CS 189: Part A

Midterm Review Session

Fall 2024

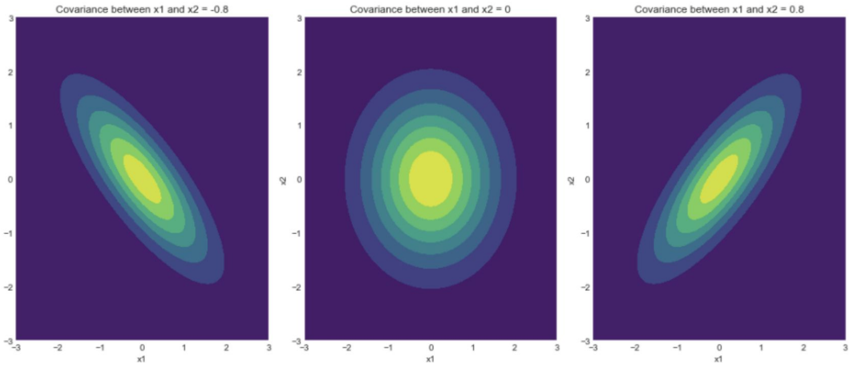
Multivariate Gaussians

$$\boxed{p_{\mu,\Sigma}(x) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)}$$

- We say a random vector follows the multivariate Gaussian distribution, iff
  - Each coordinate is marginally and conditionally Gaussian.

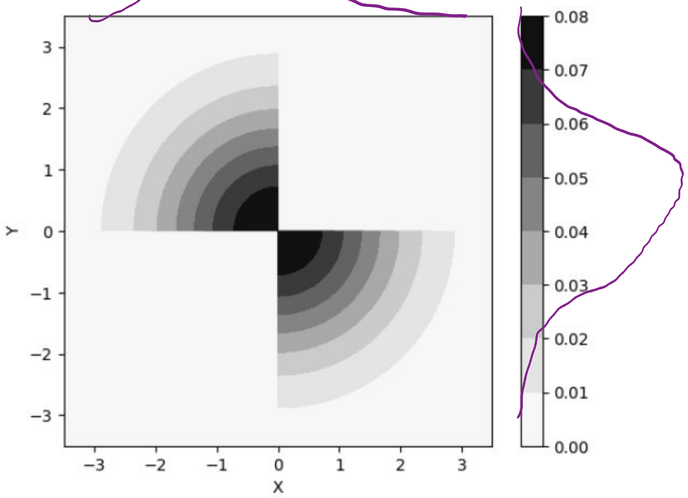
$$\Longleftrightarrow \qquad x_i \sim \mathcal{N} \quad \text{AND} \quad x_i \mid x_j \sim \mathcal{N} \qquad \longleftarrow \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$$

  - It follows the PDF of a multivariate Gaussian distribution
  - $\Longleftrightarrow$ It is a linear transformation of a multivariate gaussian.



"joint Gaussian" $\Leftrightarrow$ "multivar Gaussian"

3. Consider the following probability density function:



Select the best description.

○ X and Y appear to be both marginally and jointly Gaussian.

○ X and Y appear to be marginally Gaussian but not jointly Gaussian.

○ X and Y appear to be jointly Gaussian but not marginally Gaussian.

Solution:

(b) X and Y appear to be marginally Gaussian but not jointly Gaussian

Why? Recall that multivariate Gaussians must be conditionally gaussian. If I give you the value of X, you can immediately tell me the sign of Y. This means coordinates random variables cannot be conditionally Gaussian because there is an area with zero density.

Maximum Likelihood Estimation (MLE)

- Given some data X and an idea of what distribution the data came from (Gaussian, Poisson, etc), MLE **identifies the parameters** of that distribution that **maximize** the probability of observing the data X
- E.g. if $\{x_1, x_2, \ldots, x_n\}$ are sampled from **IID** from a dist $X \sim N(\mu, \sigma)$, solve $\arg\max_{\mu,\sigma} p(x_1, x_2, \ldots, x_n | \mu, \sigma)$
- $L(\theta; x) = p_\theta(x)$ is the **likelihood function**
- Two tricks to getting through these problems:
  - IID: turns a joint probability into a product of individual probabilities
  - Log-likelihood: applying log to the likelihood function does not shift the optimal parameters (log is strictly increasing function), but can simplify math necessary (exponents ⇒ multiplication, products ⇒ sums)

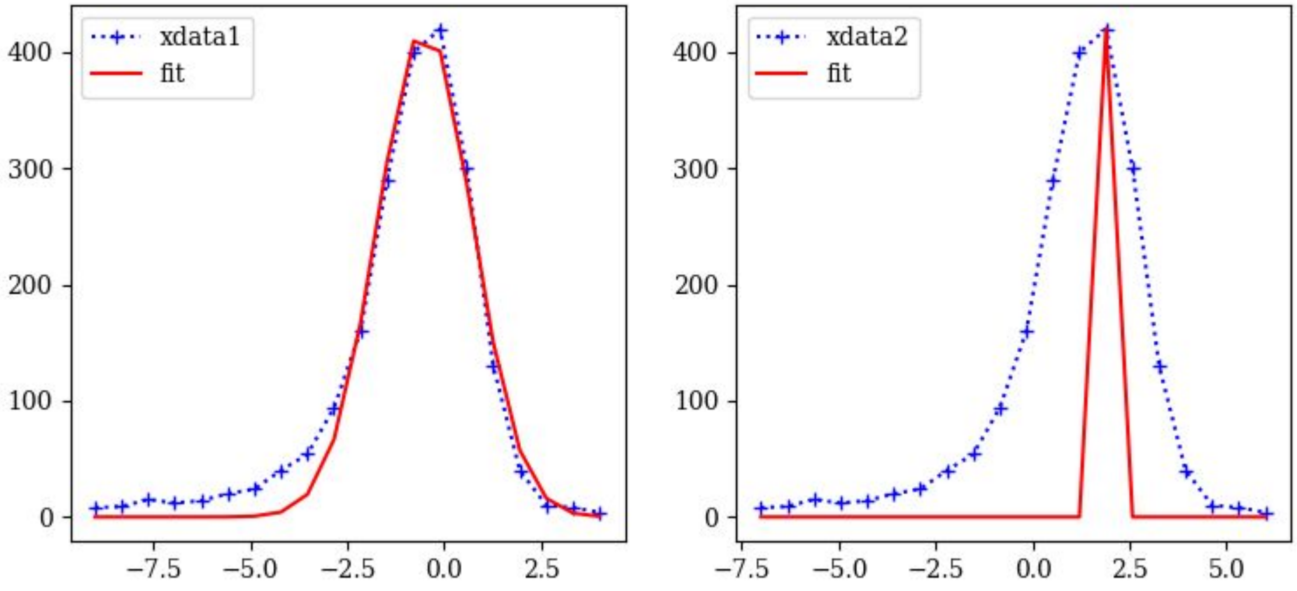
L is fn of θ with param X

p is fn of x with param θ

$$L(\theta; x) = \underbrace{P_\theta(x) = p(x|\theta) = p(x;\theta)}_{\text{you can pick any notation — I like } P_\theta(x)}$$

Maximum Likelihood Estimation (MLE)

E.g.: goal is to determine the “best” parameters mu and sigma for the blue data:



https://stackoverflow.com/questions/55353366/gaussian-data-fit-varying-depending-on-position-of-x-data

Example:

$$p(x) = \begin{cases} \frac{1}{b-a} & \text{if } a \le x \le b \\ 0 & \text{otherwise} \end{cases}$$

We observe the following data points from a uniform distribution U[a, b]:

0, 4, -1, 5, 8, 3, 4.5

What is the MLE of b?

Is the MLE of b unbiased? Recall the definition of bias:

$$\mathbb{P}_{a,b}(\hat{b} < b) = 1$$

$$\text{So } \mathbb{E}_{a,b}[\hat{b}] < b$$

$$\text{bias}(\hat{\theta};\theta) = \mathbb{E}_\theta[\hat{\theta} - \theta]$$

$$L(a,b;x) = \prod_i p(x_i) = \prod_i \frac{1}{b-a} \underbrace{1\{a \le x \le b\}}$$

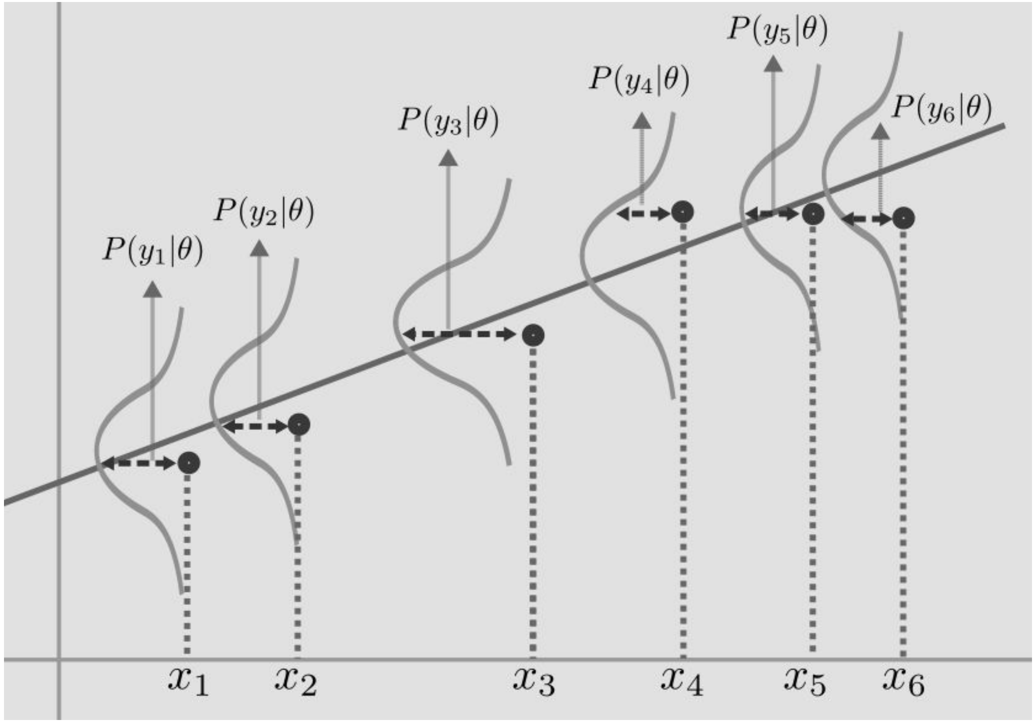
$\hat{b}$ should be smallest upper bound on x

$= \max_i x_i$

Maximum Likelihood Estimation (MLE)

Previous slide models just data points x. How to model y as a function of x?

For every x, let the distribution of y have different parameters. Use MLE to find the best parameters to explain y given the observed x.



https://suriyadeepan.github.io/2017-01-22-mle-linear-regression/

MLE for Linear Regression

Two equivalent ways of looking at this ("Gaussian linear model"):

$$\boxed{\begin{array}{c} y_i \sim N(x_i^T w, \sigma^2) \\ Y = Xw + z; \boxed{z \sim N(0, \sigma^2 I)} \end{array}}$$

noise in observation

$z_i \overset{iid}{\sim} N(0, \sigma^2)$

To find the best weights, we maximize the likelihood function:

- note that we break up the joint probability into a product of probs. since samples IID

$$\boxed{\arg\max_{w} p(y_1, y_2, \ldots, y_n | X, w)} = \arg\max_{w} \prod_{i=1}^{n} \left(\frac{1}{\sqrt{2\pi\sigma}} \exp - \frac{(x_i^T w - y_i)^2}{2\sigma^2} \right)$$

Regularization in Linear Regression

Recall MLE estimate for linear regression: $\mathbf{(X^TX)^{-1}X^Ty = w^*}$

- This assumes the data are distributed normally, but nothing is assumed about the distribution of **w** (what **w** are we more likely to see)
- MLE applied to data for the linear regression model.
- The result is the **OLS estimate!!!**

Maximum *a posteriori* estimation (MAP)

- Alternative to MLE that incorporates additional pre-existing knowledge on how the data are distributed (namely, what the parameters of the distribution likely are).
- Mathematically: reformulate objective function to maximize posterior probability:

$$\underset{\theta}{\arg\max}\, p(\theta|x) = \underset{\theta}{\arg\max}\, p(x|\theta)p(\theta)$$

$$\boxed{\arg\max_{\theta} p(X|\theta)}$$

$$\arg\max_{\theta} \boxed{p(\theta|X)} = \arg\max_{\theta} \frac{\boxed{p(X|\theta)p(\theta)}}{\cancel{p(X)}}$$

likelihood!

$$p(x) = \int p(x|\theta)\, p(\theta)\, d\theta$$

often hard to compute, but we don't need to!

MLE vs. MAP

MLE MAP

- Maximize p(data|model) vs. maximize p(model|data).
- MAP is explicit about what the prior is, but that comes with the **burden of identifying a prior**.
- Both include the likelihood function.
- Bayesian analysis in general requires normalizing by p(data), though this is avoided in MAP because p(data) does not vary with the parameters that are being maximized over.

MAP in Practice

$\pi(w)$ prior, not necessarily normalized

$\int \pi = 1$ usually not needed

If we do MAP for linear regression with a Gaussian prior, we get out ordinary least squares plus an L2 regularization term:

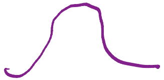
$$\begin{aligned}\arg\max_{w} \log p(y|w,X)p(w) &= \arg\max_{w} \sum_{i=1} \underline{\log N(x_i^T w, \sigma^2) + \log N(0, \underline{\tau}^2 I)} \\ &= \arg\min_{w} ||y - Xw||_2^2 + \lambda||w||_2^2\end{aligned}$$

Other regularization terms fall out if you use a different prior (e.g. Laplace prior => L1 regularization).

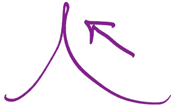
ridge → $+\lambda\|w\|_2^2$

LASSO → $+\lambda\|w\|_1$

$w_i \overset{i.i.d.}{\sim} N(0, \sigma^2)$



$w_i \overset{i.i.d.}{\sim} \text{Laplace}(0, b)$



Example: Find the MAP prior

You have an interesting guess that the l3-norm gives you a useful prior for linear regression, so you want to minimize

$$\arg\min_{w} \|Xw - y\|_2^2 + \text{[redacted]}\|w\|_3^3$$

With the l3 norm cubed is defined as

$$\|w\|_3^3 = \left(\sum_i |w_i|^3\right)$$

Find a corresponding (unnormalized) prior $\pi(w)$ such that the minimization problem is equivalent to the MAP problem of the Gaussian linear model with prior $\pi(w)$

$$\underset{w}{\operatorname{argmax}}\ P_w(Y|X)\,\pi(w)$$

$$\boxed{\pi(w) = \exp\left(-\frac{\|w\|_3^3}{2\sigma^2}\right)}$$

$$= \underset{w}{\operatorname{argmax}}\ \underbrace{C \exp\left(-\frac{\|Y-Xw\|_2^2}{2\sigma^2}\right) \exp\left(-\frac{\|w\|_3^3}{2\sigma^2}\right)}$$

$$= \underset{w}{\operatorname{argmax}}\ \log\left(\ \dots\ \right)$$

$$= \underset{w}{\operatorname{argmax}}\left[\log C - \frac{\|Y-Xw\|_2^2}{2\sigma^2} - \frac{\|w\|_3^3}{2\sigma^2}\right]$$

$$= \underset{w}{\operatorname{argmin}}\ \frac{\|Y-Xw\|_2^2}{2\sigma^2} + \frac{\|w\|_3^3}{2\sigma^2}$$

$$= \underset{w}{\operatorname{argmin}}\ \|Y-Xw\|_2^2 + \|w\|_3^3$$

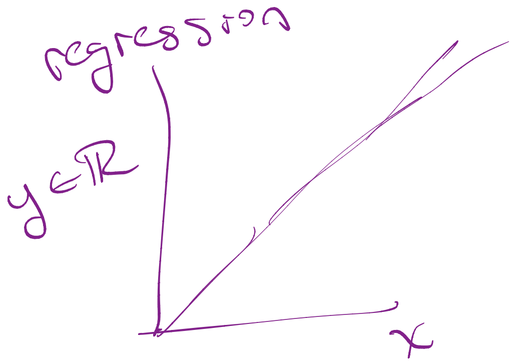
Generative vs Discriminative Classification

- Generative: models the joint p(x, y) then chooses the y that maximizes p(y | x).
- Maximizing p(y | x) is equivalent to maximizing p(x | y)p(y)
  - p(y) = sample proportions of individual labels
  - p(x | y) is usually what we model (ex. Gaussian)

class.

binary $y \in \{0,1\}$

predict y that maximizes: $p(y|x)$



binary classification:

$$\begin{tikzcd} & y=0 \\ x \arrow[ur] \arrow[dr] & \\ & y=1 \end{tikzcd}$$

Discriminative: f_θ models $P(Y|X)$ directly

Generative: f_θ models $P(X, Y)$ or $P(X|Y)$

use Bayes rule to then get $P(Y|X)$

Logistic regression

- DISCRIMINATIVE
- Uses the sigmoid function (from lecture 7 slides):

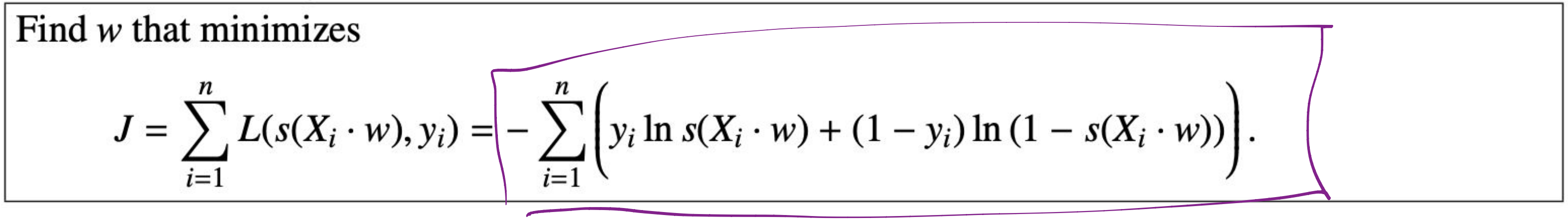
$$\sigma(z) = \underset{y}{p(k|x)} = \frac{1}{1+e^{-z}},$$

$$z = \theta^{\top} x + \theta_0,$$

Logistic regression: optimization

Minimize the following cost function (s = sigma, taken from Prof. Shewchuk's lecture notes)

= Sigmoid

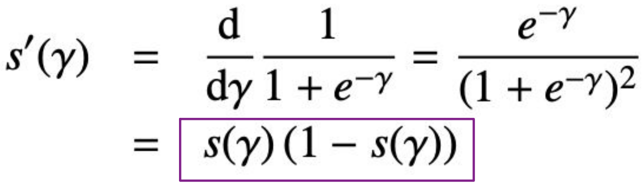


Find w that minimizes

$$J = \sum_{i=1}^{n} L(s(X_i \cdot w), y_i) = \boxed{-\sum_{i=1}^{n} \left(y_i \ln s(X_i \cdot w) + (1 - y_i) \ln (1 - s(X_i \cdot w)) \right)}.$$

No closed form solution! So we do gradient descent.

