CSCI 6521: Advanced Machine Learning I
Study Guide for Test#2

(Please don't distribute this study guide.
The guide is for your study purpose only)

1. Assume for a binary class classification modeling the posterior class probabilities are given by:

$$\textbf{Pr}(G = 1|X = x) = \frac{\exp(\beta_0 + \beta^T x)}{1+\exp(\beta_0 + \beta^T x)},$$

$$\textbf{Pr}(G = 2|X = x) = \frac{1}{1+\exp(\beta_0 + \beta^T x)}.$$

How will you derive their corresponding *logit* transformation? Explain.

Ans: The *logit* or *log-odds* of a probability p is given by, $logit(p) = \log\left(\frac{p}{1-p}\right)$. Therefore, given $p = \Pr(G = 1|X = x) = \frac{\exp(\beta_0 + \beta^T x)}{1+\exp(\beta_0 + \beta^T x)}$, we can have,

$(1\text{-}p) = \left(1-\frac{\exp(\beta_0 + \beta^T x)}{1+\exp(\beta_0 + \beta^T x)}\right) = \frac{1}{1+\exp(\beta_0 + \beta^T x)}$, which is basically, $\Pr(G = 2|X = x)$.

Thus, we can compute the log $[p/(1-p)]$ as following:

$$\log\frac{\Pr(G=1\mid X=x)}{\Pr(G=2\mid X=x)} = \beta_0 + \beta^T x$$

2. Suppose $f_k(x)$ is the class-conditional density of X in class $G = k$, and let π_k be the prior probability of class k, with $\sum_{k=1}^{K}\pi_k = 1$. A simple application of Bayes theorem gives us:

$$\textbf{Pr}(G = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}.$$

Further, assume we model each class density as multivariate Gaussian, that is,

$$f_k(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}\left|\mathbf{\Sigma}_k\right|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \mathbf{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}$$, **where *p* is the feature dimension.**

Now, to compare two classes, *k* and *ℓ*,

(a) show that,

$$\textit{Logit} = -\frac{1}{2}[x^T(\Sigma_k^{-1}-\Sigma_l^{-1})x + 2(\mu_l^T\Sigma_l^{-1}-\mu_k^T\Sigma_k^{-1})x + \left(\mu_k^T\Sigma_k^{-1}\mu_k - \mu_l^T\Sigma_l^{-1}\mu_l + \log\frac{|\Sigma_k|}{|\Sigma_l|} - 2\log\frac{\pi_k}{\pi_l}\right)]$$

(b) Further, if we assume $\Sigma_k = \Sigma_l = \Sigma$, then the logit becomes:

$$\textit{logit} = \log\frac{\pi_k}{\pi_l} - \frac{1}{2}\left(\mu_k^T\Sigma^{-1}\mu_k - \mu_l^T\Sigma^{-1}\mu_l\right) + x^T\Sigma^{-1}(\mu_k-\mu_l).$$

(c) Given the *logit* = $\log\frac{\pi_k}{\pi_l} - \frac{1}{2}\left(\mu_k^T\Sigma^{-1}\mu_k - \mu_l^T\Sigma^{-1}\mu_l\right) + x^T\Sigma^{-1}(\mu_k-\mu_l)$**, it can also be presented as:**

$$\textit{logit} = \log\frac{\pi_k}{\pi_l} - \frac{1}{2}\left(\mu_k+\mu_l\right)^T\Sigma^{-1}\left(\mu_k-\mu_l\right) + x^T\Sigma^{-1}(\mu_k-\mu_l)$$**, show how?**

Ans: (a)

$$\textit{logit} = \log\frac{\Pr(G=k \mid X=x)}{\Pr(G=l \mid X=x)} = \log\frac{f_k(x)}{f_l(x)} + \log\frac{\pi_k}{\pi_l}$$

$$= \log\frac{\pi_k}{\pi_l} + \log f_k(x) - \log f_l(x)$$

$$= \log\frac{\pi_k}{\pi_l} + \left[-\frac{p}{2}\log(2\pi) - \frac{1}{2}\log(|\Sigma_k|) - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\right]$$
$$-\left[-\frac{p}{2}\log(2\pi) - \frac{1}{2}\log(|\Sigma_l|) - \frac{1}{2}(x-\mu_l)^T\Sigma_l^{-1}(x-\mu_l)\right]$$

