Prof. T. Hofmann

Final Exam

February 3, 2020

First and Last name: ______________________________

Student ID (Legi) Nr: ______________________________

Signature: ______________________________

General Remarks

- Please check that you have all 28 pages of this exam.
- There are 111 points, and the exam is 120 minutes. **Don't spend too much time on a single question!** The maximum of points is not required for the best grade!
- Remove all material from your desk which is not permitted by the examination regulations.
- Write your answers directly on the exam sheets. If you need more space, make sure you put your **student-ID**-number on top of each supplementary sheet.
- Immediately inform an assistant in case you are not able to take the exam under regular conditions. Later complaints are not accepted.
- Attempts to cheat/defraud lead to immediate exclusion from the exam and can have judicial consequences.
- Please use a black or blue pen to answer the questions.
- Provide only one solution to each exercise. Cancel invalid solutions clearly.

	Topic	Max	Points	Visum
1	Feedforward Networks and Backpropagation	26		
2	Convolutional- and Recurrent Neural Networks	30		
3	Optimization and Learning Theory	28		
4	Graph Nets, Ensembles and Generative Models	27		
Total		111		

Grade: ....

1 Feedforward Networks and Backpropagation (26 pts)

1.1 Activation Functions (8 pts)

a) Calculate the derivative/subdifferential of the following activation functions:

1. $\mathrm{Relu}(x) = \max(0, x)$
2. $\mathrm{elu}(x) = \begin{cases} x & x > 0 \\ \alpha(\exp(x) - 1) & x \leq 0 \end{cases}, \quad 0 < \alpha \leq 1$
3. $\mathrm{softplus}(x) = \ln(1 + \exp(x))$

3 pts $\boxed{}$

....

....

....

....

....

....

b) Express the Relu activation function in terms of the absolute value function $|x| = x$ if $x \geq 0$, $|x| = -x$ otherwise and viceversa.
Hint: The functions are expressible in one line (i.e. no piece-wise definitions)!

2 pts $\boxed{}$

....

....

....

c) Now let us consider a simple one hidden layer feedforward network

$$F(\mathbf{x}) = \mathbf{W}_2 \left|\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1\right|$$

with $\mathbf{b}_1, \mathbf{x} \in \mathbb{R}^d$, $\mathbf{W}_i \in \mathbb{R}^{d \times d}$, $i \in \{1, 2\}$ and the absolute value function is applied element-wise. Construct an equivalent one hidden layer network using the Relu activation function instead of the absolute value function. State all the dimensions of the parameters.

3 pts $\square$

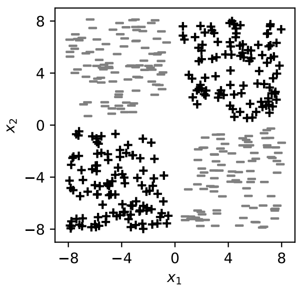
....

....

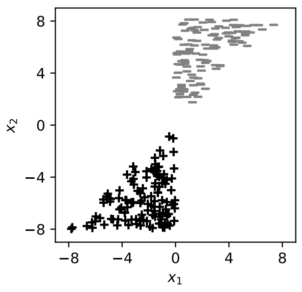
....

1.2 Linear Networks (3 pts)

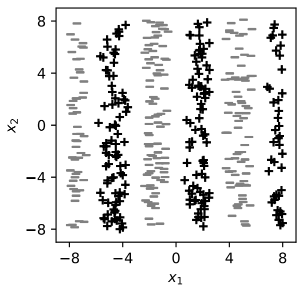
a) Consider the two-class, two-dimensional datasets depicted below, where (-) indicates class 0 and (+) indicates class 1. For each dataset, state whether a linear neural network can correctly learn to classify its data points. If not, suggest a possible input transformation that makes the dataset classifiable by a linear model.



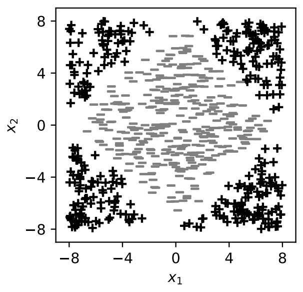
(a)



(b)



(c)



(d)

2 pts $\square$

....

....

....

....

b) Given a simple classifier that classifies a data point as (+) if the prediction $\hat{y} \geq 0$ and (-) if $\hat{y} < 0$, for the dataset (d) of the previous question, express $\hat{y}$ as a function of the features x_1 and x_2 as well as a scalar bias. You may of course use your above suggested transformation, if you suggested any.

1 pts $\square$

....

. . . .