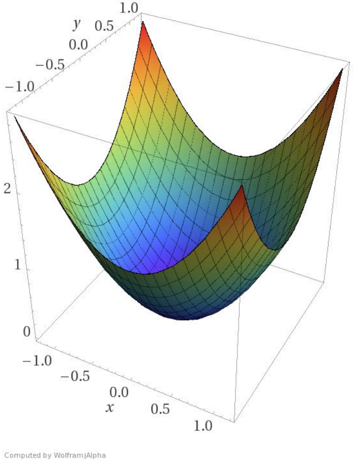
Recall the derivative of the sigmoid function (good for your cheatsheet)



$$\begin{aligned} s'(\gamma) &= \frac{\mathrm{d}}{\mathrm{d}\gamma} \frac{1}{1 + e^{-\gamma}} = \frac{e^{-\gamma}}{(1 + e^{-\gamma})^2} \\ &= \boxed{s(\gamma)\,(1 - s(\gamma))} \end{aligned}$$

Gradient Descent

Works for convex functions!

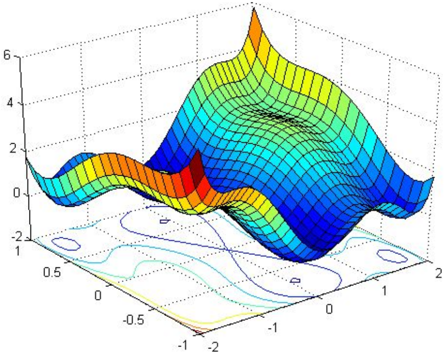


Computed by Wolfram|Alpha

The gradient descent method follows the simple algorithmic procedure:

1. Choose $x_0 \in \mathbb{R}^d$ and set $k = 0$
2. Choose $t_k > 0$ and set $x_{k+1} = x_k - t_k \nabla f(x_k)$ and $k = k + 1$,
3. Repeat 2 until converged.

Works in practice for non-convex functions involving neural networks!



Neural networks

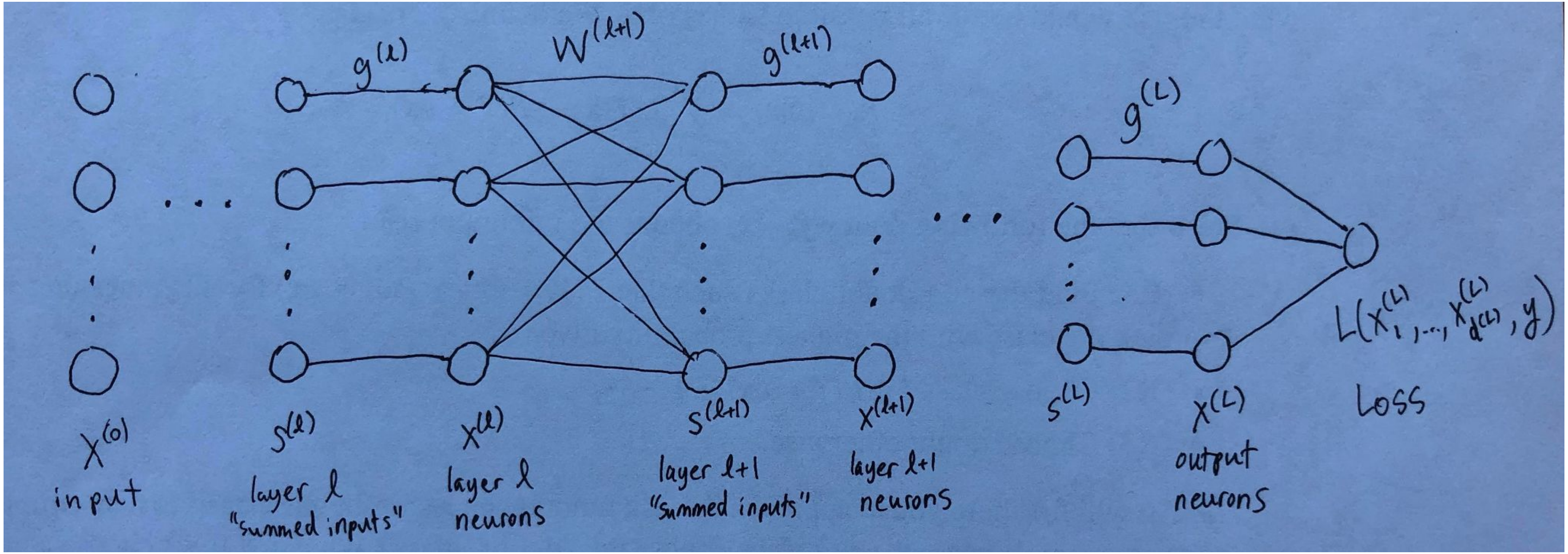
- Repeated applications of linear mapping W$^{(\ell)}$ (potentially plus bias b$^{\ell}$) followed by non-linearity g$^{\ell}$

$$s_j^{(\ell+1)} = \sum x_i^{(\ell)} W_{ij}^{(\ell+1)}$$

$$x_j^{(\ell+1)} = g^{(\ell+1)}(s_j^{(\ell+1)}) = g(\sum x_i^{(\ell)} W_{ij}^{(\ell+1)})$$

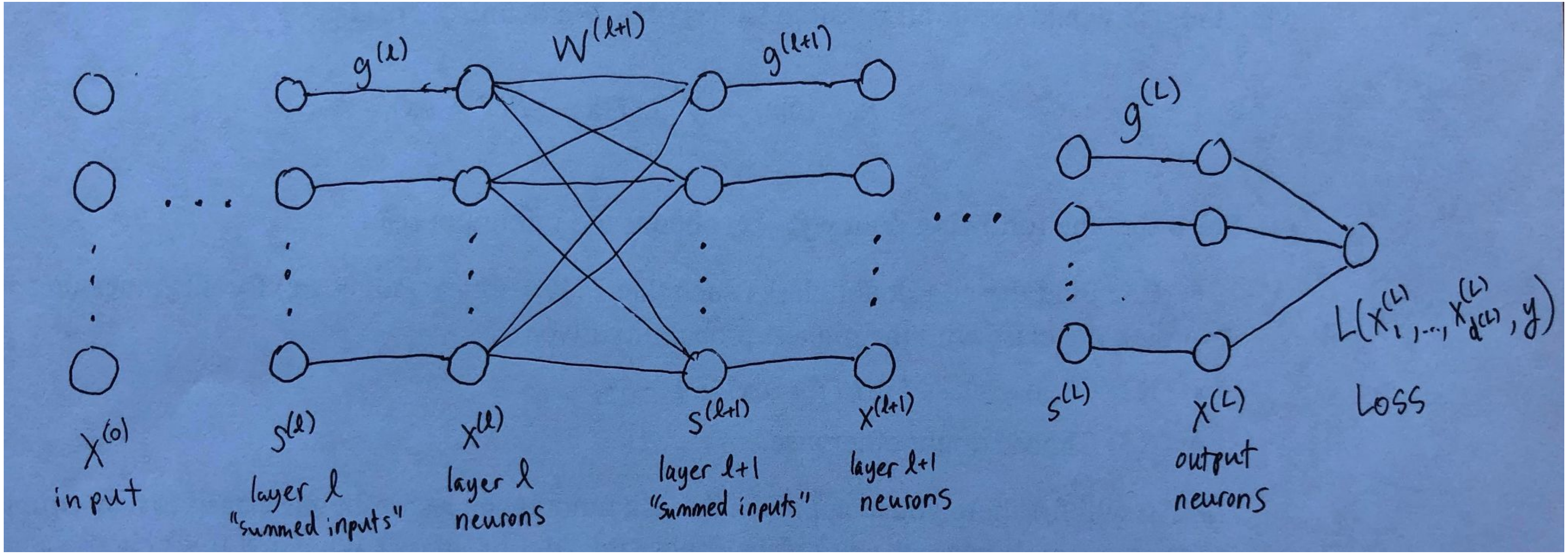
$$s^{(\ell+1)\top} = x^{(\ell)\top} W^{(\ell+1)}$$

$$x^{(\ell+1)^\top} = g(s^{(\ell+1)\top}) = g(x^{(\ell)\top} W^{(\ell+1)})$$

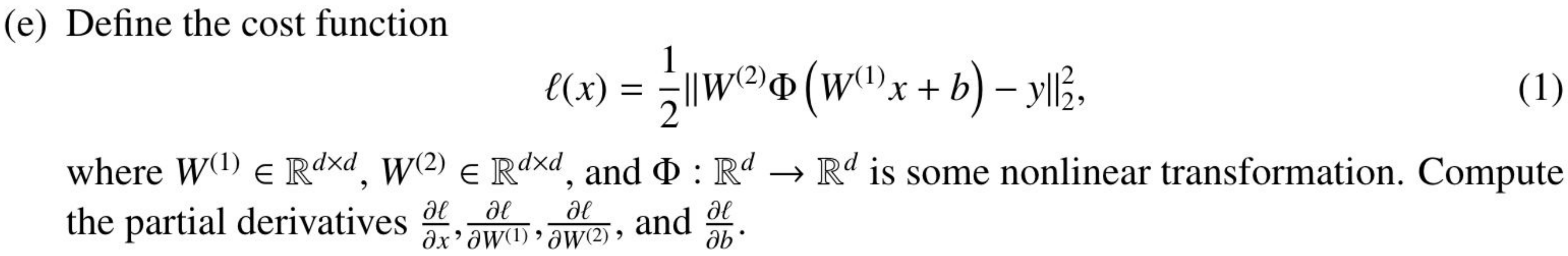


Backpropagation

- Efficient recursive algorithm to compute $\nabla_{W^{(1)}} L, \ldots, \nabla_{W^{(L)}} L$
- Base case: $\delta_i^{(L)} = \dfrac{\partial L}{\partial s_i^{(L)}} = g^{(L)\prime}(s_i^{(L)}) \dfrac{\partial L}{\partial x_i^{(L)}}$, Inductive step: $\delta_i^{(\ell)} = g^{(\ell)\prime}(s_i^{(\ell)}) \sum_j W_{ij}^{(\ell+1)} \delta_j^{(\ell+1)}$
- Partial derivatives wrt weights: $\dfrac{\partial L}{\partial W_{ij}^{(\ell+1)}} = x_i^{(\ell)} \delta_j^{(\ell+1)}$ (All these operations are efficiently vectorizable)



Backpropagation Practice

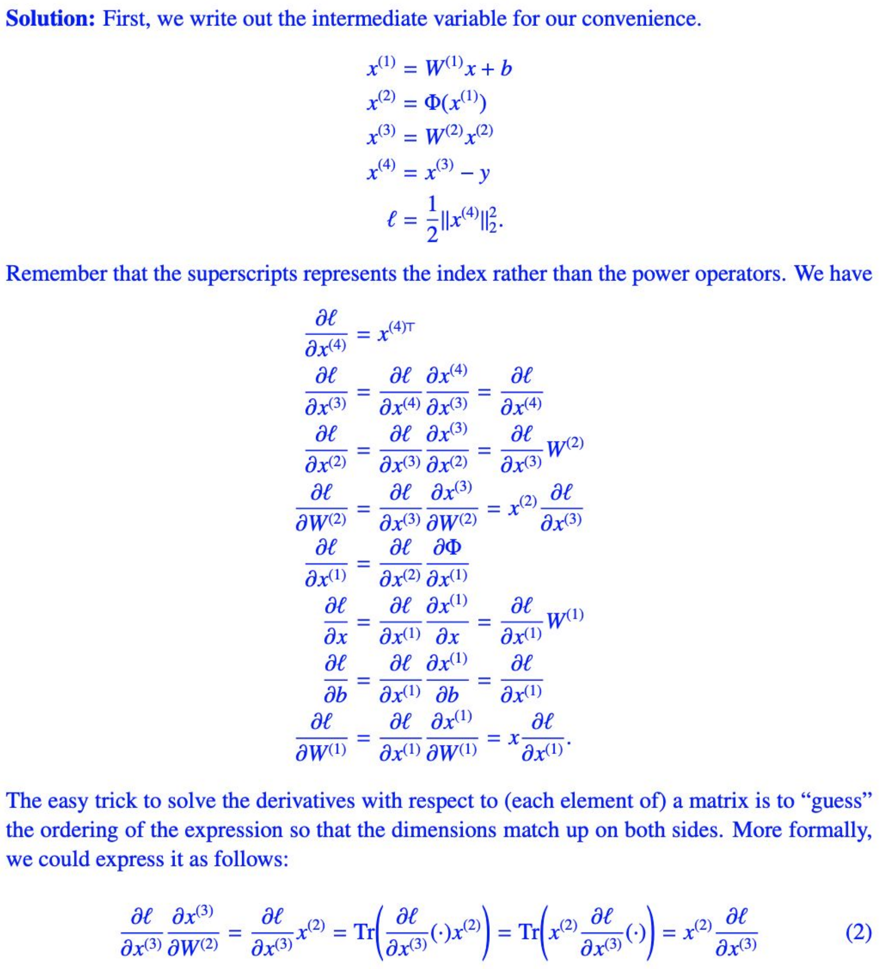


(e) Define the cost function

$$\ell(x) = \frac{1}{2}\|W^{(2)}\Phi\left(W^{(1)}x + b\right) - y\|_2^2, \tag{1}$$

where $W^{(1)} \in \mathbb{R}^{d\times d}$, $W^{(2)} \in \mathbb{R}^{d\times d}$, and $\Phi : \mathbb{R}^d \to \mathbb{R}^d$ is some nonlinear transformation. Compute the partial derivatives $\frac{\partial \ell}{\partial x}, \frac{\partial \ell}{\partial W^{(1)}}, \frac{\partial \ell}{\partial W^{(2)}}$, and $\frac{\partial \ell}{\partial b}$.

