4, \varepsilon] \\ &= \mathbb{E}_{X_2}[c + (x_{12} - X_2)\log\frac{1}{4} + X_2\log\varepsilon + x_3(\log 2\varepsilon) + x_4\log(\frac{3}{4} - 3\varepsilon)|x_{12}, x_3, x_4, \varepsilon] \\ &= \mathbb{E}_{X_2}[X_2\log\varepsilon|x_{12}, \varepsilon] + x_3(\log 2\varepsilon) + x_4\log(\frac{3}{4} - 3\varepsilon) \end{aligned}$$

Note that X_2 is binomial with sample size x_{12} and parameter $\frac{\varepsilon}{1/4+\varepsilon}$. Therefore,

$$r_2 = \mathbb{E}_{X_2}[X_2|\varepsilon] = \frac{x_{12}\varepsilon}{1/4+\varepsilon}, \quad \text{moreover,}\ r_1 = x_{12} - r_2 = \frac{x_{12}1/4}{1/4+\varepsilon}$$

Exam 2019 - 8(2).ii

Your friend comes up with an unfair four-sided die which takes values in $\{1, 2, 3, 4\}$ and asks your help to estimate its properties.

She knows that the die takes value 1 with probability $\frac{1}{4}$, 2 with probability ε, 3 with probability 2ε and 4 with probability $\frac{3}{4} - 3\varepsilon$. She already tried to estimate the probabilities and she recorded the outcomes of rolling the die. She observed that the side 4 appeared x_4 times and the side 3 appeared x_3 times. She also observed that the side 1 and side 2 appeared x_{12} times in total, unfortunately she did not note how many times 1 and 2 appeared separately. In other words, x_1 and x_2 are unknown values with $x_1 + x_2 = x_{12}$ and she only knows x_{12}. We can model the unknown occurrences of side 1 and 2 by two random variables X_1 and X_2, respectively. These two random variables then satisfy the constraint $X_1 + X_2 = x_{12}$. You immediately realize that the EM algorithm can be used to get a MLE for ε and you decide to calculate it.

Comment: Simple MLE estimate can be used to solve this problem by ignoring some information, but due to data efficiency we choose to perform EM. Also, note that capital letters denote random variables.

(ii) *EM on a tetrahedron die - M-Step (MC)*
What is the MLE of ε? You can now use the expected values of X_1 and X_2, r_1 and r_2 respectively.

☐ $\frac{x_{12}-r_2+x_3}{4(x_{12}-r_2+x_3+x_4)}$
☐ $\frac{x_{12}+r_1+x_3}{4(x_{12}+x_3+x_4)}$
☐ $\frac{x_{12}+x_4}{4(x_{12}+x_3+x_4)}$
☐ $\frac{r_1+r_2+x_4}{4(x_{12}-x_3+x_4)}$
☐ $\frac{r_1+x_4}{4(x_{12}+x_3-x_4)}$
☐ $\frac{r_1+r_2+x_4}{4(x_{12}+r_3+x_4)}$
☐ $\frac{r_1+r_2+x_4}{4(x_{12}+x_3+r_4)}$
☐ $\frac{x_4}{4(x_{12}+x_3+r_4)}$
☐ $\frac{r_1+r_2}{4(x_{12}+x_3+r_4)}$
[✓ $\frac{x_{12}-r_1+x_3}{4(x_{12}-r_1+x_3+x_4)}$]

Solution to (2).ii

We have previously computed log-likelihood at **M-step** as:

$$Q(\theta|\varepsilon) = \mathbb{E}_{X_2}[X_2|x_{12}, \varepsilon] \log \varepsilon + x_3(\log 2\varepsilon) + x_4 \log(\tfrac{3}{4} - 3\varepsilon)$$

We write first **M-step** as:

$$\theta^* = \arg\max_{\varepsilon} Q(\theta|\varepsilon) = \arg\max_{\varepsilon} r_2 \log \varepsilon + x_3(\log 2\varepsilon) + x_4 \log(\tfrac{3}{4}(1 - 4\varepsilon))$$

Setting $Q(\theta|\varepsilon)$ to 0 gives us θ^* :

$$\frac{d}{d\varepsilon} Q(\theta|\varepsilon) = \frac{d}{d\varepsilon}\left(r_2 \log \varepsilon + x_3(\log 2\varepsilon) + x_4 \log(\tfrac{3}{4} - 3\varepsilon)\right) = \frac{r_2}{\varepsilon} + \frac{x_3}{\varepsilon} - \frac{x_4}{1-4\varepsilon}$$

$$\frac{r_2}{\varepsilon} + \frac{x_3}{\varepsilon} - \frac{x_4}{1-4\varepsilon} = 0 \text{ at } \varepsilon^* = \frac{r_2+x_3}{4(r_2+x_3+x_4)}$$

Replacing r_2 with $x_{12} - r_1$ gives us the result.

QUESTIONS?

Post them on Piazza!

Thank you