1.3 Forward- vs. Reverse Mode Differentiation (5 pts)

In modern deep learning frameworks, computations are represented as a graph and therefore also the gradient computation becomes a graph. Consider the graph depicted in Figure 1.

a) Add a node for each partial derivative that is computed in a full **forward** mode differentiation path when computing $\frac{\partial l}{\partial h_1}$. (Add nodes and edges directly into Figure 1.)

$$\begin{tikzcd}
h_1 \arrow[d] \\
h_2 \arrow[d] \\
h_3 \arrow[d] \\
l
\end{tikzcd}$$

Figure 1: A simple network

2.5 pts $\boxed{}$

b) Briefly explain why backward- instead of forward mode differentiation is used in deep learning applications when computing the gradient of a neural network loss $l_{NN} : \mathbb{R}^d \to \mathbb{R}$, where d is the total number of trainable parameters.

2.5 pts $\boxed{}$

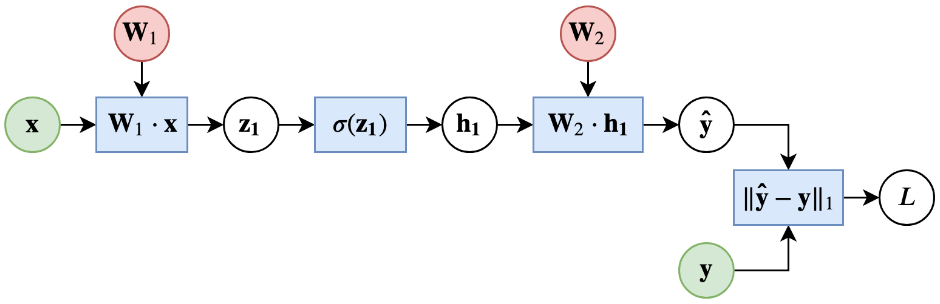
. . . .

. . . .

. . . .

1.4 Backpropagation on a neural network (5 pts)

Consider the neural network depicted below, which consists of two $d \times d$ learnable weight matrices $\mathbf{W_1}$ and $\mathbf{W_2}$. The network is trained to minimize the mean absolute error (MAE) between the predicted output $\hat{\mathbf{y}} \in \mathbb{R}^d$ and the ground-truth $\mathbf{y} \in \mathbb{R}^d$. $\sigma(\mathbf{z_1})$ denotes the element-wise sigmoid activation function, defined as $\mathbf{h_1} = \sigma(\mathbf{z_1}) = 1/(1+\exp(-\mathbf{z_1}))$. The goal is to compute the gradient of the loss L with respect to the weights, namely $\frac{\partial L}{\partial \mathbf{W_1}}$ and $\frac{\partial L}{\partial \mathbf{W_2}}$, using reverse-mode automatic differentiation.



a) Compute $\frac{\partial L}{\partial \hat{\mathbf{y}}}$ for the MAE loss ($\|\hat{\mathbf{y}} - \mathbf{y}\|_1$). In addition, explain why we do not use an activation function in the last layer, i.e. for $\hat{\mathbf{y}}$.

2 pts ☐

. . . .

. . . .

b) Now backpropagate through the sigmoid, i.e. compute $\frac{\partial L}{\partial \mathbf{z_1}}$ given $\frac{\partial L}{\partial \mathbf{h_1}}$ and $\mathbf{h_1}$.

1 pts ☐

. . . .

. . . .

c) Compute $\frac{\partial L}{\partial \mathbf{W_2}}$ as a function of $\frac{\partial L}{\partial \hat{\mathbf{y}}}$, and $\frac{\partial L}{\partial \mathbf{W_1}}$ as a function of $\frac{\partial L}{\partial \mathbf{z_1}}$

1 pts ☐

. . . .

. . . .

d) Do you need to compute $\frac{\partial L}{\partial \mathbf{x}}$ and $\frac{\partial L}{\partial \mathbf{y}}$ to train the model? Motivate your answer.

1 pts ☐

. . . .

. . . .

1.5 Networks with skip connections (5 pts)

In this exercise we consider the following network

$$\begin{aligned}
\mathbf{y}_1 &= f(\mathbf{x}, \mathbf{W}_1) + \mathbf{x} \\
\mathbf{y}_2 &= f(\mathbf{y}_1, \mathbf{W}_2) + \mathbf{y}_1 \\
\mathbf{y}_3 &= f(\mathbf{y}_2, \mathbf{W}_3) + \mathbf{y}_2 \\
&\vdots \\
\mathbf{y}_n &= f(\mathbf{y}_{n-1}, \mathbf{W}_n) + \mathbf{y}_{n-1} \\
l &= L(\mathbf{y}_n),
\end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the input of the network, $\mathbf{y_i} \in \mathbb{R}^d$ are the intermediate representations, $\mathbf{W}_i \in \mathbb{R}^{d \times d}$ are the trainable parameters, $f : \mathbb{R}^d \to \mathbb{R}^d$ is a smooth activation function and $L : \mathbb{R}^d \to \mathbb{R}^+$ is the (smooth) loss function.

a) For $n = 1$ calculate the derivative of l with respect to the input $\mathbf{y}_0 = \mathbf{x}$. What is the dimension of this derivative?