Solution (see Discussion 4)



Solution: First, we write out the intermediate variable for our convenience.

$$\begin{aligned} x^{(1)} &= W^{(1)}x + b \\ x^{(2)} &= \Phi(x^{(1)}) \\ x^{(3)} &= W^{(2)}x^{(2)} \\ x^{(4)} &= x^{(3)} - y \\ \ell &= \frac{1}{2}\|x^{(4)}\|_2^2. \end{aligned}$$

Remember that the superscripts represents the index rather than the power operators. We have

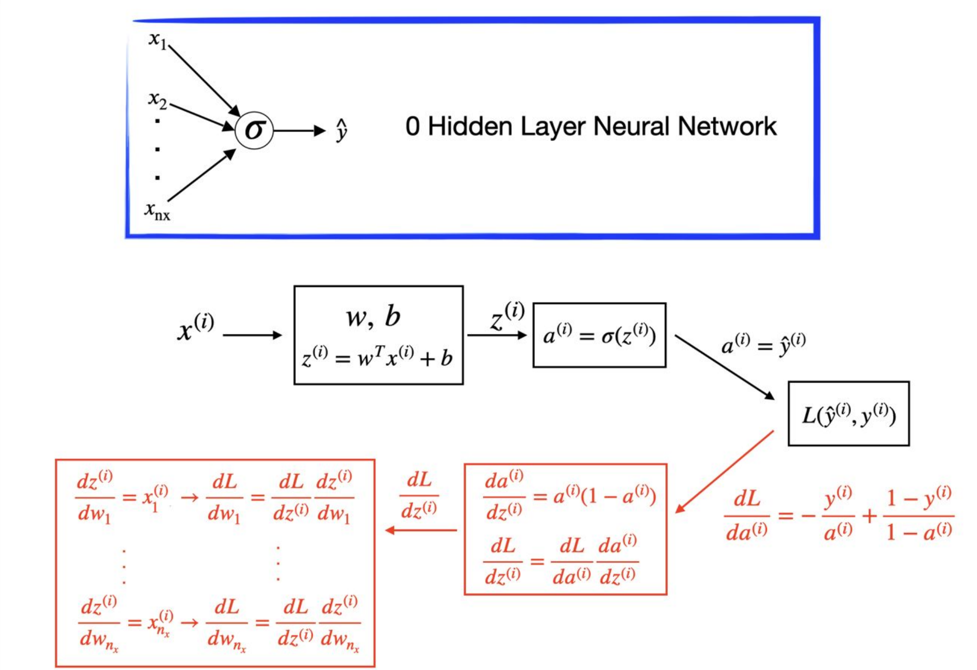
$$\begin{aligned} \frac{\partial \ell}{\partial x^{(4)}} &= x^{(4)\top} \\ \frac{\partial \ell}{\partial x^{(3)}} &= \frac{\partial \ell}{\partial x^{(4)}}\frac{\partial x^{(4)}}{\partial x^{(3)}} = \frac{\partial \ell}{\partial x^{(4)}} \\ \frac{\partial \ell}{\partial x^{(2)}} &= \frac{\partial \ell}{\partial x^{(3)}}\frac{\partial x^{(3)}}{\partial x^{(2)}} = \frac{\partial \ell}{\partial x^{(3)}}W^{(2)} \\ \frac{\partial \ell}{\partial W^{(2)}} &= \frac{\partial \ell}{\partial x^{(3)}}\frac{\partial x^{(3)}}{\partial W^{(2)}} = x^{(2)}\frac{\partial \ell}{\partial x^{(3)}} \\ \frac{\partial \ell}{\partial x^{(1)}} &= \frac{\partial \ell}{\partial x^{(2)}}\frac{\partial \Phi}{\partial x^{(1)}} \\ \frac{\partial \ell}{\partial x} &= \frac{\partial \ell}{\partial x^{(1)}}\frac{\partial x^{(1)}}{\partial x} = \frac{\partial \ell}{\partial x^{(1)}}W^{(1)} \\ \frac{\partial \ell}{\partial b} &= \frac{\partial \ell}{\partial x^{(1)}}\frac{\partial x^{(1)}}{\partial b} = \frac{\partial \ell}{\partial x^{(1)}} \\ \frac{\partial \ell}{\partial W^{(1)}} &= \frac{\partial \ell}{\partial x^{(1)}}\frac{\partial x^{(1)}}{\partial W^{(1)}} = x\frac{\partial \ell}{\partial x^{(1)}}. \end{aligned}$$

The easy trick to solve the derivatives with respect to (each element of) a matrix is to "guess" the ordering of the expression so that the dimensions match up on both sides. More formally, we could express it as follows:

$$\frac{\partial \ell}{\partial x^{(3)}}\frac{\partial x^{(3)}}{\partial W^{(2)}} = \frac{\partial \ell}{\partial x^{(3)}}x^{(2)} = \mathrm{Tr}\left(\frac{\partial \ell}{\partial x^{(3)}}(\cdot)x^{(2)}\right) = \mathrm{Tr}\left(x^{(2)}\frac{\partial \ell}{\partial x^{(3)}}(\cdot)\right) = x^{(2)}\frac{\partial \ell}{\partial x^{(3)}} \tag{2}$$

Feed Forward Neural Nets (also called MLPs)

- Use a linear function to transform inputs before activation


x_1
x_2
x_{nx}
σ
$\hat{y}$
0 Hidden Layer Neural Network
$x^{(i)}$
w, b
$z^{(i)} = w^T x^{(i)} + b$
$z^{(i)}$
$a^{(i)} = \sigma(z^{(i)})$
$a^{(i)} = \hat{y}^{(i)}$
$L(\hat{y}^{(i)}, y^{(i)})$
$\frac{dL}{da^{(i)}} = -\frac{y^{(i)}}{a^{(i)}} + \frac{1-y^{(i)}}{1-a^{(i)}}$
$\frac{da^{(i)}}{dz^{(i)}} = a^{(i)}(1-a^{(i)})$
$\frac{dL}{dz^{(i)}} = \frac{dL}{da^{(i)}} \frac{da^{(i)}}{dz^{(i)}}$
$\frac{dL}{dz^{(i)}}$
$\frac{dz^{(i)}}{dw_1} = x_1^{(i)} \rightarrow \frac{dL}{dw_1} = \frac{dL}{dz^{(i)}} \frac{dz^{(i)}}{dw_1}$
$\frac{dz^{(i)}}{dw_{n_x}} = x_{n_x}^{(i)} \rightarrow \frac{dL}{dw_{n_x}} = \frac{dL}{dz^{(i)}} \frac{dz^{(i)}}{dw_{n_x}}$

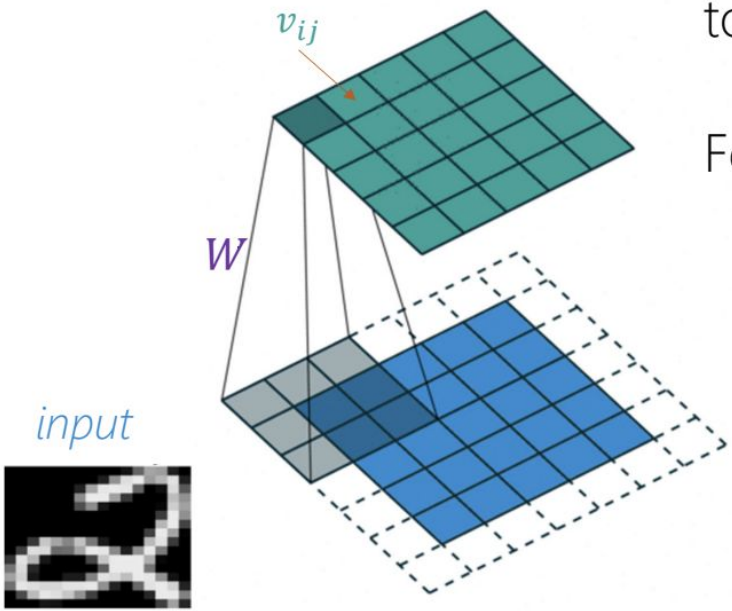
https://scientistcafe.com/ids/feedforward-neural-network

CS 189: Part B

Midterm Review Session

Fall 2024

2D Convolution

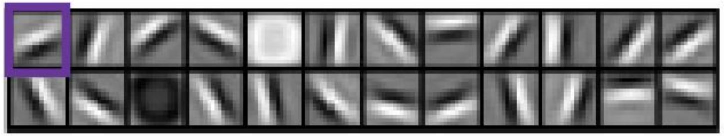


Convolve one learned “filter”, W with the input to get convolution output $\{v_{ij}\}$:

For each position, i, j:

1. Element-wise product of W with image patch centered on i, j (e.g. 3×3).
2. Sum up the results to get one v_{ij}.

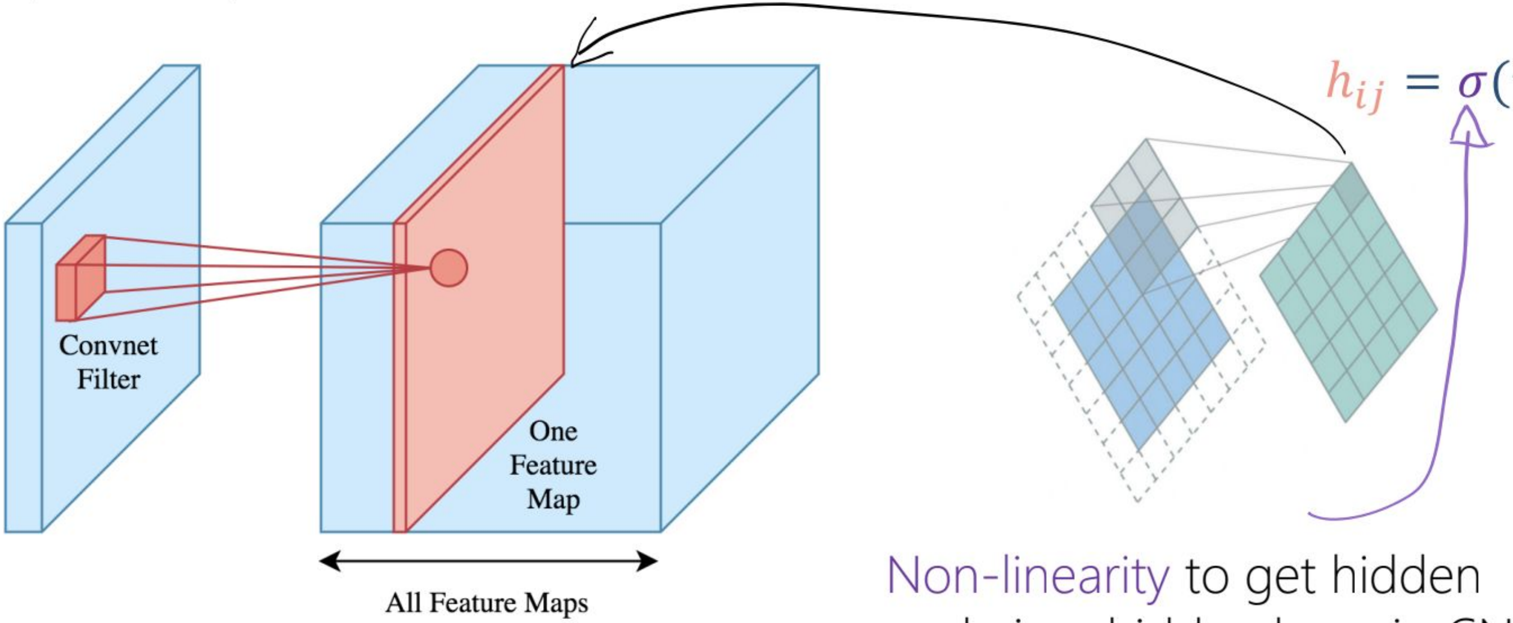
W called filter/template/kernel



https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Networks (CNNs)

- We will actually use multiple feature maps, $\{W_k\}_{k=1}^{K}$
- "Depth" of output "volume" is K:



$h_{ij} = \sigma(v_{ij})$.

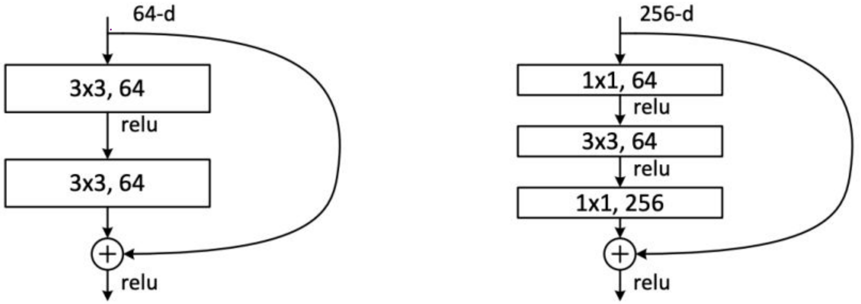
Non-linearity to get hidden node in a hidden layer in CNN

Convolutional Neural Networks (CNNs)

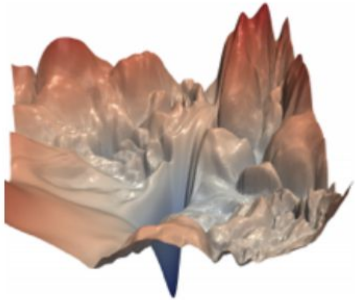
- Convolutions are translationally equivariant
  - Translate input = translate output by same amount (same function applied)
  - Reduces number of params via weight sharing
  - Uses observation that many useful image features are local
- Pooling layers are commonly used to downsample image spatial dimensions
  - We flatten or pool the final output into a one dimensional vector, pass it through one or more linear layers, and then (for classification) get our final classification vector.

Residual Connections and ResNet

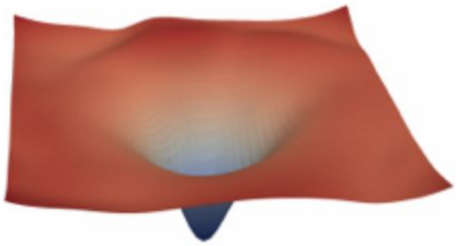
- Very simple high level idea: $x^{(l)} = \sigma\left(z^{(l-1)}\right) + x^{(l-1)}$, rather than $x^{(l)} = \sigma\left(z^{(l-1)}\right)$
- Add a “skip connection” or “residual connection” to CNN
- Allowed for training much deeper, more performant models
- The loss “landscape” of neural networks with residual connections looks nicer



He et al, 2015



Li et al, NIPS ’18



Li et al, NIPS ’18

CNN Hyperparameters

$$\text{Output size} = \left\lfloor \frac{\text{Input size} - \text{Kernel size} + 2 \times \text{Padding}}{\text{Stride}} \right\rfloor + 1$$

$$\text{Number of parameters} = (\mathrm{C}_{\text{in}} \times K_h \times K_w + 1) \times \mathrm{C}_{\text{out}}$$

Batch Normalization (BN)

- BN = normalize outputs using statistics computed from the mini batch

$$x^{(l)} \leftarrow \frac{x^{(l)} - \mu^{(l)}}{\sigma^{(l)} + \epsilon}$$

- BN also includes learnable scale and shift parameters

$$x^{(l)} \leftarrow \gamma x^{(l)} + \beta$$

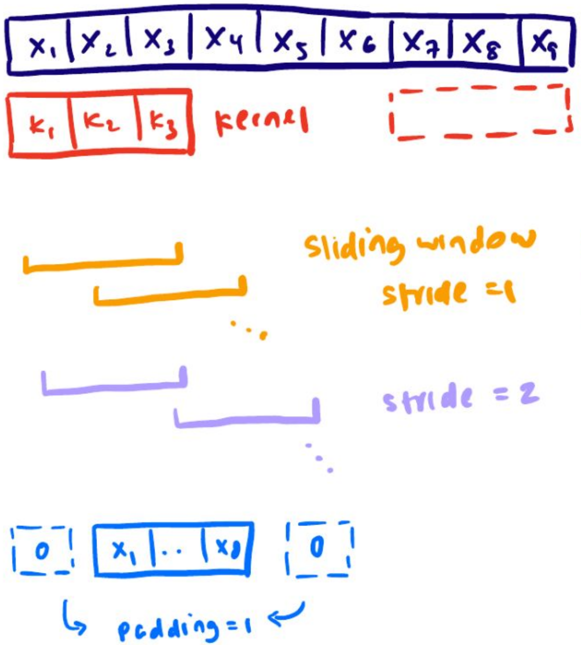
- Models with BN operate in two different modes: “train” vs. “test” or “eval”
  - Train mode: compute statistics using the mini batch
  - Eval mode: use an exponential moving average of the training statistics
- Helps solve the “vanishing gradient” problem.
  - Pushes mini batch activations to be mean 0 and variance 1

Convolution

$o_1 = k_1 x_1 + k_2 x_2 + k_3 x_3$

$o_2 = k_1 x_2 + k_2 x_3 + k_3 x_4$

...



1D data

kernel

stop sliding so kernel always matched to some data

sliding window

stride = 1

stride = 2

↳ padding = 1

Out Dim

$x \in \mathbb{R}^{1 \times d_1}$ $w \in \mathbb{R}^{1 \times k}$ $y \in \mathbb{R}^{1 \times d_2}$

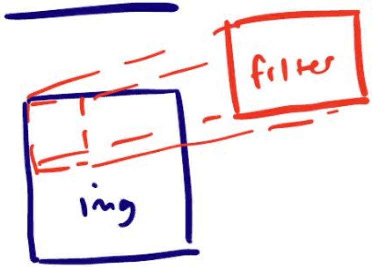
$$d_2 = \left\lfloor \frac{d_1 + 2p - k}{s} \right\rfloor + 1$$

symmetric padding on left / right

floor div for int

↳ increment counter by stride

2D images



a filter can have multiple kernels

$f_1 \dots f_{c_{out}}$

params = $C_{in} \times C_{out} \times K_h \times K_w \times n$

Practice Question

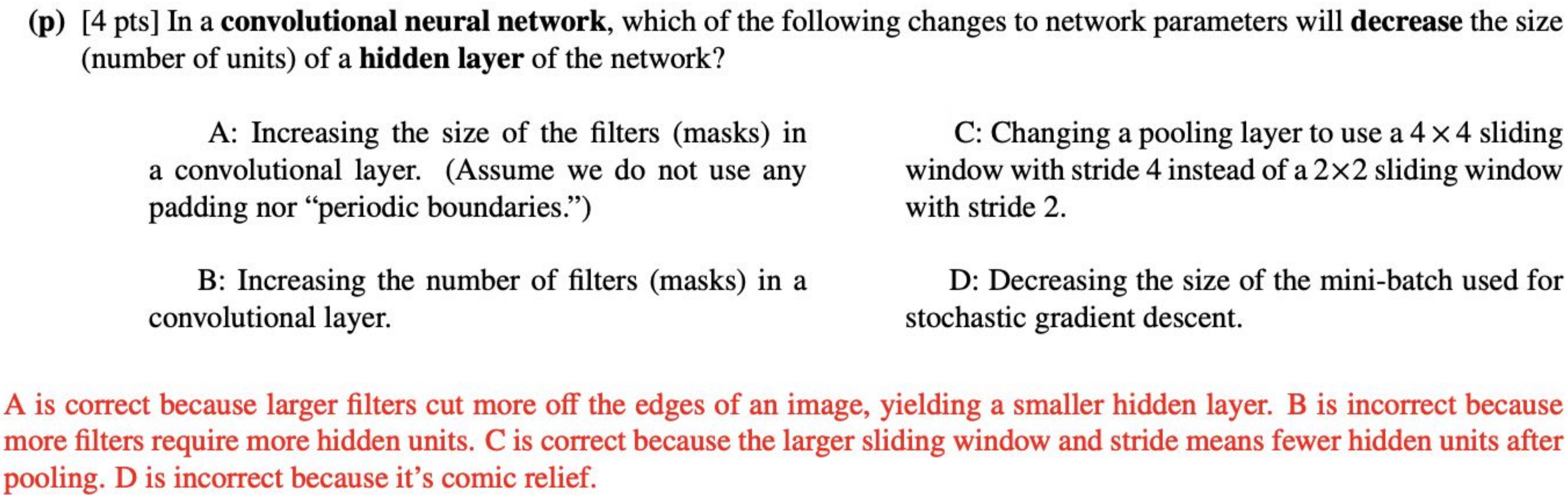
Do we maintain translational equivariance after a max pooling layer?