$$= \log\frac{\pi_k}{\pi_l} - \frac{1}{2}\left[(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) - (x-\mu_l)^T\Sigma_l^{-1}(x-\mu_l) + \log(|\Sigma_k|) - \log(|\Sigma_l|)\right]$$

$$= \log\frac{\pi_k}{\pi_l} - \frac{1}{2}[(x^T\Sigma_k^{-1}x - \mu_k^T\Sigma_k^{-1}x - x^T\Sigma_k^{-1}\mu_k + \mu_k^T\Sigma_k^{-1}\mu_k)$$
$$-(x^T\Sigma_l^{-1}x - \mu_l^T\Sigma_l^{-1}x - x^T\Sigma_l^{-1}\mu_l + \mu_l^T\Sigma_l^{-1}\mu_l) + \log(|\Sigma_k|) - \log(|\Sigma_l|)]$$

$$= \log\frac{\pi_k}{\pi_l} - \frac{1}{2}[x^T(\Sigma_k^{-1} - \Sigma_l^{-1})x - (\mu_k^T\Sigma_k^{-1}x + x^T\Sigma_k^{-1}\mu_k) + (\mu_l^T\Sigma_l^{-1}x + x^T\Sigma_l^{-1}\mu_l)$$
$$+ (\mu_k^T\Sigma_k^{-1}\mu_k - \mu_l^T\Sigma_l^{-1}\mu_l) + \log(|\Sigma_k|) - \log(|\Sigma_l|)]$$

$$= \log\frac{\pi_k}{\pi_l} - \frac{1}{2}[x^T(\Sigma_k^{-1} - \Sigma_l^{-1})x - 2(\mu_k^T\Sigma_k^{-1}x) + 2(\mu_l^T\Sigma_l^{-1}x) + (\mu_k^T\Sigma_k^{-1}\mu_k - \mu_l^T\Sigma_l^{-1}\mu_l)$$
$$+ \log(|\Sigma_k|) - \log(|\Sigma_l|)]$$

$$\left[\because x^T\Sigma^{-1}\mu = (\Sigma^{-1}\mu)^T x = \mu^T(\Sigma^{-1})^T x = \mu^T\Sigma^{-1}x \right.$$
$$\left. \text{Note}:\because \Sigma \text{ is symmetric, } \therefore \Sigma^T = \Sigma \text{ and } (\Sigma^{-1})^T = \Sigma^{-1} \right]$$

$$= -\frac{1}{2}[x^T(\Sigma_k^{-1} - \Sigma_l^{-1})x + 2(\mu_l^T\Sigma_l^{-1} - \mu_k^T\Sigma_k^{-1})x + \left(\mu_k^T\Sigma_k^{-1}\mu_k - \mu_l^T\Sigma_l^{-1}\mu_l + \log\frac{|\Sigma_k|}{|\Sigma_l|} - 2\log\frac{\pi_k}{\pi_l}\right)]$$

Therefore,

$$\textit{logit} = -\frac{1}{2}[x^T(\Sigma_k^{-1} - \Sigma_l^{-1})x + 2(\mu_l^T\Sigma_l^{-1} - \mu_k^T\Sigma_k^{-1})x + \left(\mu_k^T\Sigma_k^{-1}\mu_k - \mu_l^T\Sigma_l^{-1}\mu_l + \log\frac{|\Sigma_k|}{|\Sigma_l|} - 2\log\frac{\pi_k}{\pi_l}\right)]$$

(b) From, (a), using the condition, $\Sigma_k = \Sigma_l = \Sigma$, we get:

$$\textit{logit} = -\frac{1}{2}[2(\mu_l^T\Sigma^{-1} - \mu_k^T\Sigma^{-1})x + \left(\mu_k^T\Sigma^{-1}\mu_k - \mu_l^T\Sigma^{-1}\mu_l - 2\log\frac{\pi_k}{\pi_l}\right)]$$

$$= \log\frac{\pi_k}{\pi_l} - (\mu_l^T\Sigma^{-1} - \mu_k^T\Sigma^{-1})x - \frac{1}{2}\left(\mu_k^T\Sigma^{-1}\mu_k - \mu_l^T\Sigma^{-1}\mu_l\right)$$