2 pts $\boxed{}$

. . . .

. . . .

. . . .

b) For $n > 1$ calculate the derivative of l with respect to the weights $\mathbf{W}_k$ for a given $1 \leq k < n$.

3 pts $\boxed{}$

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

2 Convolutional- and Recurrent Neural Nets (30 pts)

2.1 Convolution Theory (11 pts)

Consider a one-dimensional signal $f : \mathbb{Z} \to \mathbb{R}$ and a filter $h : \mathbb{Z} \to \mathbb{R}$ with discrete support:

$$h[i] = \begin{cases} 1/5 & \text{for } i \in \{-2, -1, 0, 1, 2\}, \\ 0 & \text{otherwise.} \end{cases}.$$

a) Write down the definition of "h convolved with f" (which was denoted as $(h * f)$ in the lecture) in full generality, and then simplify it by plugging in our choice for the filter h.

1 pts $\boxed{}$

. . . .

. . . .

. . . .

b) Find a signal $f : \mathbb{Z} \to \mathbb{R}$ such that $(h * f) = f$.

1 pts $\boxed{}$

. . . .

. . . .

c) Let $f[i] \sim \mathcal{N}(0, 1)$ (normal distribution with mean 0 and unit variance) for all $i \in \mathbb{Z}$. Also assume that $f[i]$ is uncorrelated from $f[j]$, for $i \neq j$. Compute the mean and the variance of $(h * f)[i]$ at any $i \in \mathbb{Z}$.

2 pts $\boxed{}$

. . . .

. . . .

. . . .

. . . .

d) In the setting described in the last point, let $y = (h * f)$ be the output of the convolution operation. Is $y[i]$ uncorrelated from $y[j]$ for $i \neq j$? Answer by computing the following expectation with respect to the randomness in f: $\mathbf{E}\big[y[i]y[j]\big]$.

Hint: It might be beneficial to draw a picture instead of directly performing the computation.

3 pts $\boxed{}$

....

....

....

....

....

e) Let now f be a *deterministic* sinusoidal signal: $f[i] = \alpha \sin(\omega i)$ for all $i \in \mathbb{Z}$, with $\alpha, \omega > 0$. Let $y = (h * f)$, with h given above, just before point (a). Assume $\omega \ll 1$, so that f is periodic with period considerably bigger than 5 (the support of the filter h). Is y going to look periodic? If so, is the period of y going to be approximately equal, bigger, or smaller than the period of f?

2 pts $\boxed{}$

....

....

....

f) In the setting we just described in point (e), prove that

$$\sup_{i \in \mathbb{Z}} y[i] \leq \alpha.$$

2 pts $\boxed{}$

....

....

....

2.2 Convolutional Neural Networks (6 pts)

We focus on 1D text convolution. Consider the common architecture in which each word in a document is represented as a d-dimensional embedding vector, a single convolutional layer with m filters is applied. Given n-words input text $w_1, ..., w_n$, we embed each symbol as d dimensional vector, resulting in word vectors $\mathbf{w}_1, ..., \mathbf{w}_n \in \mathbb{R}^d$. The resulting $d \times n$ matrix is then fed into a convolutional layer where we apply m narrow convolution filters (i.e. no padding) with a l-word sliding window over word vectors with stride 1.

(a) How many trainable parameters are there in the convolutional layer including biases?

2 pts ☐

....

....

(b) What is the size of the output of the convolutional layer?

2 pts ☐

....

....

(c) In Inception module, what is the purpose to use a 1x1 convolution after a previous convolutional layer? In our context, when does it make sense to add one more 1x1 convolution after the convolutional layer?

2 pts ☐

....

....

....

2.3 Recurrent Neural Networks (13 pts)

(a) A language model is a function M mapping a sequence of input tokens $W := (w_t)_{1 \leq t \leq T}$ (where w_t comes from a fixed vocabulary) to a probability distribution over the next token $P(w_{T+1}) = M(W)$. Explain how a *trained* RNN implementing a language model can be used to generate a set of 10 different plausible natural language sentences, hereby sketch your proposed architecture/algorithm.

2 pts $\boxed{}$

. . . .

. . . .

. . . .

(b) Assume fixed-length input sequences of word vectors $X := (x_t)_{1 \leq t \leq T}$ where $x_t \in \mathbb{R}^d$. Consider an MLP with Tanh activation function and *two hidden layers* (no bias parameters) of size 20 and 10, respectively, solving a binary sentiment classification problem, i.e. with each sequence X we associate a label $y \in \{-1, 1\}$. State the number of free (learnable) parameters in this MLP; each parameter is assumed to be a real number. Propose a RNN architecture with *the same* number of free parameters as the MLP. Justify your result by writing down the set of learnable parameters and clearly labeling their dimensions.