Solution

Do we maintain translational equivariance after a max pooling layer?

No! Consider translating the smallest element in an image. Max pooling does not mirror this translation.

Practice Question (SP24 Final)



(p) [4 pts] In a **convolutional neural network**, which of the following changes to network parameters will **decrease** the size (number of units) of a **hidden layer** of the network?

A: Increasing the size of the filters (masks) in a convolutional layer. (Assume we do not use any padding nor "periodic boundaries.")

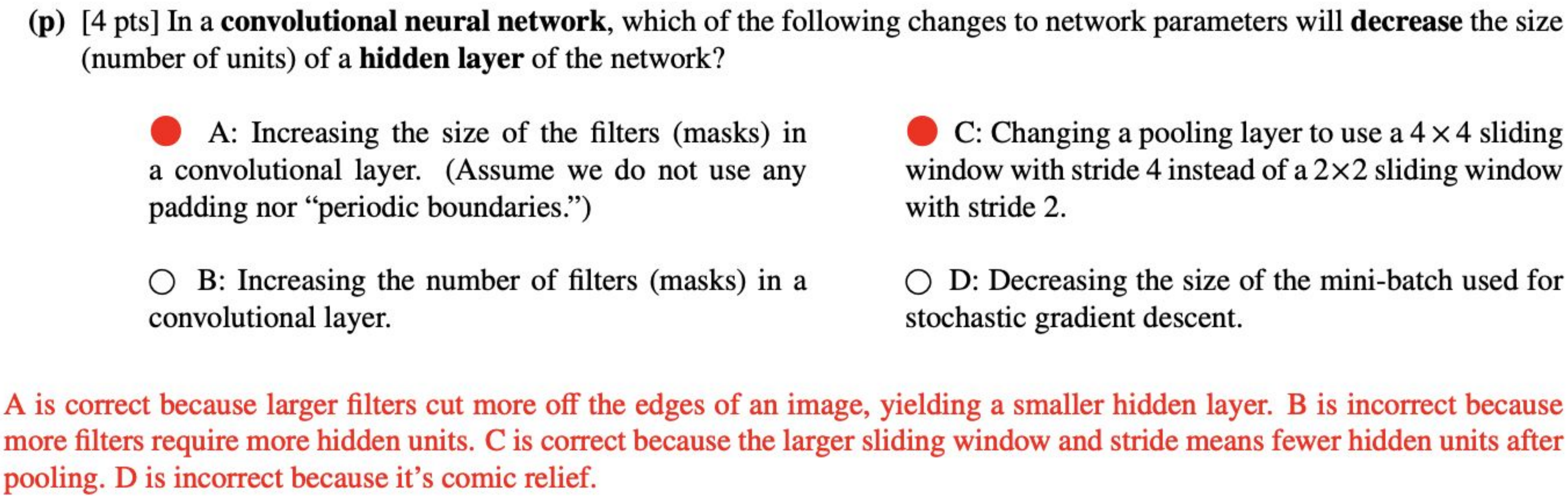
B: Increasing the number of filters (masks) in a convolutional layer.

C: Changing a pooling layer to use a 4 × 4 sliding window with stride 4 instead of a 2×2 sliding window with stride 2.

D: Decreasing the size of the mini-batch used for stochastic gradient descent.

A is correct because larger filters cut more off the edges of an image, yielding a smaller hidden layer. B is incorrect because more filters require more hidden units. C is correct because the larger sliding window and stride means fewer hidden units after pooling. D is incorrect because it's comic relief.

Solution



(p) [4 pts] In a **convolutional neural network**, which of the following changes to network parameters will **decrease** the size (number of units) of a **hidden layer** of the network?

● A: Increasing the size of the filters (masks) in a convolutional layer. (Assume we do not use any padding nor "periodic boundaries.")

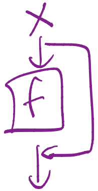
○ B: Increasing the number of filters (masks) in a convolutional layer.

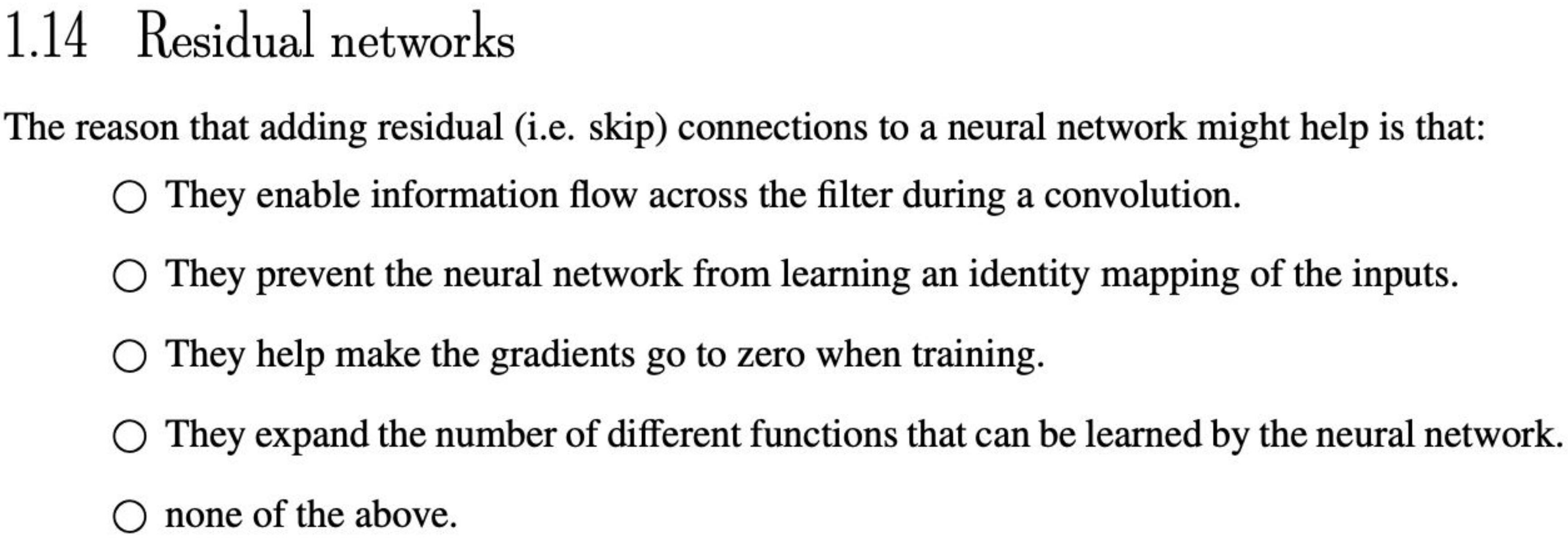
● C: Changing a pooling layer to use a 4 × 4 sliding window with stride 4 instead of a 2×2 sliding window with stride 2.

○ D: Decreasing the size of the mini-batch used for stochastic gradient descent.

A is correct because larger filters cut more off the edges of an image, yielding a smaller hidden layer. B is incorrect because more filters require more hidden units. C is correct because the larger sliding window and stride means fewer hidden units after pooling. D is incorrect because it's comic relief.

Practice Question (FA22 Final)



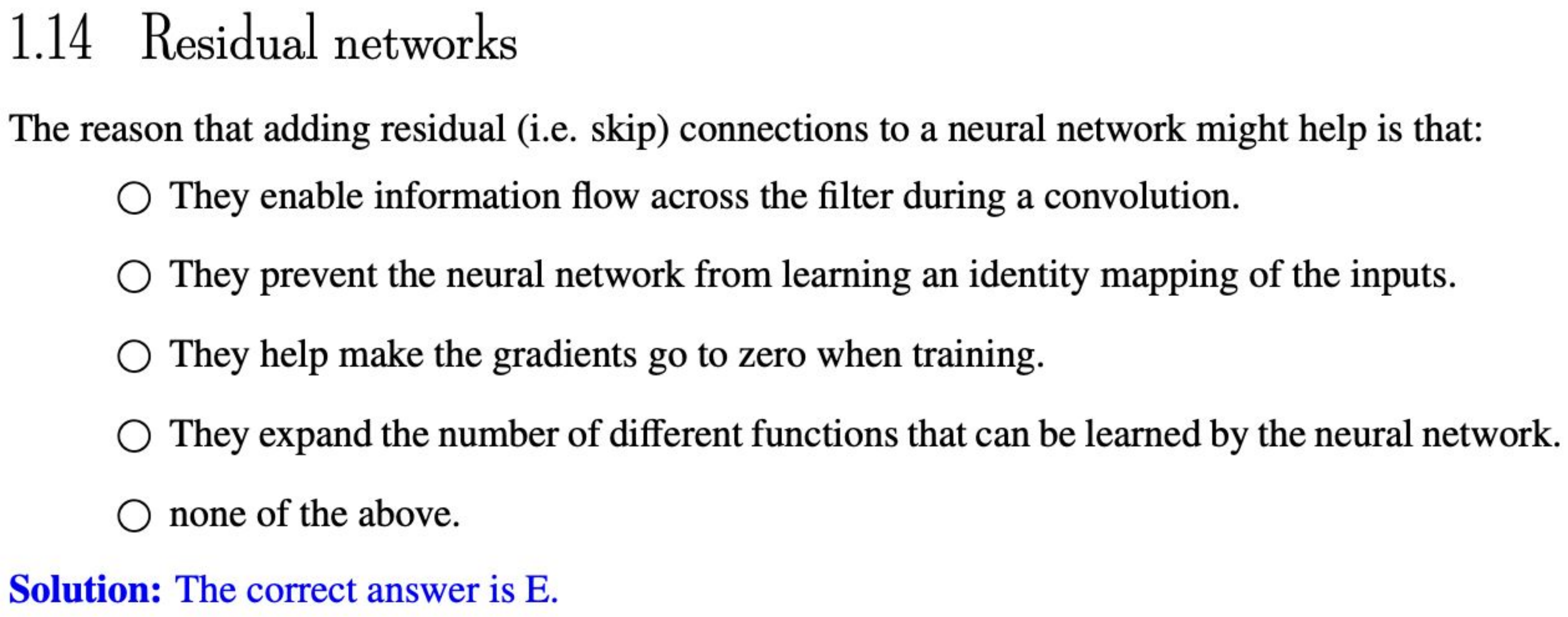


1.14 Residual networks

The reason that adding residual (i.e. skip) connections to a neural network might help is that:

- ○ They enable information flow across the filter during a convolution.
- ○ They prevent the neural network from learning an identity mapping of the inputs.
- ○ They help make the gradients go to zero when training.
- ○ They expand the number of different functions that can be learned by the neural network.
- ○ none of the above.

Solution



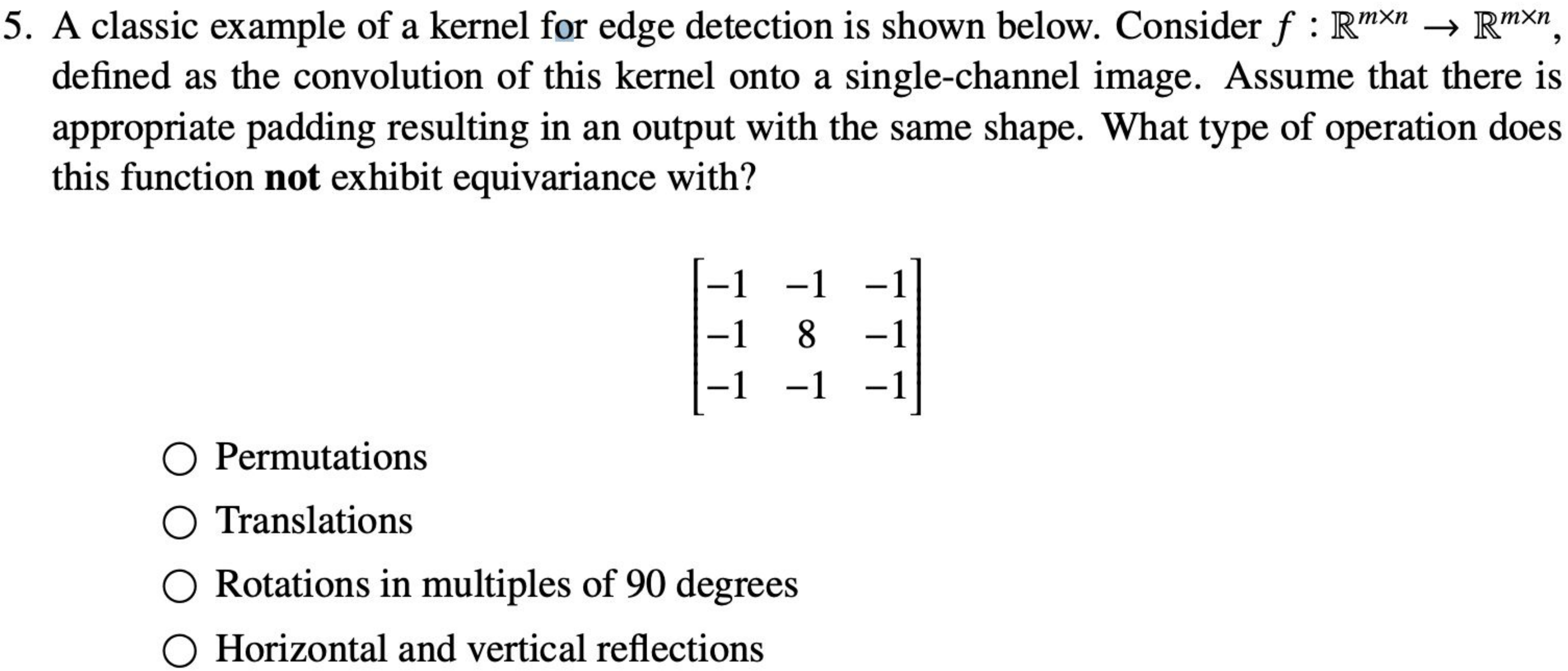
1.14 Residual networks

The reason that adding residual (i.e. skip) connections to a neural network might help is that:

- ○ They enable information flow across the filter during a convolution.
- ○ They prevent the neural network from learning an identity mapping of the inputs.
- ○ They help make the gradients go to zero when training.
- ○ They expand the number of different functions that can be learned by the neural network.
- ○ none of the above.

Solution: The correct answer is E.

Practice Question (FA23 Final)



5. A classic example of a kernel for edge detection is shown below. Consider $f : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}$, defined as the convolution of this kernel onto a single-channel image. Assume that there is appropriate padding resulting in an output with the same shape. What type of operation does this function **not** exhibit equivariance with?