$$= \log\frac{\pi_k}{\pi_l} - x^T\Sigma^{-1}\mu_l + x^T\Sigma^{-1}\mu_k - \frac{1}{2}\left(\mu_k^T\Sigma^{-1}\mu_k - \mu_l^T\Sigma^{-1}\mu_l\right)$$

$$[\because x^T\Sigma^{-1}\mu = (\Sigma^{-1}\mu)^T x = \mu^T(\Sigma^{-1})^T x = \mu^T\Sigma^{-1}x]$$

$$= \log\frac{\pi_k}{\pi_l} - \frac{1}{2}\left(\mu_k^T\Sigma^{-1}\mu_k - \mu_l^T\Sigma^{-1}\mu_l\right) + x^T\Sigma^{-1}(\mu_k - \mu_l)$$

Therefore,

$$\textit{logit} = \log\frac{\pi_k}{\pi_l} - \frac{1}{2}\left(\mu_k^T\Sigma^{-1}\mu_k - \mu_l^T\Sigma^{-1}\mu_l\right) + x^T\Sigma^{-1}(\mu_k - \mu_l)$$

(c) (By analogy)

First, we need to show: $X^T AY = Y^T AX$, when A is a symmetric matrix (and square matrix).

$$
\begin{aligned}
X^T AY &= (X)^T (AY) \\
&= (AY)^T (X) && [\because P^T Q = Q^T P] \\
&= Y^T A^T X && [\because (PQ)^T = Q^T P^T] \\
&= Y^T AX && [\because A \text{ is symmetric, } A = A^T]
\end{aligned}
$$

So, we have shown:

$$X^T AY = Y^T AX \qquad (i)$$

Now we like to show: $(X+Y)^T A(X-Y) = X^T AX - Y^T AY$

$$
\begin{aligned}
(X+Y)^T A(X-Y) &= (X^T + Y^T)A(X-Y) && [\because (P+Q)^T = P^T + Q^T] \\
&= X^T AX + Y^T AX - X^T AY - Y^T AY \\
&= X^T AX + (Y^T AX - X^T AY) - Y^T AY \\
&= X^T AX - Y^T AY && [\because from(i),\ (Y^T AX - X^T AY) = 0]
\end{aligned}
$$

So, we have shown:

$$(X+Y)^T A(X-Y) = X^T AX - Y^T AY$$

Thus, by analogy, we can write:

$$\log\frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T \Sigma^{-1}(\mu_k - \mu_l) + x^T \Sigma^{-1}(\mu_k - \mu_l) = \log\frac{\pi_k}{\pi_l} - \frac{1}{2}\left(\mu_k^T \Sigma^{-1}\mu_k - \mu_l^T \Sigma^{-1}\mu_l\right) + x^T \Sigma^{-1}(\mu_k - \mu_l)$$

3. What are the main tasks that autoencoders are used for?

Ans: Here are some of the main tasks that autoencoders are used for:

- Feature extraction
- Unsupervised pretraining
- Dimensionality reduction
- Generative models
- Anomaly detection (an autoencoder is generally bad at reconstructing outliers)

4. Suppose you want to train a classifier, and you have plenty of unlabeled training data but only a few thousand labeled instances. How can autoencoders help? How would you proceed?

Ans: If you want to train a classifier and you have plenty of unlabeled training data but only a few thousand labeled instances, then you could first train a deep autoencoder on the full dataset (labeled + unlabeled), then reuse its lower half for the classifier (i.e., reuse the layers up to the codings layer, included) and train the classifier using the labeled data. If you have little labeled data, you probably want to freeze the reused layers when training the classifier.

5. If an autoencoder perfectly reconstructs the inputs, is it necessarily a good autoencoder? How can you evaluate the performance of an autoencoder?