2 pts $\boxed{}$

. . . .

. . . .

. . . .

. . . .

(c) LSTM-based RNNs are an example of neural network architectures which use weight sharing, i.e. some given weight matrix $W^{(t)}$ in a LSTM cell is constrained to be the same for all other time-steps $t' \neq t$. Show that $\partial L / \partial W = \sum_{t=1}^{T} \partial L / \partial W^{(t)}$, where L is the loss function. Hint: Augment the computational graph with a dummy parameter W which ties all weight matrices.

1 pts $\boxed{}$

. . . .

. . . .

(d) Consider a simple LSTM-based RNN with *scalar* inputs/outputs and states. The cell state is updated as $C_t = C_{t-1} * f_t + i_t * \tilde{C}_t$ where (f_t, i_t) are the forget/input gates, and $(C_{t-1}, \tilde{C}_t, C_t)$ are the previous, tentative and updated cell states. The forget gate is defined as $f_t := \sigma(W_f \cdot [x_t, h_{t-1}] + b_f)$. Propose a suitable initialization of b_f at the beginning of training to discourage the vanishing gradient problem. (Here $*$ denotes elementwise multiplication and $\cdot$ denotes matrix multiplication.)

2 pts $\boxed{}$

. . . .

. . . .

(e) Consider an anomaly detection problem for financial transactions. For simplicity, assume that there are T transactions, and each transaction x_t is of a specific type $\tau \in \mathcal{T}$ where $\mathcal{T}$ is the ground set of possible transaction types. Propose and draw a RNN-based architecture which allows you to solve this problem in an unsupervised way, i.e. you have a very large pool of transactions histories of length T, but *no labeling* of the transactions. You are required to raise an alarm if a transaction is anomalous.

4 pts $\boxed{}$

. . . .

. . . .

. . . .

. . . .

. . . .

(f) Now assume that you receive partial supervision on the transaction histories. Assume that for each transaction x_t you either (i) have no information about whether it is malicious, or (2) the information that it is malicious, or (3) that it is normal. Propose a small modification to your previous architecture to use this supervision signal. Assume that in the test-set such information is available immediately after you have made your prediction (i.e. you have access to the true label of step t at the start of the next time step $t + 1$).

2 pts $\boxed{}$

. . . .

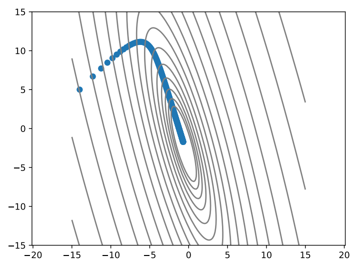
. . . .

. . . .

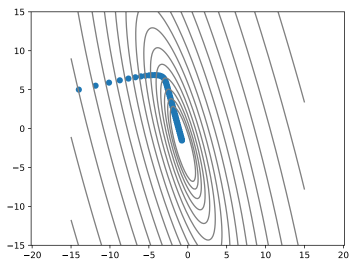
3 Optimization and Learning Theory (28 pts)

3.1 Optimization paths (3 pts)

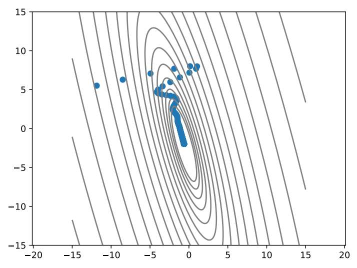
We run 3 different methods to optimize a two dimensional least squares problem: (i) gradient descent, (ii) gradient descent with momentum, and (iii) Adagrad. All methods use the same learning rate. The following figure traces the steps of these methods during optimization.



(a) ......



(b) ......



(c) ......

Figure 2: Convergence behavior of different optimization methods: we plot the level set of the objective as well as the steps taken by each optimization method.

a) Map each optimization method (i-iii) to its corresponding convergence plot (a-c) in the above figure. (1pt per correct answer, no negative points. Please indicate right below the figure.)

3 pts

3.2 Multiple choice on gradient methods (5 pts)

Are the following statements true or false? (+1 point for correct answers, −1 point for incorrect answers, no negative total points)

5 pts

(i) Early stopping can be seen as an approximate L2 regularization.
[ ] True [ ] False

(ii) Stochastic gradient descent only updates a fraction of the parameters in each iteration.
[ ] True [ ] False

(iii) The larger the Lipschitz constant of the function, the smaller one should choose the learning rate of SGD.
[ ] True [ ] False

(iv) The stochasticity in SGD can be seen as a "feature" because it can help to escape saddle points.
[ ] True [ ] False

(v) Adagrad decays the step size for weights that have more relative variance in their stochastic gradients.
[ ] True [ ] False

3.3 Gradient descent on the linear least square problem (9 pts)

Consider the problem of solving the linear system

$$\mathbf{X}\mathbf{w} = \mathbf{y} \tag{1}$$

where $\mathbf{X} \in \mathbb{R}^{n\times d}$ is the data matrix, $\mathbf{w} \in \mathbb{R}^d$ the parameter vector of the model, and $\mathbf{y} \in \mathbb{R}^n$ the target. We assume that there exists a unique solution provided by the mean-squared loss

$$\mathbf{w}_* = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \left(\frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 =: f(\mathbf{w}) \right). \tag{2}$$

We consider the gradient descent method

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla f(\mathbf{w}_t) \tag{3}$$

with a fixed step size α.

a) Calculate the gradient $\nabla f(\mathbf{w})$ and use it to re-write the gradient descent iterates of Eq. (3).

