Problem 1 (2 points). What letter grade would you give yourself for this course? Give a one sentence justification.

① I would give myself an A- in this course, because of my performance on HW assignments, participation in lectures and our final project performance.

Definitely lot of room for improvement, as evidenced by my midterm exam score.

Problem 2 (10 points). For $x, \mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}_+$, evaluate

$$\nabla_\mu \mathop{\mathbb{E}}_{x \sim N(\mu,\, \sigma^2)} \left[x^2\right],$$

$$\nabla_\sigma \mathop{\mathbb{E}}_{x \sim N(\mu,\, \sigma^2)} \left[x^2\right] \text{ and}$$

$$\nabla_\mu \mathop{\mathbb{E}}_{x \sim N(\mu,\, \sigma^2)} \left[x^3\right],$$

where $N(\mu, \sigma^2)$ denotes the Normal distribution with mean μ and variance σ^2.

$\nabla_\mu \mathop{\mathbb{E}}_{x \sim N(\mu,\sigma^2)} [x^2]$ First, $\mathop{\mathbb{E}}_{x \sim N(\mu,\sigma^2)} [x^2] = \int x^2 \cdot \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

$$= \left(\frac{1}{\sqrt{2\pi}}\right)^3 \int_{-\infty}^{\infty} e^{\frac{-3(x-\mu)^2}{2\sigma^2}} \cdot \frac{1}{\sigma^3}\, d\mu$$

Can be simplified further by taking out the terms which do not contain μ

$$(1) = \left(\frac{1}{\sqrt{2\pi}\,\sigma}\right)^3 \int \left(e^{x^2} e^{\mu^2} e^{-2x\mu}\right)^{\frac{-3}{2\sigma^2}} d\mu$$

$\nabla_\mu \Rightarrow$ take a derivative of (1) w.r.t. μ, and apply the Fundamental Theorem of calculus

QUESTION 3

1. **(3 points)** For $x \in \mathbb{R}$, consider a probability distribution $p(x)$ which is a mixture of two Gaussians

$$p(x) = \alpha N(x;\ \mu_1, \sigma_1^2) + (1-\alpha) N(x;\ \mu_2, \sigma_2^2);$$

here $\alpha = 0.75$ is the mixture probability and μ_1, σ_1^2 and μ_2, σ_2^2 are the means and variances of the two Gaussians. If we solve a variational optimization problem

$$q^* = \underset{q \in \mathcal{Q}}{\text{argmin}}\ \text{KL}(q \,||\, p)$$

where $\mathcal{Q} = \mathcal{P}(\mathbb{R})$ is the set of all probability distributions with support on the entire real line, then what is q^*? Explain your answer. Hint: This question does not require any calculation. Think logically.

The linear combination of 2 Gaussians is also a Gaussian

$\Rightarrow p \sim N(x; \mu_3, \sigma_3^2)$ where

$$\mu_3 = 0.75\,\mu_1 + 0.25\,\mu_2$$
$$\sigma_3^2 = (0.75)^2 \sigma_1^2 + (0.25)^2 \sigma_2^2$$
$$= 0.56\,\sigma_1^2 + 0.06\,\sigma_2^2$$

The KL-divergence is minimized ($=0$) when

$q^* \sim N(x; \mu_3, \sigma_3^2)$ as well

(2)

2. **(3 points)** Answer True or False with a 1-2 sentence explanation. There exist first-order non-stochastic optimization algorithms that converge faster than Nesterov's accelerated gradient descent for all convex functions $\ell(w)$.

False. Any first-order deterministic optimization algorithm like Nesterov's accelerated GD needs at least $O\left(\sqrt{\frac{L}{\mu}} \log\left(\frac{1}{\epsilon}\right)\right)$ iterations to achieve an ϵ-ball approximation of the global optima.

Since this lower bound matches the upper bound of Nesterov, this method cannot be accelerated further.

3. **(3 points)** Answer True or False with a 1-2 sentence explanation. After a Generative Adversarial Network (GAN) has converged and is generating images that look like Nature's images, the discriminator $d_u(g_v(z))$ has an accuracy close to 100% for all noise vectors $z \sim N(0, I)$.

False. After a GAN converges, i.e. the generator gets better, the performance of the discriminator detericorates because it can no longer distinguish nature vs. the generator.

Thus, the discriminator will eventually converge to an Accuracy of 50%, after which the discriminator's feedback just keeps getting worse.

Problem 4. (10 points)

1. **(3 points)** Explain how inverted Dropout works in 1-2 sentences.

2. **(3 points)** We were training a neural network on the CIFAR-10 dataset with 10 output classes using the cross-entropy loss on the softmax predictions. We saw that the training loss after randomly initializing the weights of the network was 2.3. Can you explain why?
3. **(4 points)** Why is the convolutional operator used as a building block in CNNs? Why do typical CNNs have Average/Max-Pooling operators?