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

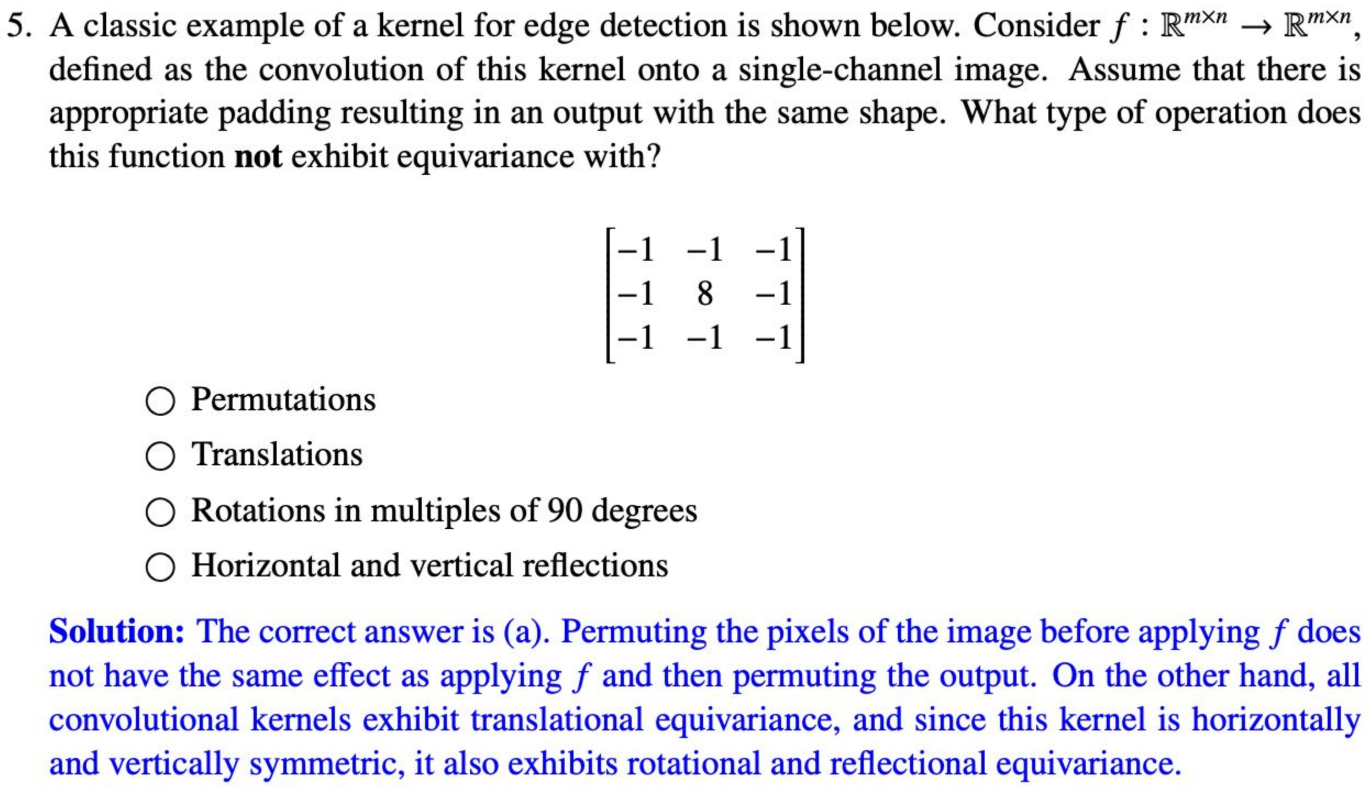
○ Permutations

○ Translations

○ Rotations in multiples of 90 degrees

○ Horizontal and vertical reflections

Solution



5. A classic example of a kernel for edge detection is shown below. Consider $f : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}$, defined as the convolution of this kernel onto a single-channel image. Assume that there is appropriate padding resulting in an output with the same shape. What type of operation does this function **not** exhibit equivariance with?

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

○ Permutations

○ Translations

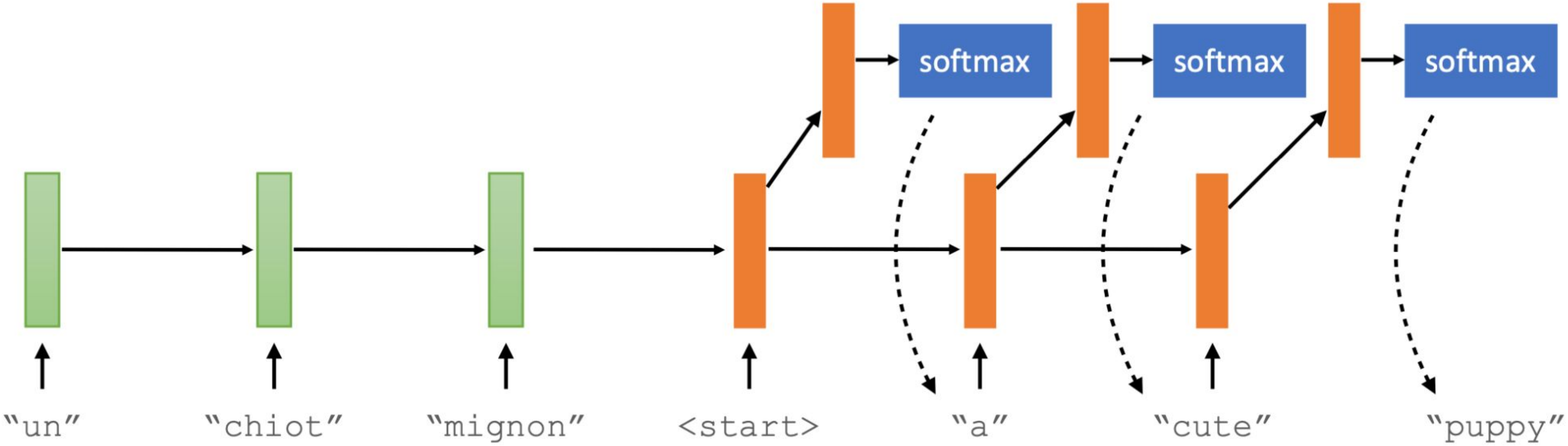
○ Rotations in multiples of 90 degrees

○ Horizontal and vertical reflections

Solution: The correct answer is (a). Permuting the pixels of the image before applying f does not have the same effect as applying f and then permuting the output. On the other hand, all convolutional kernels exhibit translational equivariance, and since this kernel is horizontally and vertically symmetric, it also exhibits rotational and reflectional equivariance.

Sequence Modelling (RNNs)

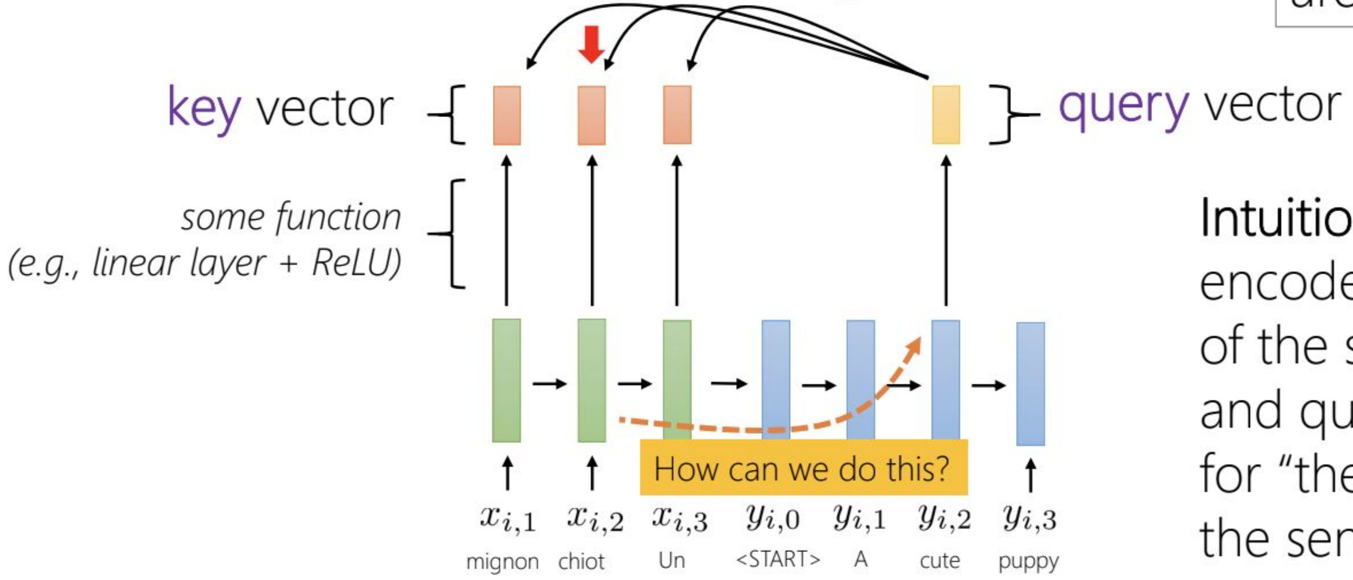
- Can handle arbitrarily long sequences
- Suffer from bad training dynamics (try to derive the gradient)



The Attention Mechanism

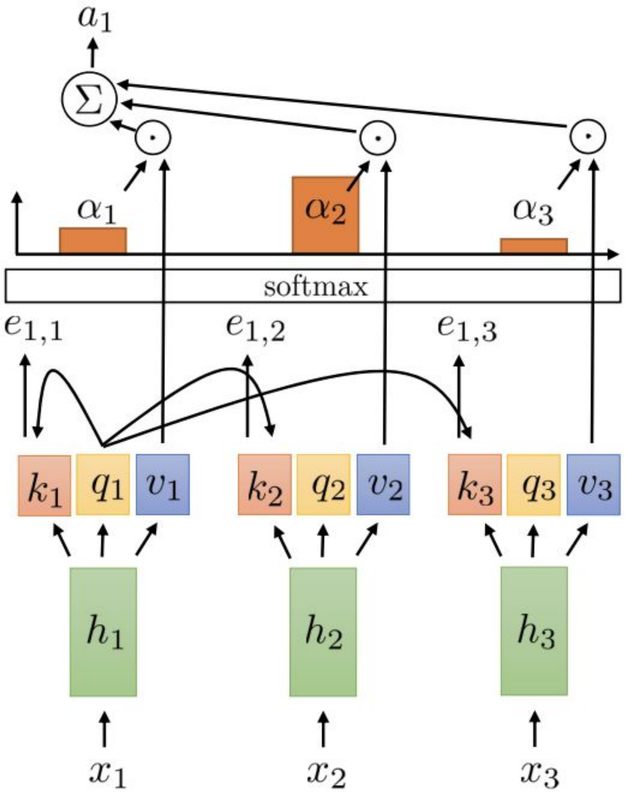
compare query to each key to find the closest one to get the right value

Keys and queries are **learned.**



Intuition: key might encode "the subject of the sentence," and query might ask for "the subject of the sentence".

Self-Attention (One Layer)



$$a_l = \sum_t \alpha_{l,t} v_t$$

$$\alpha_{l,t} = \exp(e_{l,t}) / \sum_{t'} \exp(e_{l,t'})$$

$$e_{l,t} = q_l \cdot k_t$$

we'll see why this is important soon

$v_t = v(h_t)$ before just had $v(h_t) = h_t$, now e.g. $v(h_t) = W_v h_t$

$k_t = k(h_t)$ (just like before) e.g., $k_t = W_k h_t$

$q_t = q(h_t)$ e.g., $q_t = W_q h_t$

this is *not* a recurrent model!

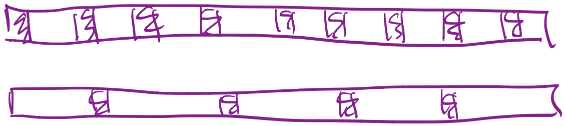
but still weight sharing:

$$h_t = \sigma(W x_t + b)$$

shared weights at all time steps

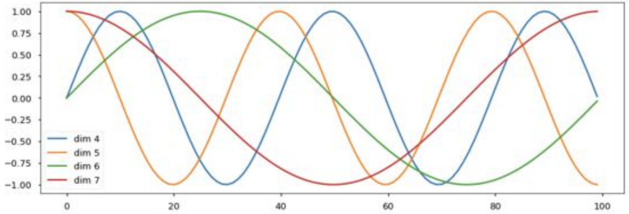
(or any other nonlinear function)

Positional Encodings



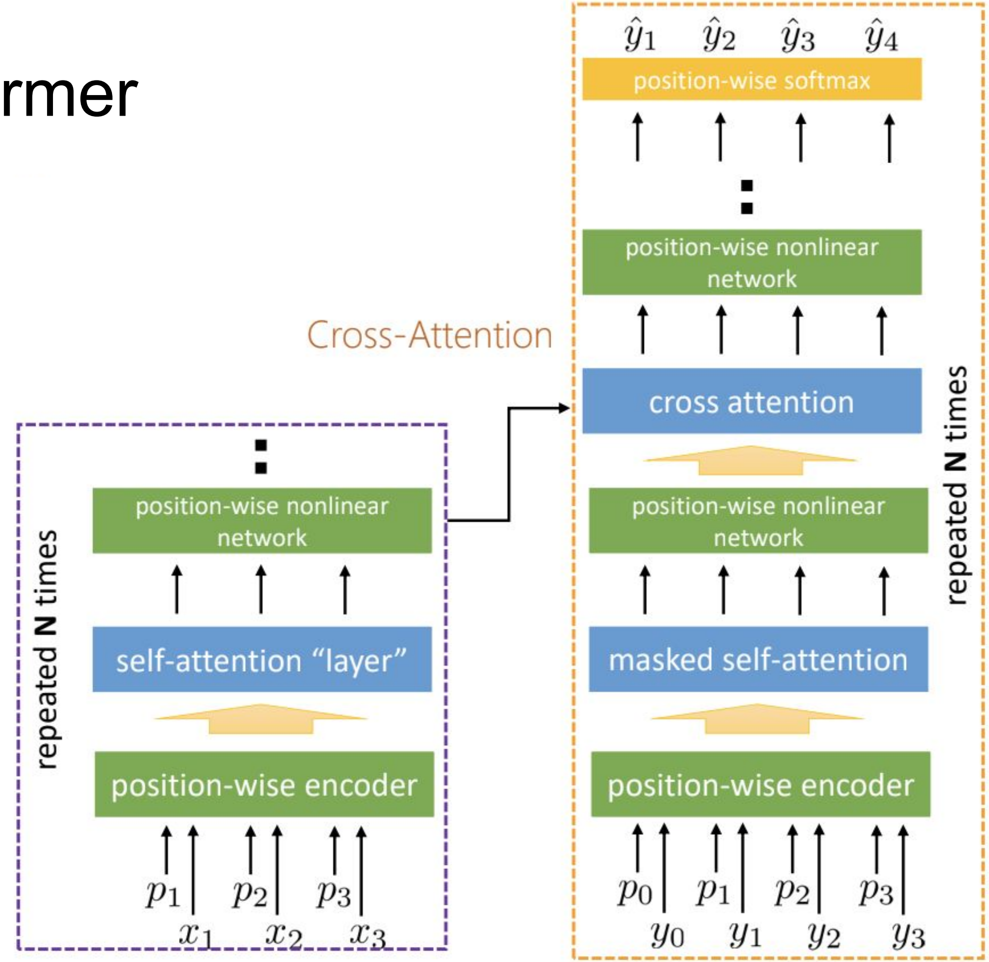
- Attention has no concept of ordering. When you're trying to model a sequence this ordering can be very important.
- We often encode position as a sin transformation of sequence position.

$$p_t = \begin{bmatrix} \sin(t/10000^{2*1/d}) \\ \cos(t/10000^{2*1/d}) \\ \sin(t/10000^{2*2/d}) \\ \cos(t/10000^{2*2/d}) \\ \dots \\ \sin(t/10000^{2*\frac{d}{2}/d}) \\ \cos(t/10000^{2*\frac{d}{2}/d}) \end{bmatrix}$$

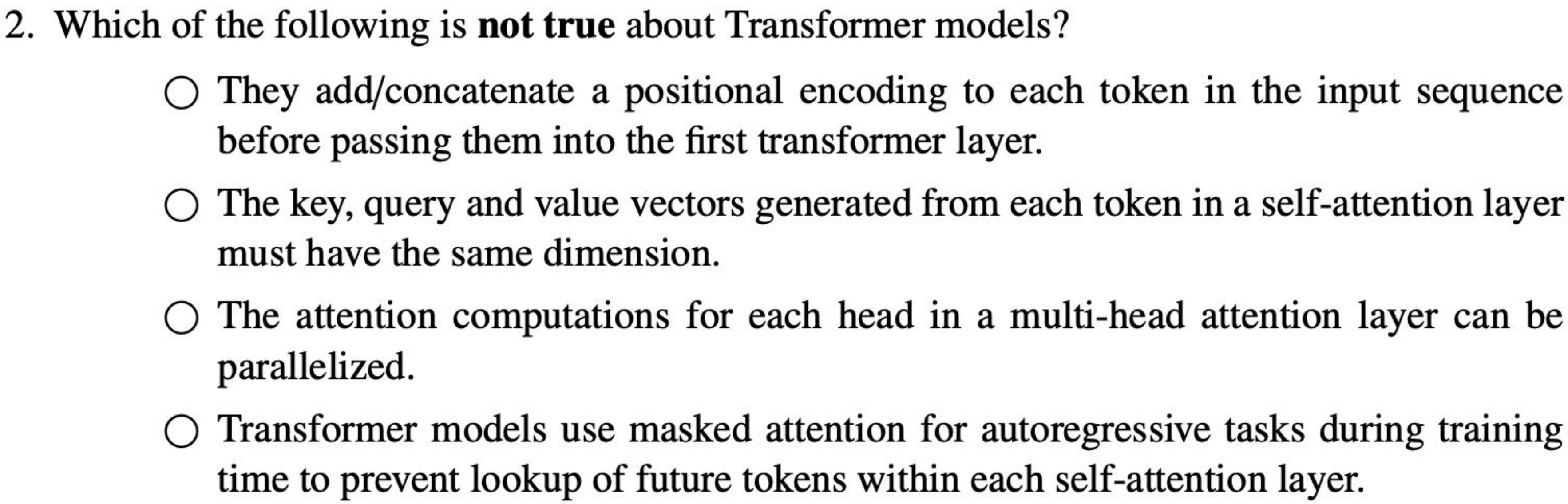


d, is the dimensionality of positional encoding

Encoder-Decoder Transformer



Practice Question (FA23 Final)