2 pts $\boxed{}$

....

....

b) Calculate the Hessian $\nabla^2 f(\mathbf{w})$ and argue whether or not $f(\mathbf{w})$ is convex.

2 pts $\boxed{}$

....

....

c) Show that the following bound on the distance to the optimizer holds

$$\|\mathbf{w}_{t+1} - \mathbf{w}_*\|_2 \leq \sigma_{\max}(\mathbf{I} - \alpha\mathbf{X}^\top\mathbf{X})^t \|\mathbf{w}_1 - \mathbf{w}_*\|_2 \tag{4}$$

- *Hint 1:* Remember that $\mathbf{y} = \mathbf{X}\mathbf{w}_*$

- *Hint 2:* $\|\mathbf{A}\|_2 = \sigma_{\mathsf{max}}(\mathbf{A}) = \max_{\mathbf{x}\in\mathbb{R}^d\setminus\{0\}} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$ for $\mathbf{A} \in \mathbb{R}^{n\times d}$

4 pts ☐

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

d) For what values of α does Gradient Descent converge, i.e. $\|\mathbf{w}_{t+1} - \mathbf{w}_*\|_2 \to 0$ as $t \to \infty$?

1 pts ☐

. . . .

3.4 Generalization (11 pts)

Least squares. We are given n i.i.d pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ ($\mathbf{x} \in \mathbb{R}^d$ and $y \in \mathbb{R}$) drawn from an unknown distribution $\mathcal{P}$. The classical least squares estimate of a linear model is computed by minimizing the so-called *empirical* risk:

$$\arg\min_{\mathbf{w}\in\mathbb{R}^d} \left(\mathcal{R}_n(\mathbf{w}) := \frac{1}{n}\sum_{i=1}^n \left(\mathbf{w}^\top \mathbf{x}_i - y_i\right)^2 \right) \tag{5}$$

which constitutes an empirical estimate of the *expected* risk minimizer given by

$$\arg\min_{\mathbf{w}\in\mathbb{R}^d} \left(\mathcal{R}(\mathbf{w}) = \mathbf{E}_{(\mathbf{x},y)\sim\mathcal{P}} \left[\left(\mathbf{w}^\top \mathbf{x} - y\right)^2 \right] \right). \tag{6}$$

We also introduce the following notations

$$c_n^2 := \sum_{i=1}^n y_n^2, \quad \mu_n := \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i y_i, \quad \mathbf{S}_n := \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top \tag{7}$$

According to the definition, $\mathbf{S}_n$ is a semi-positive definite matrix, namely $\mathbf{w}^\top \mathbf{S}_n \mathbf{w} \geq 0$
a) Express $\mathcal{R}_n(\mathbf{w})$ as a function of the quantities introduced in Eq. (7)

2 pts ☐

....

....

....

....

Model averaging. Model averaging relies on an average of $\mathcal{R}_n(\mathbf{w})$ over an ensemble of weights $\mathbf{w}$. Particularly, we consider averaging with respect to an arbitrary distribution over weights denoted by $\mathcal{Q}$ as

$$\mathcal{R}_n(\mathcal{Q}) := \mathbf{E}_{\mathbf{w}\sim\mathcal{Q}} \left[\mathcal{R}_n(\mathbf{w}) \right]. \tag{8}$$

b) Suppose that $\mathbf{E}_{\mathbf{w}\sim\mathcal{Q}}[\mathbf{w}] = \mathbf{v} \in \mathbb{R}^d$ and $\mathbf{E}\left[(\mathbf{w}-\mathbf{v})(\mathbf{w}-\mathbf{v})^\top\right] = \mathbf{S} \in \mathbb{R}^{d\times d}$. Then prove the following holds:

$$\mathcal{R}_n(\mathbf{S}, \mathbf{v}) := \mathcal{R}_n(\mathcal{Q}) = c_n^2 - 2\mathbf{v}^\top \mu_n + \mathbf{v}^\top \mathbf{S}_n \mathbf{v} + \mathsf{Tr}(\mathbf{S}_n \mathbf{S}). \tag{9}$$

3 pts ☐

. . . .

. . . .

. . . .

. . . .