Ans: The fact that an autoencoder perfectly reconstructs its inputs does not necessarily mean that it is a good autoencoder; perhaps it is simply an overcomplete autoencoder that learned to copy its inputs to the codings layer and then to the outputs. In fact, even if the codings layer contained a single neuron, it would be possible for a very deep autoencoder to learn to map each training instance to a different coding (e.g., the first instance could be mapped to 0.001, the second to 0.002, the third to 0.003, and so on), and it could learn "by heart" to reconstruct the right training instance for each coding. It would perfectly reconstruct its inputs without really learning any useful pattern in the data. In practice, such a mapping is unlikely to happen, but it illustrates the fact that perfect reconstructions are not a guarantee that the autoencoder learned anything useful. However, if it produces very bad reconstructions, then it is almost guaranteed to be a bad autoencoder. To evaluate an autoencoder's performance, one option is to measure the reconstruction loss (e.g., compute the MSE or the mean square of the outputs minus the inputs). Again, a high reconstruction loss is a good sign that the autoencoder is bad, but a low reconstruction loss is not a guarantee that it is good. You should also evaluate the autoencoder according to what it will be used for. For example, if you are using it for unsupervised pretraining of a classifier, then you should also evaluate the classifier's performance.

6. What are undercomplete and overcomplete autoencoders? What is the main risk of an excessively undercomplete autoencoder? What about the main risk of an overcomplete autoencoder?

Ans: An undercomplete autoencoder is one whose codings layer is smaller than the input and output layers. If it is larger, then it is an overcomplete autoencoder. The main risk of an excessively undercomplete autoencoder is that it may fail to reconstruct the inputs. An overcomplete autoencoder's main risk is that it may just copy the inputs to the outputs without learning any useful features.

7. How do you tie weights in a stacked autoencoder? What is the point of doing so?

Ans: To tie the weights of an encoder layer and its corresponding decoder layer, you simply make the decoder weights equal to the encoder weights' transpose. This reduces the number of parameters in the model by half, often making training converge faster with less training data and reducing the risk of overfitting the training set.

8. What is a generative model? Can you name a type of generative autoencoder?

Ans: A generative model is a model capable of randomly generating outputs that resemble the training instances. For example, once trained successfully on the MNIST dataset, a generative model can be used to generate realistic images of digits randomly. The output distribution is

typically similar to the training data. For example, since MNIST contains many images of each digit, the generative model would output roughly the same number of images of each digit. Some generative models can be parametrized—for example, to generate only some kinds of outputs. An example of a generative autoencoder is the variational autoencoder.

9. What is a GAN? Can you name a few tasks where GANs can shine?

Ans: A generative adversarial network is a neural network architecture composed of two parts, the generator and the discriminator, which have opposing objectives. The generator's goal is to generate instances similar to those in training set to fool the discriminator. The discriminator must distinguish the real instances from the generated ones. At each training iteration, the discriminator is trained like a normal binary classifier, and then the generator is trained to maximize the discriminator's error. GANs are used for advanced image processing tasks such as super-resolution, colorization, image editing (replacing objects with realistic background), turning a simple sketch into a photorealistic image, or predicting the next frames in a video. They are also used to augment a dataset (to train other models), generate other types of data (such as text, audio, and time series), and identify the weaknesses in other models and strengthen them.

10. What are the main difficulties when training GANs?

Ans: Training GANs is notoriously difficult because of the complex dynamics between the generator and the discriminator. The biggest difficulty is mode collapse, where the generator produces outputs with very little diversity. Moreover, training can be terribly unstable: it may start out fine and then suddenly start oscillating or diverging, without any apparent reason. GANs are also very sensitive to the choice of hyperparameters.

11. Given two discrete probability distributions P and Q, write the KL divergence equation between these distributions, noted $D_{KL}(P||Q)$.

Ans: The KL divergence equation is:

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

12. Mention six applications of Generative Adversarial Networks (GAN).

Ans: GANs are now widely used for -

- super resolution (increasing the resolution of an image),
- colorization,
- powerful image editing (e.g., replacing photobombers with realistic background),
- turning a simple sketch into a photorealistic image,
- predicting the next frames in a video,

- augmenting a dataset (to train other models),
- generating other types of data (such as text, audio, and time series),
- identifying the weaknesses in other models and strengthening them,

and more.

13. Draw a schematic diagram of a Variational AutoEncoder (VAE), including the transformation of the space of the instances.