2. Which of the following is **not true** about Transformer models?

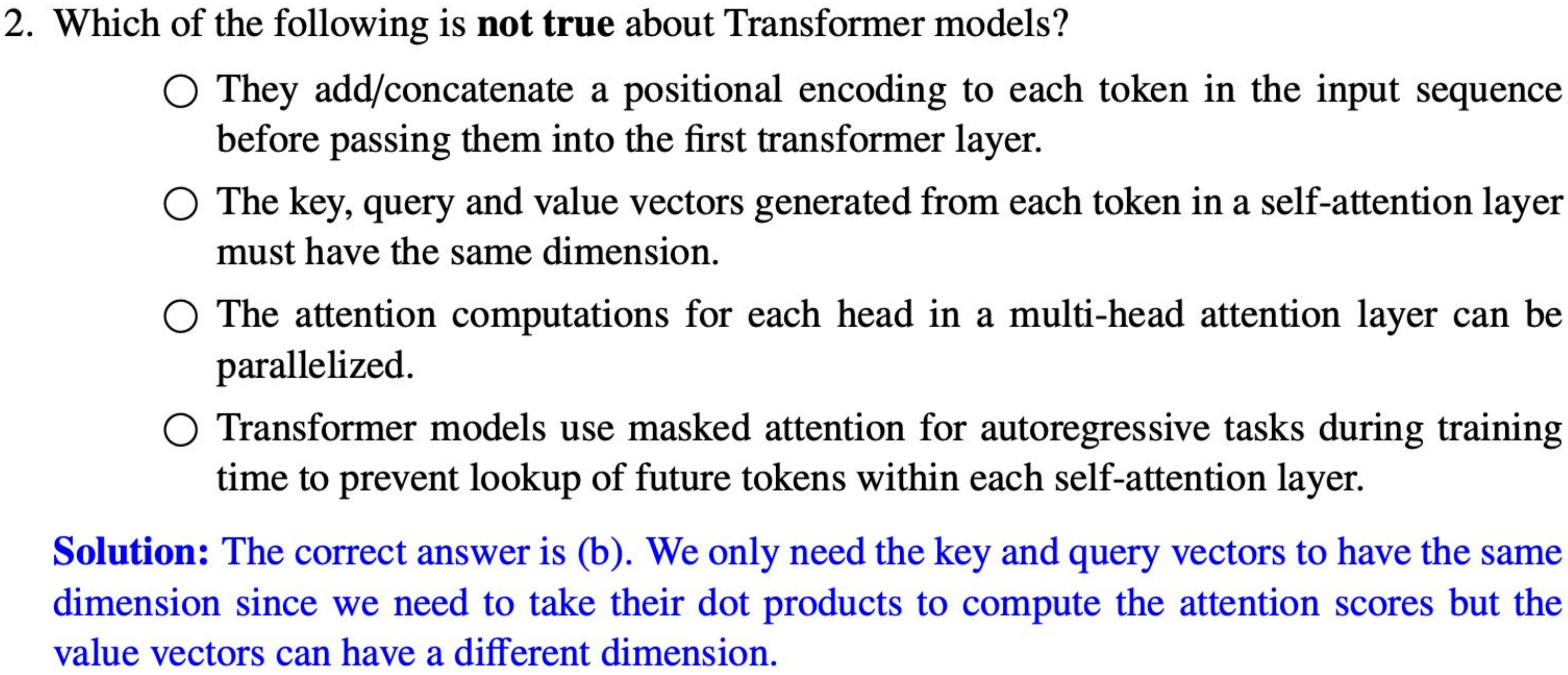
○ They add/concatenate a positional encoding to each token in the input sequence before passing them into the first transformer layer.

○ The key, query and value vectors generated from each token in a self-attention layer must have the same dimension.

○ The attention computations for each head in a multi-head attention layer can be parallelized.

○ Transformer models use masked attention for autoregressive tasks during training time to prevent lookup of future tokens within each self-attention layer.

Solution



2. Which of the following is **not true** about Transformer models?

   ○ They add/concatenate a positional encoding to each token in the input sequence before passing them into the first transformer layer.

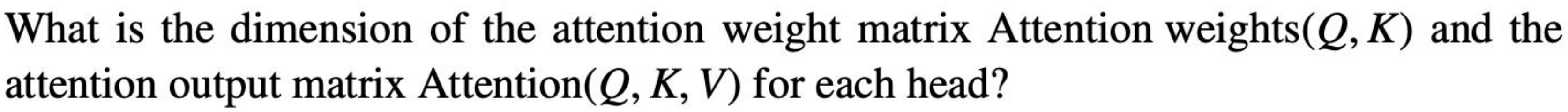
   ○ The key, query and value vectors generated from each token in a self-attention layer must have the same dimension.

   ○ The attention computations for each head in a multi-head attention layer can be parallelized.

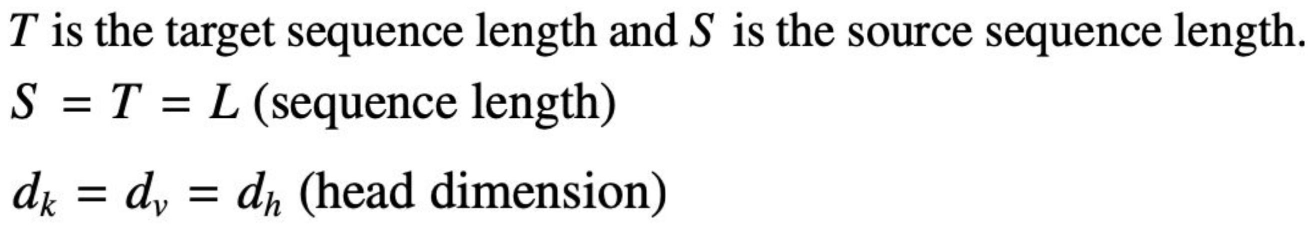
   ○ Transformer models use masked attention for autoregressive tasks during training time to prevent lookup of future tokens within each self-attention layer.

Solution: The correct answer is (b). We only need the key and query vectors to have the same dimension since we need to take their dot products to compute the attention scores but the value vectors can have a different dimension.

Practice Question (FA21 Final)



What is the dimension of the attention weight matrix Attention weights(Q, K) and the attention output matrix Attention(Q, K, V) for each head?



T is the target sequence length and S is the source sequence length.

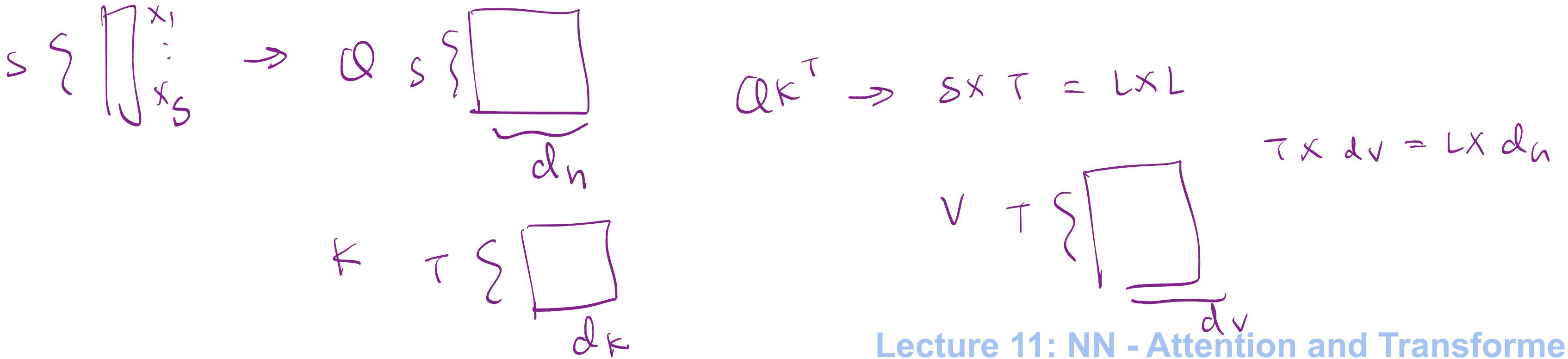
$S = T = L$ (sequence length)

$d_k = d_v = d_h$ (head dimension)

Practice Question (FA21 Final)

What is the dimension of the attention weight matrix Attention weights(Q, K) and the attention output matrix Attention(Q, K, V) for each head?

Solution: The attention weight matrix Attention weights(Q, K) has dimensions $T \times S = L \times L$. The attention output matrix Attention(Q, K, V) has dimensions $T \times d_v = L \times d_h$.


s
x_1
x_s
Q
s
d_h
K
T
d_k
$QK^T \to S \times T = L \times L$
V
T
d_v
$T \times d_v = L \times d_h$

Dimensionality Reduction + Clustering

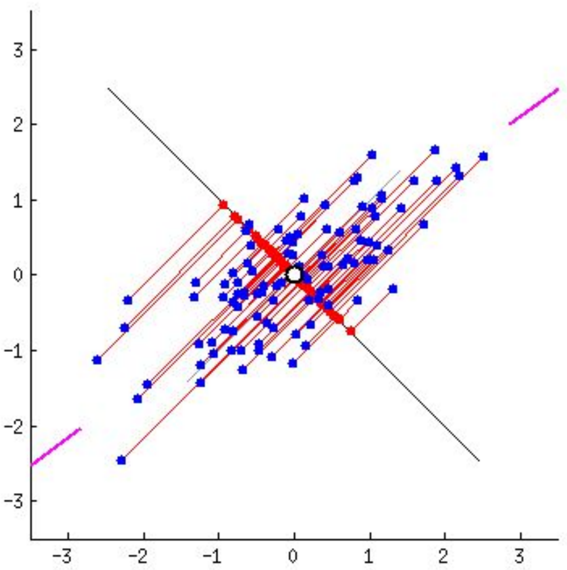
- Very related but not replacements for each other
  - usually go hand-in-hand
  - first we reduce dimensions of data, then cluster
- Dimensionality Reduction
  - PCA decomposition
  - t-SNE projection
- Clustering
  - K-means

Principal Component Analysis (PCA)

First, we want a new axis which has the shortest distance to all data.

Next we want to find the second best to the first criterion but subject to being orthogonal to previous axes.

And so on...

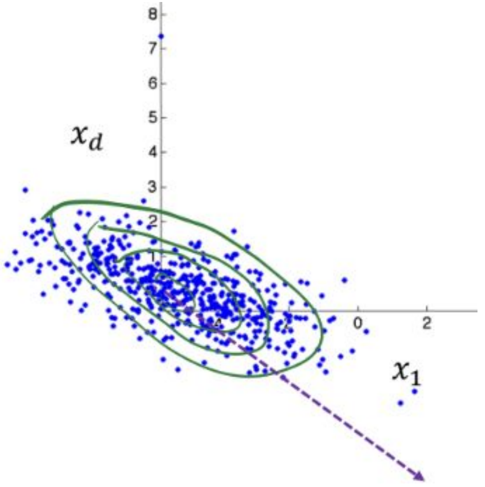


PCA: Algorithm 1 (via Spectral Decomposition)

Given n data points of dimension d, $X \in \mathbb{R}^{n\times d}$, to perform PCA, we:

2. Compute the covariance matrix, ~~$\Sigma = \bar{X}^T\bar{X}$.~~ $\Sigma = \frac{1}{n} X^T X$ (although scaling doesn't matter if we only need PCs)

3. Compute $\bar{X}^T\bar{X} = QDQ^T$ to get eigenvectors (aka ~~principal directions~~ principal components).



4. Keep the k eigenvectors, $\boxed{Q_k \equiv Q_{:,1:k},}$ with the most variance (highest eigenvalues in D).

5. Project your points (original, or new ones) down to this subspace, $\bar{X}_k = \boxed{\bar{X}Q_k} \in \mathbb{R}^{n\times k}$, these are your ~~principal components~~. PC scores.

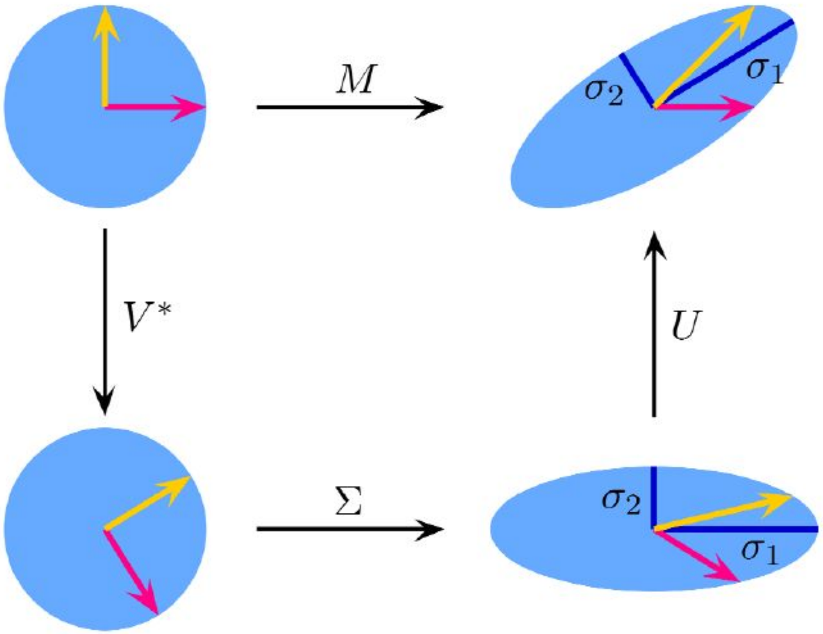
$$\begin{bmatrix} - x_1 - \\ \vdots \\ - x_n - \end{bmatrix} \begin{bmatrix} | & & | \\ q_1 & \cdots & q_k \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ Xq_1 & \cdots & Xq_k \\ | & & | \end{bmatrix}$$

PCA: Algorithm 2 (via SVD)

1. Pre-process: Subtract mean, Normalize variance if needed.

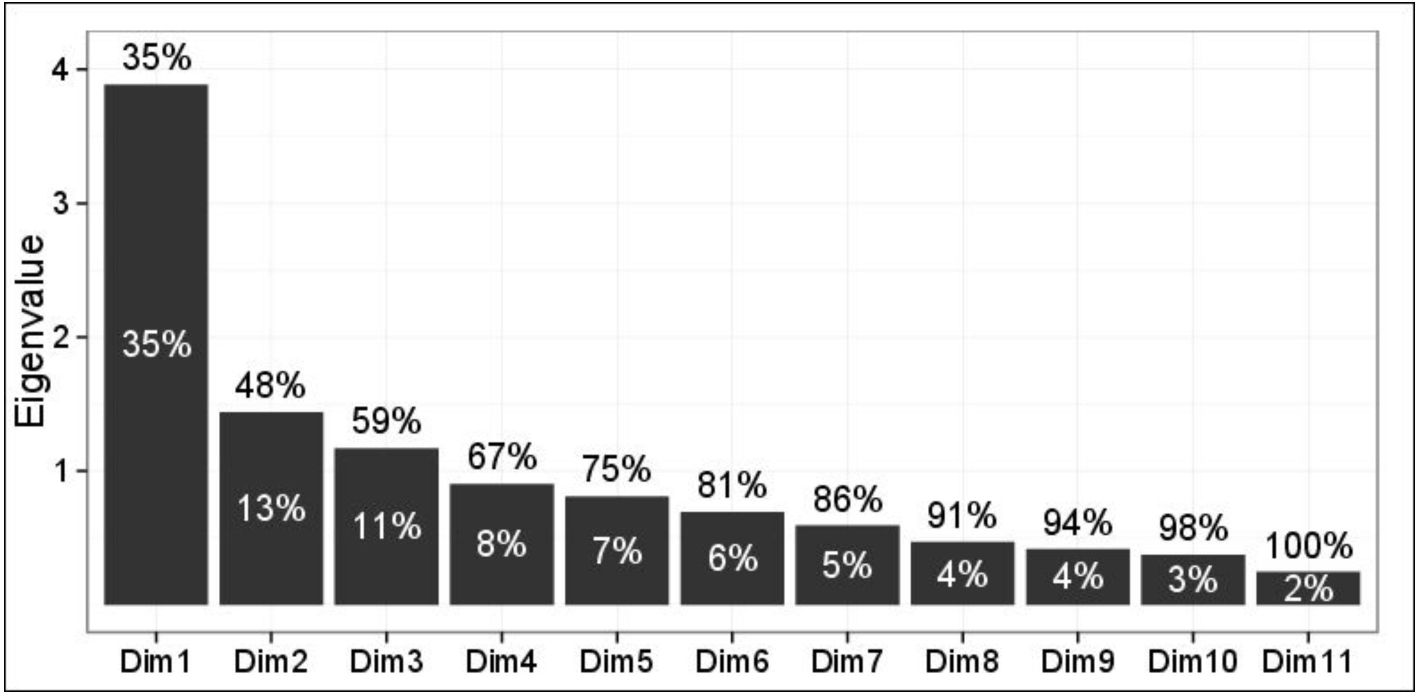
2. Calculate SVD of X.
X = UDV’

3.The new basis is just the first k right singular vectors.
(first k columns of V)



$$M = U \cdot \Sigma \cdot V^*$$

PCA: Metric – Explained variance



$$X^TX = QDQ^T$$

$$V_i \equiv \frac{D_{i,i}}{\boxed{\sum_j D_{j,j}}}$$

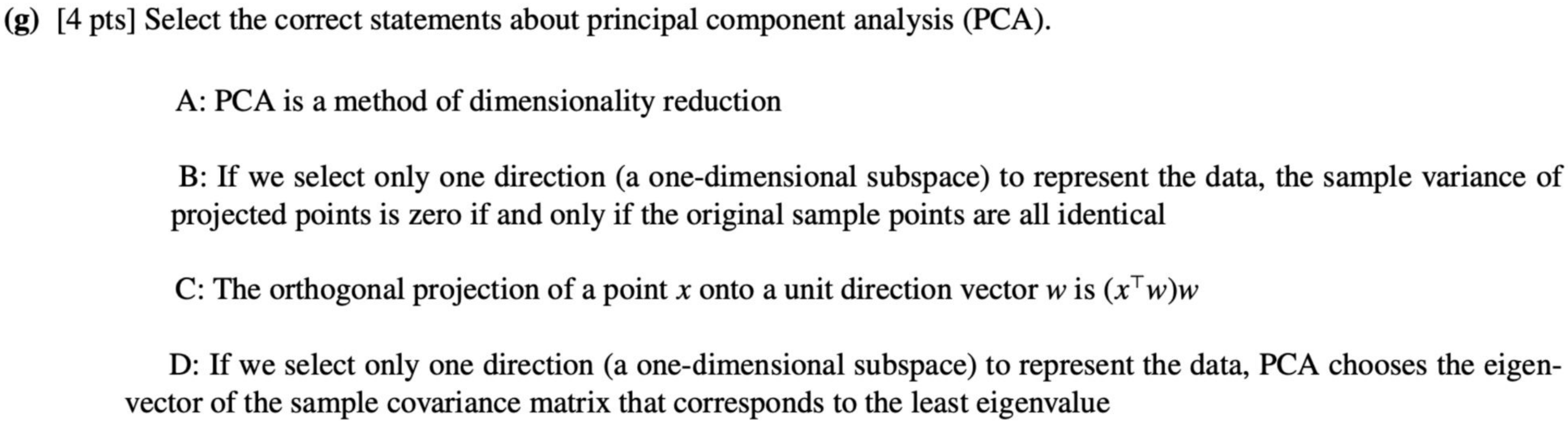
PCA: Limitations and Extensions

- Works best when data follows Gaussian distribution
  - PCA assumes only mean and variance affect distribution.
  - Other distributions may result in poor fitting
- Limited to linear relationships between feats
  - One solution is Kernel PCA
- Isn't robust to scale of features or outliers

(But you can, and should normalize you data prior to PCA — so the scaling issue can be fixed.)