. . . .

c) Prove that model averaging does not improve the *empirical* risk, namely that

$$\mathcal{R}_n(\mathbf{v}) \leq \mathcal{R}_n(\mathbf{v}, \mathbf{S}). \tag{10}$$

holds.

2 pts ☐

. . . .

. . . .

. . . .

. . . .

PAC-Bayesian bound. Consider the a multivariate normal prior on the weight $\mathbf{w}$ with mean zero and covariance matrix $\gamma\mathbf{I}$: $\mathbf{w} \sim N(0, \mathbf{I})$. Then, the PAC-Bayesian bound on the expect risk formulates as

$$\mathbf{E}_{\mathbf{w}\sim\mathcal{Q}}\left[\mathcal{R}(\mathbf{w})\right] \leq \mathcal{R}_n(\mathcal{Q}) + \sqrt{\mathsf{KL}\left(N(0,\gamma\mathbf{I})||\mathcal{Q}\right) + \ln(1/\delta) + 2.5\ln(n) + 8}/\sqrt{2n-1}, \tag{11}$$

d) Consider two different model averaging distributions: normal distribution as $\mathbf{Q}_N = N(\mathbf{v}, \mathbf{S})$ and uniform distribution $\mathbf{Q}_U$ with same mean and covariance matrices. Explain why $\mathcal{R}_n(\mathbf{Q}_N) = \mathcal{R}_n(\mathbf{Q}_U)$ holds. Then, indicate where and why a difference between these two averaging schemes is to be found.

2 pts ☐

. . . .

. . . .

. . . .

. . . .

e) In part c), we observed that $\mathcal{R}_n(\mathbf{v}) \leq \mathcal{R}_n(\mathcal{Q})$ for all $\mathcal{Q}$. Now, compare the PAC-Bayesian generalization bounds on $\mathcal{R}(\mathbf{v})$ and $\mathcal{R}(\mathcal{Q})$ for $\mathcal{Q} = N(\mathbf{v}, \mathbf{I})$.
Hint: recall that $KL(\delta(\mathbf{v})||N(0, \gamma\mathbf{I})) = \infty$ *where* $\delta(\mathbf{v})$ *is the delta Dirac on* $\mathbf{v}$.

2 pts

....

....

....

....

4 Graph Nets & Generative Models (27 pts)

4.1 Graph Neural Networks (7 pts)

Consider a setting where data $\mathbf{X} \in \mathbb{R}^{d\times 4}$ is related as represented by the graph given in Figure 3. Traditionally, one passes data through a (for example) one hidden layer ReLU network as $f(\mathbf{X}) = \text{ReLU}(\mathbf{W}\mathbf{X})$. In Graph Neural Networks, however, the forward pass is of the form $f_{\mathbf{Q}}(\mathbf{X}) = \text{ReLU}(\mathbf{W}\mathbf{X}\mathbf{Q})$.

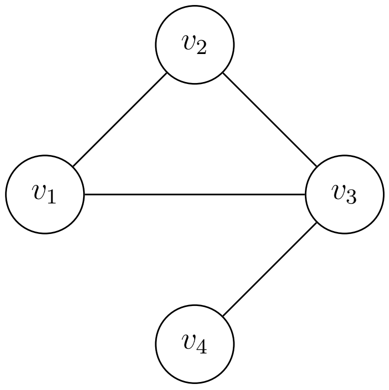


Figure 3: Graph $G = (V, E)$

a) Explain the high level difference between $f(\mathbf{X})$ and $f_{\mathbf{Q}}(\mathbf{X})$. That is, what is the goal of introducing the matrix $\mathbf{Q}$?

1 pts ☐

. . . .

. . . .

The matrix $\mathbf{Q}$ is computed as $\mathbf{Q} := \mathbf{D}^{-\frac{1}{2}}\tilde{\mathbf{A}}\mathbf{D}^{-\frac{1}{2}}$, where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_4$ and

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{12}$$

b) What does the matrix $\mathbf{A}$ respresent?

1 pts ☐

. . . .

c) Compute $\mathbf{D}$ and $\mathbf{D}^{-1/2}$.

1 pts $\boxed{}$

. . . .

. . . .

. . . .

. . . .

d) Compute $\mathbf{Q}$.

2 pts $\boxed{}$

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

e) What does the matrix $\mathbf{D}$ respresent and why do we pre- and postmultiply $\tilde{\mathbf{A}}$ by $\mathbf{D}^{-\frac{1}{2}}$?

2 pts $\boxed{}$

. . . .

. . . .

. . . .