① Traditional dropout requires the model to scale all the activations by the keep probability at test-time. In contrast, Inverted dropout performs the scaling at train time. The implementation of this idea involves scaling the activations at train time by $(1-p)$, which has the added benefit of not requiring any changes at evaluation time, regardless of whether dropout is enabled.

② Cross entropy loss + Softmax prediction

⇒ After randomly initializing the weights + training, all outputs could still have the same values. This would explain the 2.3 loss,

$$\therefore \mathcal{L} \approx \frac{1}{10}\sum_{j=1}^{10} -\log_e\left(\frac{1}{10}\right) = 2.3$$

A workaround could be to tune the loss or the learning rate.

2. **(3 points)** We were training a neural network on the CIFAR-10 dataset with 10 output classes using the cross-entropy loss on the softmax predictions. We saw that the training loss after randomly initializing the weights of the network was 2.3. Can you explain why?
3. **(4 points)** Why is the convolutional operator used as a building block in CNNs? Why do typical CNNs have Average/Max-Pooling operators?

(4) convolutional operators & convolutional layers are the building blocks of a CNN, because they allow the CNN to use kernels that are smaller than input pictures, and apply the same set of weights to regions of the image, as opposed to a 1:1 mapping of pixel to weight.

Enabling weight sharing across any part of the image is crucial to building a tolerance for parameters like image rotations or translations, which improves generalizability

CNNs have average/max-pool operators because these help in down-sampling feature maps by summarizing the presence of features in patches of the image. Without these operators, the output of convolutional layers would be very sensitive to the position of a feature within an image; leading to bad test performance.

Problem 5 (10 points). Let $\mathcal{F}$ be a finite hypothesis class. We saw in PAC-learning that for any $\epsilon, \delta > 0$, if we draw a dataset D with n samples and if

$$n \geq \frac{1}{2\epsilon^2}\left(\log|\mathcal{F}| + \log\frac{2}{\delta}\right)$$

then with probability at least $1-\delta$ for all hypotheses $f \in \mathcal{F}$, we have

$$|\hat{R}(f) - R(f)| \leq \epsilon.$$

The quantities $\hat{R}(f)$ and $R(f)$ are the empirical risk and the population risk of a hypothesis f respectively. In particular, we have

$$|R(f_{\text{ERM}}) - R(f^*)| \leq 2\epsilon$$

with probability at least $1-2\delta$ if uniform convergence holds. This says that the population risk of the hypothesis f_{ERM} obtained using the training set is close to the population risk of f^* which is the hypothesis that achieves the smallest population risk in $\mathcal{F}$.

1. **(2 points)** John was trying to fit a model after learning this result and found some gap in the training and validation error with $n = 100$ samples. If he wants to reduce the gap by half, how many samples should he use?
2. **(4 points)** John collected the extra samples and observed that the gap between training and validation error did not halve. Give **two reasons** as to why this might be the case. Assume that John was diligent in collecting the new data and made sure that there is no mismatch in the distributions of the training and test data.
3. **(4 points)** Unlike John, you feel that it is cumbersome to gather more data and would instead like to use your knowledge of machine learning to create a good model. One strategy to use is as follows: you start fitting simple models on the dataset and sequentially increase the model complexity. In the first iteration, you use a hypothesis class $\mathcal{F}^1$ to get a model f^1_{ERM}, in the second iteration, you use a larger hypothesis class $\mathcal{F}^2$ to get a model f^2_{ERM}, and so on always ensuring that

$$\mathcal{F}^1 \subset \mathcal{F}^2 \subset \cdots$$

Suppose that all these hypothesis classes are finite and you can correctly compute $\log|\mathcal{F}^k|$ at each step. Using your knowledge of the the generalization bound given above, **without using a validation set or cross-validation** (say, the training dataset is too small to even do cross-validation) answer which model out of

$$\left\{f^1_{\text{ERM}}, f^2_{\text{ERM}}, \ldots, \right\}.$$

will you choose? Provide a 1-2 sentence explanation for your choice.

PROBLEM 5

① Given that $n \geq \frac{1}{2\epsilon^2} \log \frac{2|\mathcal{F}|}{\delta}$

to reduce the gap by half, we would need $4 \times 100 = 400$ samples of training data.

② Gap did not halve because

→ The finite hypothesis class $|\mathcal{F}|$ does not contain distributions similar to the population data, so the gap is the best you can do with that class

→ the n cited above was only a lower bound, so perhaps because the problem & dataset contain many nuances, requiring > 400+ more samples.

③ f^n because this hypothesis can be upper bounded at n.

($|\mathcal{F}|^n$ is the number of ~~hypothesis in the hypothesis class $\mathcal{F}$~~ samples.