- De-mean your data when you do PCA!
  (make sure features have zero-mean.)

Practice Question (SP21 Final)



(g) [4 pts] Select the correct statements about principal component analysis (PCA).

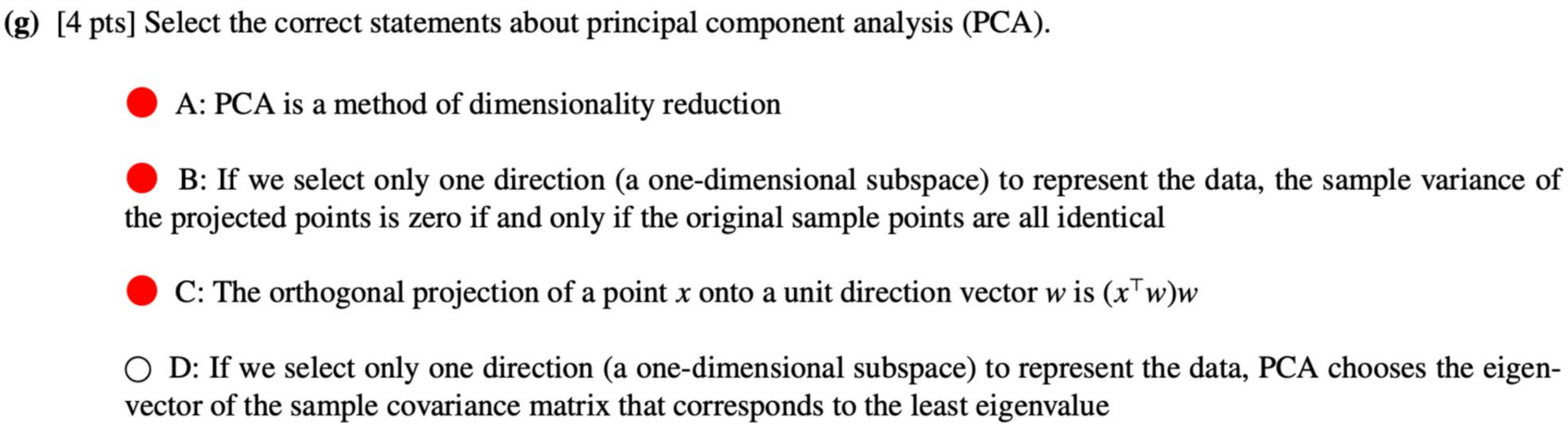
A: PCA is a method of dimensionality reduction

B: If we select only one direction (a one-dimensional subspace) to represent the data, the sample variance of projected points is zero if and only if the original sample points are all identical

C: The orthogonal projection of a point x onto a unit direction vector w is $(x^\top w)w$

D: If we select only one direction (a one-dimensional subspace) to represent the data, PCA chooses the eigenvector of the sample covariance matrix that corresponds to the least eigenvalue

Solution



(g) [4 pts] Select the correct statements about principal component analysis (PCA).

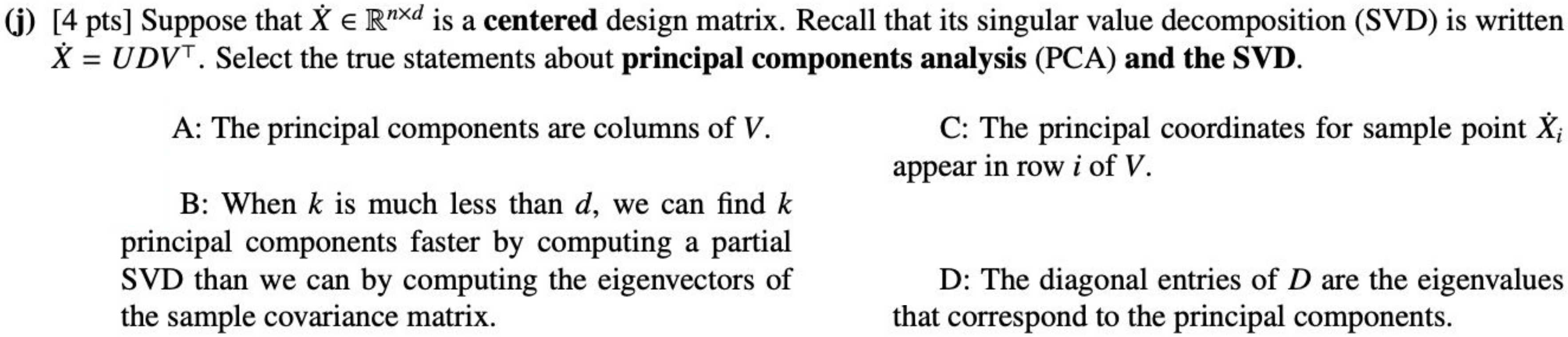
● A: PCA is a method of dimensionality reduction

● B: If we select only one direction (a one-dimensional subspace) to represent the data, the sample variance of the projected points is zero if and only if the original sample points are all identical

● C: The orthogonal projection of a point x onto a unit direction vector w is $(x^\top w)w$

○ D: If we select only one direction (a one-dimensional subspace) to represent the data, PCA chooses the eigenvector of the sample covariance matrix that corresponds to the least eigenvalue

Practice Question (SP24 Final)



(j) [4 pts] Suppose that $\dot{X} \in \mathbb{R}^{n\times d}$ is a **centered** design matrix. Recall that its singular value decomposition (SVD) is written $\dot{X} = UDV^\top$. Select the true statements about **principal components analysis** (PCA) **and the SVD**.

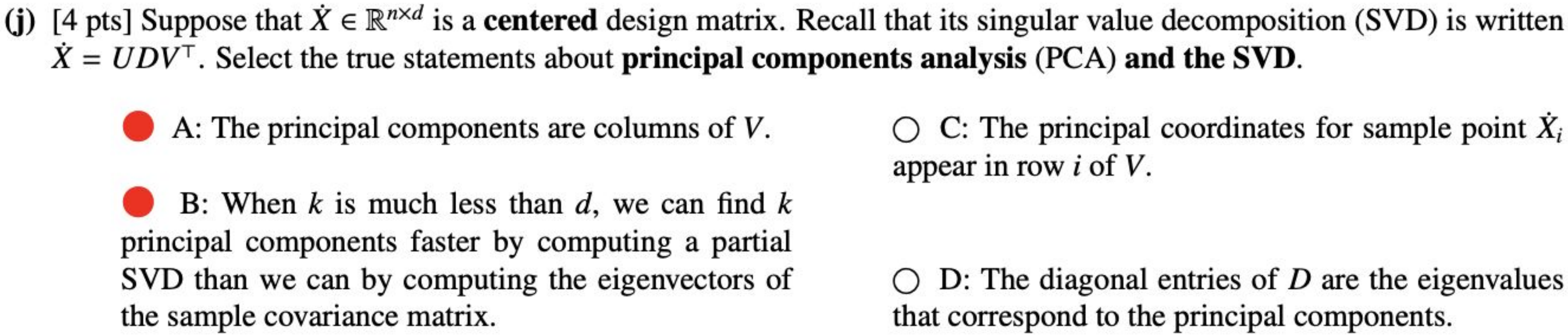
A: The principal components are columns of V.

B: When k is much less than d, we can find k principal components faster by computing a partial SVD than we can by computing the eigenvectors of the sample covariance matrix.

C: The principal coordinates for sample point $\dot{X}_i$ appear in row i of V.

D: The diagonal entries of D are the eigenvalues that correspond to the principal components.

Solution



(j) [4 pts] Suppose that $\dot{X} \in \mathbb{R}^{n \times d}$ is a **centered** design matrix. Recall that its singular value decomposition (SVD) is written $\dot{X} = UDV^\top$. Select the true statements about **principal components analysis** (PCA) **and the SVD**.

● A: The principal components are columns of V.

● B: When k is much less than d, we can find k principal components faster by computing a partial SVD than we can by computing the eigenvectors of the sample covariance matrix.

○ C: The principal coordinates for sample point $\dot{X}_i$ appear in row i of V.

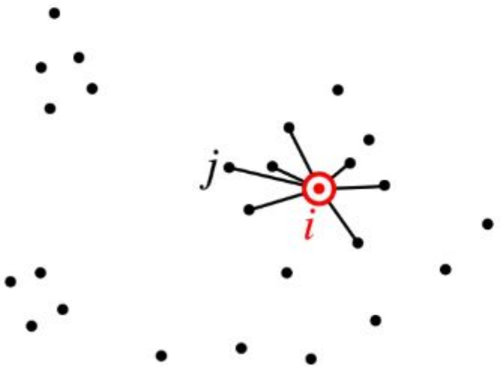
○ D: The diagonal entries of D are the eigenvalues that correspond to the principal components.

t-distributed Stochastic Neighbor Embedding (t-SNE)

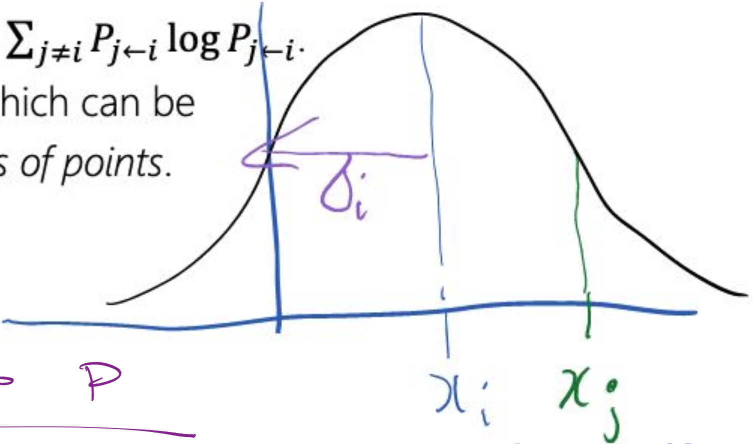
- PCA models the global structure in data like the major principal axes and the explained variances.
- t-SNE models the local data structure ie. the neighbourhood of each data point and not the overall pattern itself.
  - Use if non-linear relationships define clusters
  - Can yield orthogonal insights to PCA
  - Allows visualizing very high dimensions

From NE to *stochastic NE* (SNE)

- Now make the event of two samples being neighbors a random variable:
- The probability that $\boldsymbol{x}_i$ "chooses" $\boldsymbol{x}_j$ as its neighbor ($\boldsymbol{x} \in \mathbb{R}^d$) is given by $P_{j\leftarrow i} = \frac{exp\left(-\|x_i - x_j\|^2/2\sigma_i^2\right)}{\sum_{k\neq i} exp\left(-\|x_i - x_k\|^2/2\sigma_i^2\right)}$, and $P_{j\leftarrow j} = 0$
- Smells like a Gaussian, but normalized so that $1 = \sum_j P_{j\leftarrow i}$.



- Set σ_i^2 adaptively such that entropy of $P_{:\leftarrow i}$ is constant, $\sum_{j\neq i} P_{j\leftarrow i} \log P_{j\leftarrow i}$.
- Symmetrize & normalize, $P_{ij} = P_{ji} = \frac{1}{2n}\left(P_{j\leftarrow i} + P_{i\leftarrow j}\right)$ which can be interpreted as *probability to pick this pair out of all pairs of points*.



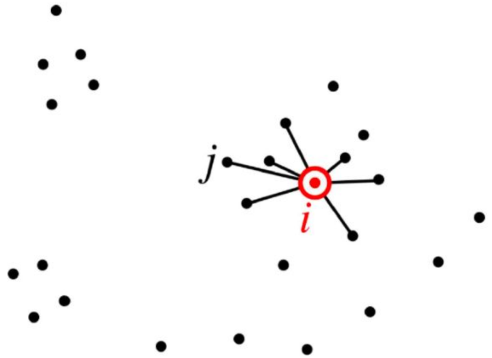
$\mathbb{R}^d \ni x_i \mapsto y_i \in \mathbb{R}^k$ $\quad k < d$

$P_{ij} \qquad Q_{ij}$

Goal: make Q similar to P

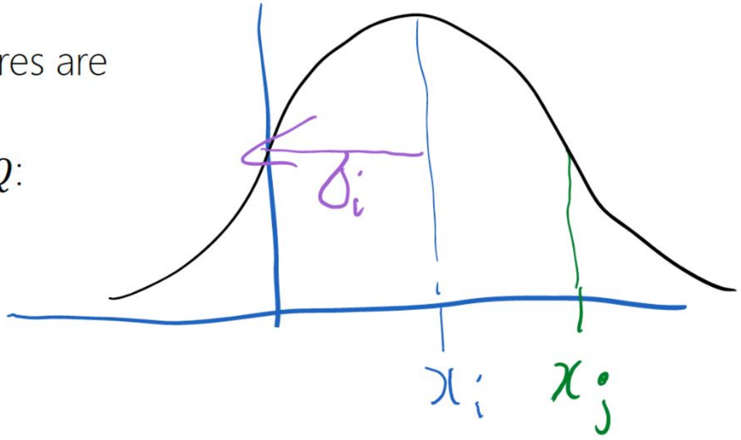
From NE to *stochastic NE* (SNE)

$$P_{j\leftarrow i} = \frac{exp\left(-\|x_i - x_j\|^2/2\sigma_i^2\right)}{\sum_{k\neq i} exp(-\|x_i - x_k\|^2/2\sigma_i^2)}$$



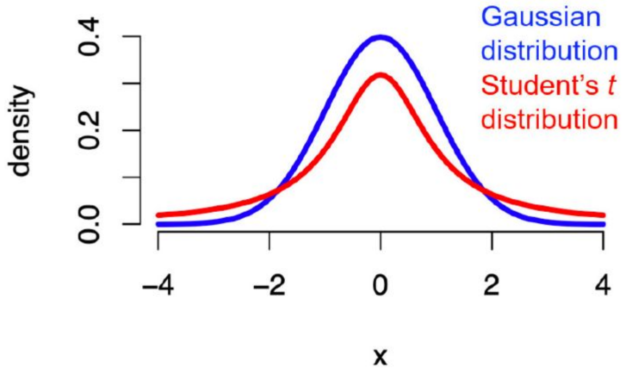
- From original data, $X \in \mathbb{R}^{n\times d}$, we have defined stochastic neighborhoods with probability distribution, $P = \{P_{ij}\}$.
- Now posit low-dimensional representations, $Y \in \mathbb{R}^{n\times k}$, and define stochastic neighborhoods for them, $Q = \{Q_{ij}\}$,
- $Q_{ji} = \frac{exp\left(-\|y_i - y_j\|^2\right)}{\sum_{l\neq k} exp(-\|y_l - y_k\|^2)}$ (setting $\sigma^2 = \frac{1}{2}$)
- Goal: find Y such that stochastic neighborhood structures are preserved ($Q \approx P$).
- Solution: minimize the KL-divergence between P and Q:
- $\boxed{\hat{Y} = \underset{Y}{\text{argmin}}\, KL(P||Q) = \sum_{i,j} P_{ij}\, log\, \frac{P_{ij}}{Q_{ij}}.}$