4.2 Variational Autoencoders (6 pts)

Let $q_\phi(\mathbf{z}|\mathbf{x})$ denote the probabilistic encoder and let $p_\theta(\mathbf{x}|\mathbf{z})$ denote the probabilistic decoder. Let $\{\mathbf{x}_i\}_{i=1}^N$ be a data set of N i.i.d. samples of some continuous variable $\mathbf{x}$. The variational lower bound $\mathcal{L}$ of the marginal log-likelihood of datapoint i is given by

$$\log p_\theta(\mathbf{x}_i) \geq \mathcal{L}(\theta, \phi; \mathbf{x}_i) = \mathbf{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)}[\log p_\theta(\mathbf{x}_i|\mathbf{z})] - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}_i)||p(\mathbf{z})) \qquad (13)$$

where D_{KL} denotes the KL-divergence and where for simplicity we assume that the prior over latent variables $p(\mathbf{z})$ is fixed (i.e. we don't have to learn the prior's parameters), e.g. $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$.

In the following, we assume that the decoder is Gaussian, i.e. it is given by the output of a neural network with $\mathbf{z}$-dependent mean $\boldsymbol{\mu}_\theta(\mathbf{z})$ and fixed diagonal covariance $\sigma^2\mathbf{I}$, i.e.

$$p_\theta(\mathbf{x}_i|\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{z}), \sigma^2\mathbf{I}) \qquad (14)$$

where θ denotes the parameters of the neural network.

(i) Show that the reconstruction error term $\mathbf{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)}[\log p_\theta(\mathbf{x}_i|\mathbf{z})]$ for the above Gaussian decoder is equivalent to a sum of squares error between the original data and the network output $\boldsymbol{\mu}_\theta(\mathbf{z})$. *Hint: A multivariate Gaussian has density exp*$\big(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\intercal\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\big)/c$, *where* $c > 0$ *and* $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ *are the mean vector and covariance matrix respectively.*

3 pts $\boxed{}$

. . . .

. . . .

. . . .

. . . .

. . . .

(ii) Training the encoder requires computing the gradient of expectations such as

$$\nabla_\phi\, \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} f(\mathbf{z}) = \nabla_\phi \int q_\phi(\mathbf{z}|\mathbf{x}) f(\mathbf{z}) d\mathbf{z}, \qquad (15)$$

where f is an arbitrary smooth function. Since backpropagation cannot flow through a random node, a common technique to compute such expressions is to reparameterize $\mathbf{z} = g_\phi(\mathbf{x}, \boldsymbol{\epsilon})$ with an auxiliary noise variable $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ in order to decouple the encoder parameters ϕ from the randomness in $\mathbf{z}$.

Assuming there exists such a transformation $\mathbf{z} = g_\phi(\mathbf{x}, \boldsymbol{\epsilon})$, that leaves the differential mass invariant, i.e. $q_\phi(\mathbf{z}|\mathbf{x})d\mathbf{z} = p(\boldsymbol{\epsilon})d\boldsymbol{\epsilon}$, show that the parameter gradient in Eq. (15) can be computed as an unbiased estimator with samples $\{\boldsymbol{\epsilon}_\ell\}_{\ell=1}^L$ from the noise distribution $p(\boldsymbol{\epsilon})$.

3 pts $\boxed{}$

. . . .

. . . .

. . . .

. . . .

. . . .

4.3 f-Divergences and Maximum Likelihood Estimation (8 pts)

(i) An f-divergence is a function $D_f(P||Q)$ that measures the difference between two probability distributions P and Q that possess absolutely continuous densities $p(\mathbf{x})$ and $q(\mathbf{x})$:

$$D_f(P||Q) = \int_X q(\mathbf{x}) f\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right) d\mathbf{x}, \tag{16}$$

The function $f : \mathbb{R}_+ \to \mathbb{R}$ is convex and satisfies $f(1) = 0$.

Show that the f-divergence is always positive and that it is zero if and only if the measures P and Q coincide. *Hint: use Jensen's inequality.*

3 pts $\boxed{}$

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

(ii) The KL-divergence is a special case of f-divergence given by the function $f(t) = t \log t$. Show that minimizing the KL-divergence $D_{\mathrm{KL}}(P||Q_\omega)$ between a model density $q_\omega(\mathbf{x})$ and the data density $p(\mathbf{x})$ is equivalent to maximizing the expected log-likelihood of $q_\omega(\mathbf{x})$ under $p(\mathbf{x})$ with respect to the model parameters ω.

2 pts $\boxed{}$

....

....

....

....

....

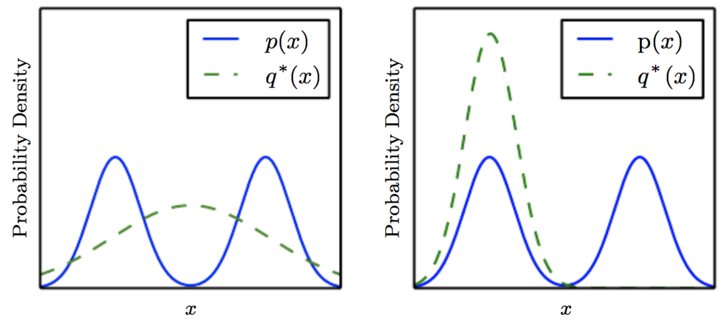


Figure 4: KL-divergences. Which **(i)** $q^* = \mathrm{argmin}_q D_{\mathrm{KL}}(p||q)$ vs. **(ii)** $q^* = \mathrm{argmin}_q D_{\mathrm{KL}}(q||p)$ corresponds to which **(left)** or **(right)** plot?