Suppose that all these hypothesis classes are finite and you can correctly compute $\log|\mathcal{F}^k|$ at each step. Using your knowledge of the the generalization bound given above, **without using a validation set or cross-validation** (say, the training dataset is too small to even do cross-validation) answer which model out of

$$\left\{ f^1_{\text{ERM}}, f^2_{\text{ERM}}, \dots, \right\}.$$

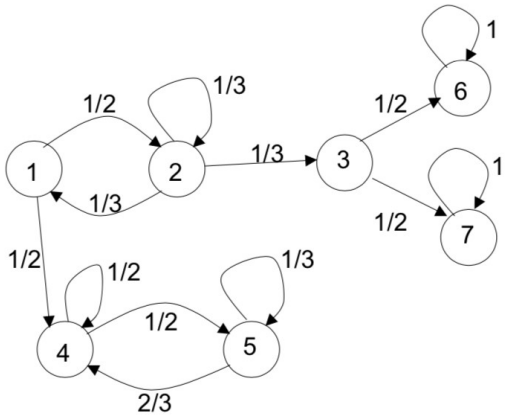
will you choose? Provide a 1-2 sentence explanation for your choice.

Problem 6. (10 points)

1. **(3 points)** For stochastic gradient descent with constant step-size η for an m-strongly convex and L-smooth function with global optimum w^*, the sub-optimality of SGD is

$$\mathop{\mathbb{E}}_{\omega_0,\dots,\omega_t} \left[\ell(w^{t+1}) - \ell(w^*)\right] \leq \frac{\eta L \sigma_0}{2m} + (1-\eta m)^t \left(\ell(w^0) - \ell(w^*) - \frac{\eta L \sigma_0}{2m}\right).$$

   The second term here converges very quickly with time while the first term is a constant with respect to time t. Does SGD converge with a constant step-size? How do we change the step-size so that SGD converges to the global minimum as $t \to \infty$ from any initial condition?
2. **(3 points)** We know that mini-batch SGD is not any cheaper computationally than SGD in theory; the reduction in the number of weight updates required to reach an ϵ-neighborhood of the global optimum (for convex objectives) is balanced out by the extra amount of computational work done during each weight update if the batch-size is larger than one. Why do we then use mini-batch size larger than 1 to train neural networks in practice?
3. **(4 points)** Consider the Markov chain with transition probabilities shown below.



   How many steady-state distributions are there? Argue your answer in 1-2 sentences. You do not need any computations to solve this problem, think logically.

(6) (1) Does SGD converge with a constant step size?

No, the step-size/learning rate needs to $\to 0$ per SGD to stop.

You can make step size a linear function of time, where step size gets reduced by a factor of $2\times$ every X weight updates (X can be any +ve value).

(2) We use min-batch larger than 1 because when computing SGD gradients with only a single sample, gradients are very volatile depending on which data gets sampled.

To smooth out some of this noise & amplify the signal, we use a batch of b data points, average out their gradients, and then apply the weight updates.

(3) Steady state distributions:

- 6, 7 } when the model reaches these states, there's no way out, so in steady state when eventually the model reaches one of these 4, it gets stuck permanently.
- 4 or 5, but not both ✗

Problem 7 (8 points). Estimate the integral

$$\int_{x\in\mathbb{R}} e^{-N(2x^2+x^4)}\,\mathrm{d}x$$

using the Laplace approximation if N is large. You will find it useful to know that

$$\int_{x\in\mathbb{R}} e^{-\frac{x^2}{2\sigma^2}}\,\mathrm{d}x = \sqrt{2\pi}\,\sigma.$$

Problem 8 (6 points). We assumed while studying SGD for machine learning problems that there exist constants $\sigma_0 \geq 0$ and $\sigma \geq 0$ that give an upper bound on the norm of the stochastic gradients

$$\mathop{\mathbb{E}}_{\omega}\left[\|\nabla \ell_\omega(w)\|^2\right] \leq \sigma_0 + \sigma\|\nabla \ell(w)\|^2$$

for any w in the domain.

1. **(4 points)** Show that $\sigma \geq 1$.
2. **(2 points)** Can σ_0 be zero? Explain your answer.

Problem 9 (14 points). Consider optimizing a one-dimensional loss function

$$\ell(w) = \frac{1}{2n}\sum_{i=1}^{n}(a^i w - b^i)^2$$

where $w, a^i, b^i \in \mathbb{R}$ and $\left\{(a^i, b^i)\right\}_{i=1,\ldots,n}$ is the dataset. We will use stochastic gradient descent (SGD) with a constant step-size $\eta > 0$ and constant batch-size of 1 in this problem. You can use the Markov chain model of SGD to answer the following questions; in this case, since the batch-size is 1, the updates of SGD are modeled as

$$W^{t+1} = W^t - \eta\,\nabla \ell(W^t) + \eta\,\xi^t$$

where $\xi^t \sim N(0, I)$ is standard Gaussian noise.