$\rightarrow$ make Q similar to P



Fixing SNE with t-SNE

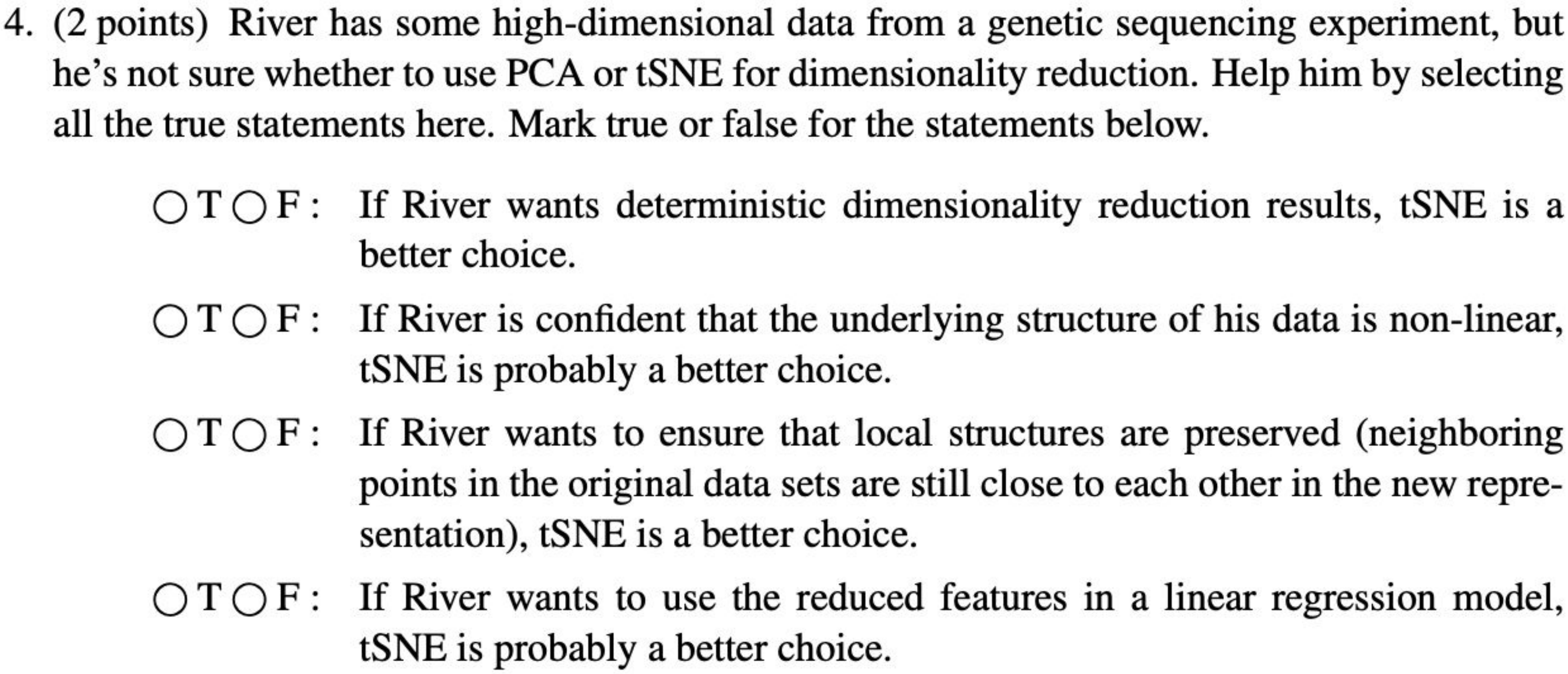
- Change the distribution in the embedding space, Q_{ij} to have a heavier-tailed distribution, a t-distribution.
- SNE used $Q_{j\leftarrow i} = \frac{exp\left(-\|y_i - y_j\|^2\right)}{\sum_{i\neq k} exp(-\|y_i - y_k\|^2)}$.
- t-SNE uses $Q_{j\leftarrow i} = \frac{\left(\|y_i - y_j\|^2\right)^{-1}}{\sum_{i\neq k}(\|y_i - y_k\|^2)^{-1}}$.
- Everything else remains the same as in SNE.



$$p(x) \propto \left(1 + \frac{x^2}{v}\right)^{-(v+1)/2}$$

for $v = 1$ we get $p(x) \propto \frac{1}{1+x^2}$

Practice Question (FA21 Final)



4. (2 points) River has some high-dimensional data from a genetic sequencing experiment, but he's not sure whether to use PCA or tSNE for dimensionality reduction. Help him by selecting all the true statements here. Mark true or false for the statements below.

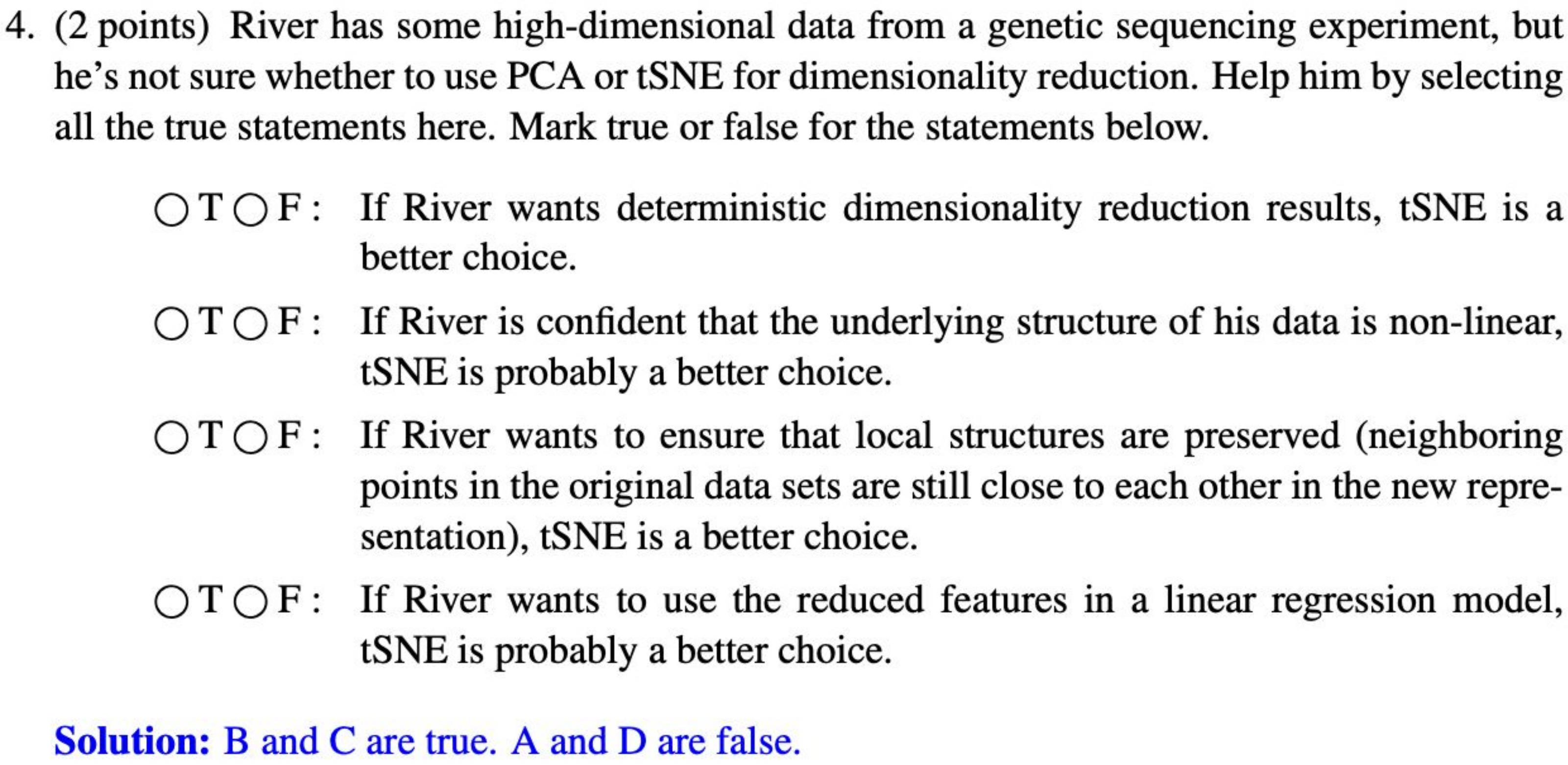
○ T ○ F : If River wants deterministic dimensionality reduction results, tSNE is a better choice.

○ T ○ F : If River is confident that the underlying structure of his data is non-linear, tSNE is probably a better choice.

○ T ○ F : If River wants to ensure that local structures are preserved (neighboring points in the original data sets are still close to each other in the new representation), tSNE is a better choice.

○ T ○ F : If River wants to use the reduced features in a linear regression model, tSNE is probably a better choice.

Solution



4. (2 points) River has some high-dimensional data from a genetic sequencing experiment, but he's not sure whether to use PCA or tSNE for dimensionality reduction. Help him by selecting all the true statements here. Mark true or false for the statements below.

   ○ T ○ F : If River wants deterministic dimensionality reduction results, tSNE is a better choice.

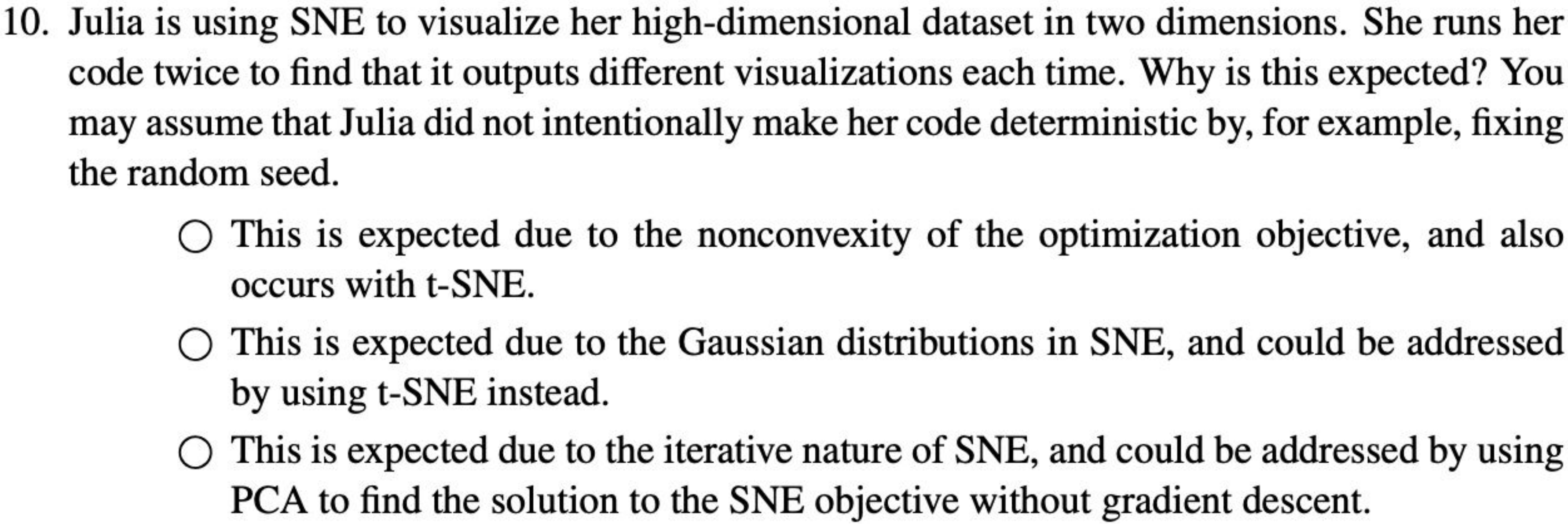
   ○ T ○ F : If River is confident that the underlying structure of his data is non-linear, tSNE is probably a better choice.

   ○ T ○ F : If River wants to ensure that local structures are preserved (neighboring points in the original data sets are still close to each other in the new representation), tSNE is a better choice.

   ○ T ○ F : If River wants to use the reduced features in a linear regression model, tSNE is probably a better choice.

Solution: B and C are true. A and D are false.

Practice Question (FA22 Midterm)



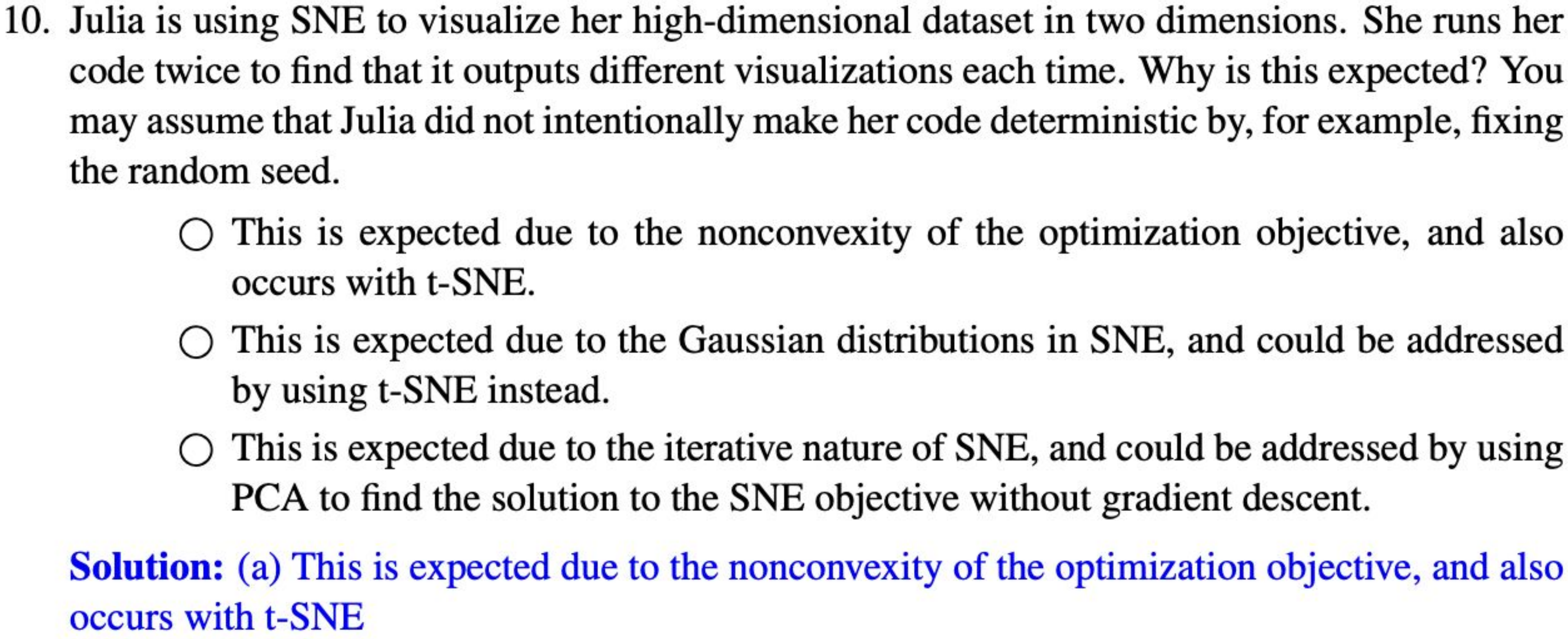
10. Julia is using SNE to visualize her high-dimensional dataset in two dimensions. She runs her code twice to find that it outputs different visualizations each time. Why is this expected? You may assume that Julia did not intentionally make her code deterministic by, for example, fixing the random seed.

 ○ This is expected due to the nonconvexity of the optimization objective, and also occurs with t-SNE.

 ○ This is expected due to the Gaussian distributions in SNE, and could be addressed by using t-SNE instead.

 ○ This is expected due to the iterative nature of SNE, and could be addressed by using PCA to find the solution to the SNE objective without gradient descent.

Solution



10. Julia is using SNE to visualize her high-dimensional dataset in two dimensions. She runs her code twice to find that it outputs different visualizations each time. Why is this expected? You may assume that Julia did not intentionally make her code deterministic by, for example, fixing the random seed.

   ○ This is expected due to the nonconvexity of the optimization objective, and also occurs with t-SNE.

   ○ This is expected due to the Gaussian distributions in SNE, and could be addressed by using t-SNE instead.

   ○ This is expected due to the iterative nature of SNE, and could be addressed by using PCA to find the solution to the SNE objective without gradient descent.

Solution: (a) This is expected due to the nonconvexity of the optimization objective, and also occurs with t-SNE

Conclusion

Some General Study Tips

- Linear Algebra is in general very important for machine learning
  - Advice: review linear algebra from HW1, HW2 and ensure that you are comfortable with manipulating matrices
- Know how to take derivatives
  - We may ask you to derive some gradients, try not to get too bogged down and use your sanity checks: gradients wrt scalars should be the same shape as the denominator.
  - Element-wise always works!
- Review probability theory
  - Basic properties of expectation and variance (like linearity and independence rules will be helpful).
  - Understanding MLE and MAP estimation will also be useful
- Relate concepts back to first principles
  - It will be generally helpful to have a strong grasp of the basics (linear+logistic regression, gradient descent) to guide your approach to more complicated problems
- Get a good night's sleep!

Inspo and Such

These slides were heavily inspired (and in some cases directly taken) from CS182 SP22 review materials:

https://cs182sp22.github.io/assets/lecture_slides/2022.02.28-mt1-review.pdf

https://cs182sp22.github.io/assets/lecture_slides/2022.04.25-mt2-review.pdf

We also reused materials from the previous Fall Iteration of 189.