(iii) Suppose you have a distribution $p(\mathbf{x})$ and wish to approximate it with another distribution $q(\mathbf{x})$. Since the KL divergence is asymmetric, the choice of which KL divergence to use is task-dependent: some applications require an approximation that tries to place high probability everywhere that the true distribution places high probability, while other applications require an approximation that avoids placing high probability anywhere that the true distribution places low probability.

Looking at Figure 4, specify which direction of the KL-divergence was used to obtain $q^*(\mathbf{x})$, i.e. specify which of **(i)** $q^* = \mathrm{argmin}_q D_{\mathrm{KL}}(p||q)$ vs. **(ii)** $q^* = \mathrm{argmin}_q D_{\mathrm{KL}}(q||p)$ corresponds to which plot, **(left)** or **(right)** in Figure 4 (2 pt), and explain your answer (2 pts).

Hint: Recall the definition of the KL-divergence in part (b).

3 pts $\boxed{}$

(i) $q^* = \mathrm{argmin}_q D_{\mathrm{KL}}(p||q)$ corresponds to .... plot

(ii) $q^* = \mathrm{argmin}_q D_{\mathrm{KL}}(q||p)$ corresponds to .... plot

. . . .

. . . .

. . . .

. . . .

4.4 GANs (6 pts)

In the original GAN formulation, the parameters Θ of the generator G_Θ are optimized by the following update step: $\Theta \leftarrow \Theta - \triangle\Theta$, where

$$\triangle\Theta = \nabla_\Theta \mathbf{E}_{z\sim p(\mathbf{z})} \log\left(1 - D(G_\Theta(\mathbf{z}))\right), \tag{17}$$

where D denotes the discriminator, and z is the latent variable. Let $D^* = \frac{P_r}{P_{g_{\Theta_0}} + P_r}$ be the optimal discriminator for a fixed value Θ_0 . We have seen in the lectures that this update rule corresponds to minimizing the Jensen-Shannon divergence between the real data distribution, P_r, and the generated distribution, P_{g_Θ}, i.e.:

$$\mathbf{E}_{\mathbf{z}\sim p(\mathbf{z})}[\nabla_\Theta \log\left(1 - D^*(G_\Theta(\mathbf{z}))\right)|_{\Theta=\Theta_0}] = \nabla_\Theta 2D_{\text{JSD}}(P_{G_\Theta}||P_r)|_{\Theta=\Theta_0}. \tag{18}$$

This update rule suffers from vanishing gradients i.e. early in the training when G is very poor and D is able to accurately distinguish generated from real samples, the gradients are close to zero. For this reason, an alternative update rule has been proposed:

$$\triangle\Theta = -\nabla_\Theta \mathbf{E}_{z\sim p(\mathbf{z})} \log\left(D(G_\Theta(\mathbf{z}))\right) \tag{19}$$

Your task is to derive which cost function is being optimized by this gradient step.

(i) Show that $D_{\text{KL}}(P_{G_\Theta}||P_r)$ can be expressed only in terms of D^*, P_{g_Θ} and $P_{g_{\Theta_0}}$. That is, show that:

$$D_{\text{KL}}(P_{G_\Theta}||P_r) = D_{\text{KL}}(P_{G_\Theta}||P_{G_{\Theta_0}}) - \mathbf{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log \frac{D^*(G_\Theta(\mathbf{z}))}{1 - D^*(G_\Theta(\mathbf{z}))}].$$

3 pts ☐

. . . .

. . . .

. . . .

. . . .

(ii) Prove that the alternative update rule corresponds to minimizing the inverted KL minus two JSD:

$$\mathbf{E}_{\mathbf{z}\sim p(\mathbf{z})}[\nabla_\Theta \log\left(D^*(G_\Theta(\mathbf{z}))\right)|_{\Theta=\Theta_0}] = \nabla_\Theta[D_{\mathrm{KL}}(P_{G_\Theta}||P_r) - 2D_{\mathrm{JSD}}(P_{G_\Theta}||P_r)]|_{\Theta=\Theta_0}.$$

Hint: Start by calculating $\nabla_\Theta D_{\mathrm{KL}}(P_{G_\Theta}||P_r)|_{\Theta=\Theta_0}$ *and use equation (18).*

3 pts $\boxed{}$

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

Supplementary Sheet

Supplementary Sheet

Supplementary Sheet

Supplementary Sheet