1. **(4 points)** Write down all critical points of $\ell(w)$, i.e., all locations where the gradient $\nabla \ell(w) = 0$.
2. **(5 points)** What is the asymptotic expected value

   $$\lim_{t\to\infty} \mathop{\mathbb{E}}_{\omega_1,\ldots,\omega_t}\left[w^{t+1}\right]$$

   of the iterates of SGD? Here ω_t denotes the sample of the datum $(a^{\omega_t}, b^{\omega_t})$ used to update the weights w^t at the t^{th} iteration.
3. **(5 points)** If we have $a^1 = a^2 = \ldots = a^n = 1$, what is the asymptotic variance of the iterates of SGD?

   $$\lim_{t\to\infty} \mathop{\mathrm{Var}}_{\omega_0,\ldots,\omega_t}\left[w^{t+1}\right]$$

⑦ Estimate: $\int_{x \in \mathbb{R}} e^{-N(2x^2+x^4)}$ using the Laplace transformation

$$\int e^{-nl(w)}\varphi(w)dw \approx e^{-nl(w^*)} \int \varphi(w)\, e^{-n/2}$$

Laplace transformation

(8) ① Show that $\sigma \geq 1$

② can σ_0 be 0?

⇒ Yes, this occurs naturally when our prototype perfectly design. sounds good.

Consider a proof by contradiction.

→ $\frac{1}{x}$ Cadberry code

Problem 10 (10 points). In your homework problem, you developed a variational generative model to synthesize MNIST digits. At test time, you sampled latent factors $z \sim N(0, I)$ and ran the decoder $p_v(x|z)$ to get new images different from the ones in the training set. However, you did not get any control as to which digit was generated by the decoder.

1. Explain in 4-5 sentences how you will modify the variational auto-encoder (VAE) to synthesize images of a particular digit. At test time, your model should take in as input a one-hot encoding of the digit and synthesize images only of that particular digit; it should be able to accept any of the 10 digits as input.
2. Clearly explain the inputs, outputs of all parts of your model, the loss function you will use to train the model and the trainable parameters.
3. Explain what the synthesized image will look like if instead of providing a one-hot vector as input, you provide a two-hot vector as input at test time, e.g.,

$$[0, 0, 1, 0, 1, 0, \ldots] = \text{one-hot}(2) + \text{one-hot}(4).$$

Hint: To answer this part, think of how the latent space $z \in \mathbb{R}^m$ is structured for a standard VAE.

① We know that the structure of the latent space of MNIST-10 contains distinct clusters for each of the digits, so identifying the specific regions responsible for each digit will allow us to see on which ~~part~~ digit we are generating.

Thus, based on the 1-hot encoded input, we can have all the digits.

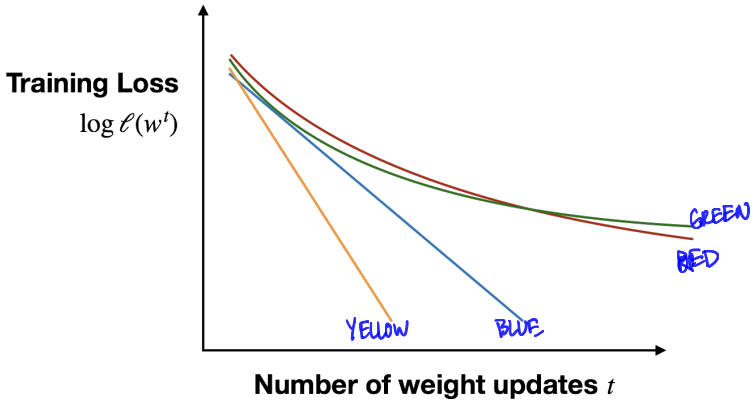
② Trainable params: model weights for 6 FC layers w/ ReLu between them

Inputs: Image of MNIST digits (downsampled + binarized)

Output: reconstructed image.

3 A 2-hot vector would not work because each digit occupies ~~exclusive~~ a distinct location in feature space; so trying to sample from both digits & create a case with is very rare.

(11)



END OF EXAM

(1) Gradient descent,
(2) Gradient descent with Nesterov's acceleration,
(3) Stochastic gradient descent with a small batch-size, and
(4) SGD with Nesterov's acceleration and a small batch-size.

RED represents SGD w Nesterov's acceleration

GREEN represents SGD w/ small batch size. This is the worst curve interms of training time, because the smaller batch size adds more variance/noise than Gradient Descent

BLUE represents gradient descent w/ Nesterov, which always makes steady progress towards the objective.

YELLOW represents GD w/ Nesterov's acceleration

We know that momentum does not help accelerate SGD significantly since the gradient is always incorrect due to the stochastic nature of the method.