Ans:

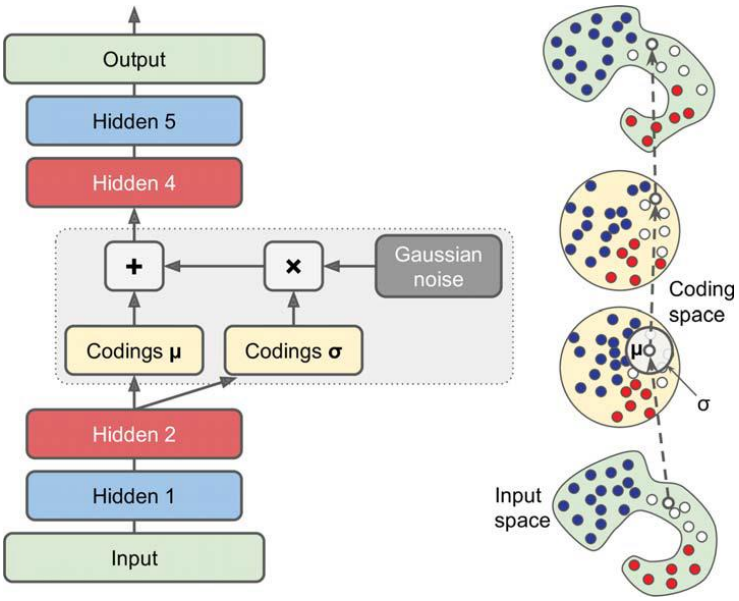

Figure: Variational autoencoder (left) and an instance going through it (right).

14. Describe two approaches to avoid the mode collapse while training a GAN?

Ans: To avoid the mode collapse while training a GAN, we can use the following approaches:

Experience Replay **(ER)** - ER is done by storing the images produced by the generator at each iteration in a replay buffer (gradually dropping older generated images) and training the discriminator using real images plus fake images drawn from this buffer (rather than just fake images produced by the current generator). This reduces the chances that the discriminator will overfit the latest generator's outputs.

Mini-Batch Discrimination **(MBD)** – MBD measures how similar images are across the batch and provides this statistic to the discriminator to easily reject a whole batch of fake images that

lack diversity. This encourages the generator to produce a greater variety of images, reducing the chance of mode collapse.

15. Describe the main guidelines for building stable convolutional GANs.

Ans: The main guidelines are:

- Replace any pooling layers with strided convolutions (in the discriminator) and transposed convolutions (in the generator),
- Use Batch Normalization in both the generator and the discriminator, except in the generator's output layer and the discriminator's input layer,
- Remove fully connected hidden layers for deeper architectures,
- Use ReLU activation in the generator for all layers except the output layer, which should use tanh,
- Use leaky ReLU activation in the discriminator for all layers.

These guidelines will work in many cases, but not always, so we may still need to experiment with different hyperparameters.

16. Draw the schematic diagram of StyleGAN's generator.

Ans:

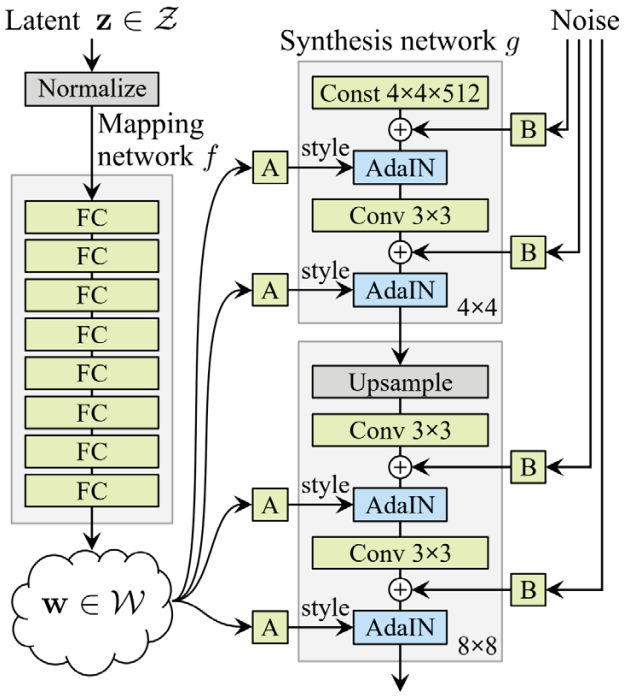

Figure: Architecture of StyleGAN's generator.

--- x ---