PURDUE

BME 64600 – 001 and ECE 60146 – 001

Midterm #2, Spring 2024

NAME ______________________________________

PUID ____

Exam instructions:

- You have 75 minutes to work the exam.
- This is a closed-book and closed-note exam. You may not use or have access to your book, notes, any supplementary reference, a calculator, or any communication device including a cell-phone or computer.
- You may not communicate with any person other than the official proctor during the exam.

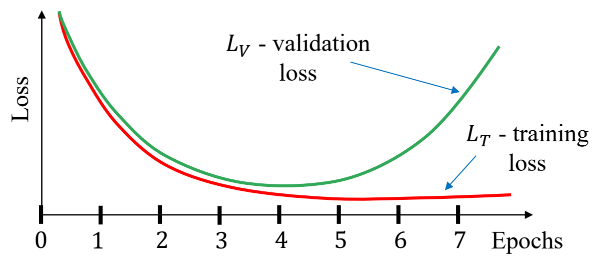
To ensure Gradescope can read your exam:

- Write your full name and PUID above and on the top of every page.
- Answer all questions in the area designated for each problem.
- Write only on the front of the exam pages.
- DO NOT run over to the next question.

Name/PUID: Key

Problem 1. (30pt) Training, generalization, and regularization

You are training a deep neural network, and you get the result pictured below.



a) What number of epochs is likely to the result in the best parameter estimates? Why?

Solution: 4 epochs because this is the value at which the validation loss is approximately minimized.

---

b) What is likely to happen if you choose the parameter estimates from 7 epochs? Why?

Solution: In this case, the parameter estimates will likely be overfit to the training data, so they will not generalize well.

---

c) What does this curve suggest about the capacity of your model?

Solution: The curve suggests that the model's capacity may be a bit too high given the amount of available training data.

---

d) Given your answer to c) above and assuming that you can not change the model, then what other modification to the training might be helpful? Why?

Solution: It might be helpful to introduce regularization of the parameters into the loss function. For example, l_1 regularization can enforce sparsity in the model, there by removing the unnecessary parameters, and effectively reducing the model order.

---

e) If you increase the training data by a factor of 10, then what would you expect to happen to the **training** loss curve?

Solution: Since it is more difficult to fit a larger amount of data, I would expect the training loss to go up.

---

f) If you increase the training data by a factor of 10, then what would you expect to happen to the **validation** loss curve?

Solution: Since the larger amount of training data will reduce overfitting, I would expect the validation loss to go down.

Name/PUID: ______________________________

Problem 2. (25pt) An image is a point in a space and can have a distribution.

Let $X \in [0,1]^{N^2}$ be an $N \times N$ image that takes values in the range $[0,1]$.
Furthermore, we will assume it is a random vector with distribution $X \sim p(x)$.
Also assume that $Y \in \Re^{N^2}$ is an image and that $Y = AX$ for some matrix A.
a) What is the shape of A?

Solution: A is a $N^2 \times N^2$ matrix.

---

b) If X is a 1024×1024 image (i.e., 1 Mega pixel), and A is stored with single precision floats, then how much memory does it take to directly store the matrix A on a standard computer?

Solution: Since A is a then a $(2^{10})^2 \times (2^{10})^2$ matrix, and since a single precision float is 4 bytes on a standard computer, we know that A will take 2^{42} bytes or equivalently 4 Terra Bytes of storage.

---

c) Let X_i be a single pixel in X. Then write an expression for the marginal distribution of $X_i \sim p_i(x_i)$ in terms of $p(x)$.

Solution:

$$p_i(x_i) = \int_{\Re} \cdots \int_{\Re} p(x) dx_0 \cdots dx_{i-1}\, dx_{i+1} \cdots dx_{N^2-1} \ .$$

---

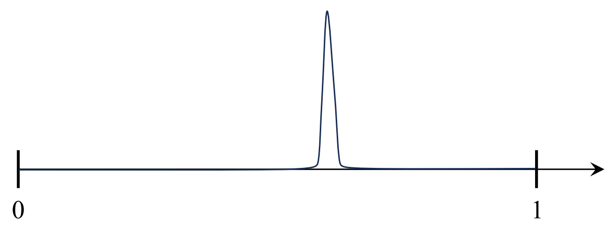
d) Write an expression for $p_i(x_i|x_0, \cdots, x_{i-1}, x_{i+1}, \cdots, x_{N^2-1})$, the conditional probability distribution of X_i given $(X_0 \cdots, X_{i-1}, X_{i+1}, \cdots, X_{N^2-1})$.

Solution:

$$p_i(x_i|x_0, \cdots, x_{i-1}, x_{i+1}, \cdots, x_{N^2-1}) = \frac{p(x_0, \cdots, x_{N^2-1})}{\int_{\Re} p(x_0, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, , x_{N^2-1}) dx_i}$$

---

e) Notice that both $p_i(x_i)$ and $p_i(x_i|x_0, \cdots, x_{i-1}, x_{i+1}, \cdots, x_{N^2-1})$ are density functions for X_i. If someone gives you the following plot of a density function, would you guess that it was the marginal distribution or the conditional distribution of X_i? Why?



Solution: I would guess that it is the conditional distribution because it is relatively narrow. If it was the marginal distribution, then I would expect it to be broad since a pixel might take any value between 0 and 1. However, the conditional distribution of a pixel given its neighbors usually has very little uncertainty since the pixel should be similar to its neighbors.

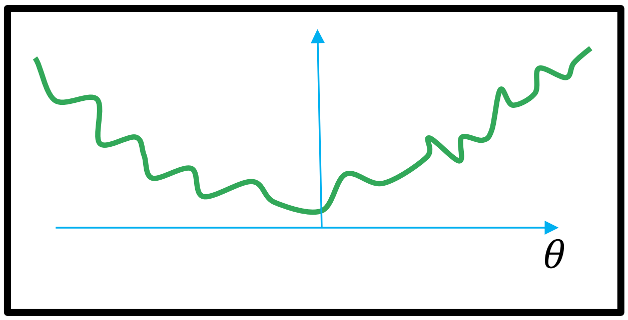
Name/PUID: ______________________________

Problem 3. (25pt) Stochastic Gradient Descent

Consider the loss function given by,

$$L(\theta) = \sum_{k=0}^{K-1} \|x_k - f_\theta(y_k)\|^2$$

that we would like to minimize as a function of θ using stochastic gradient descent.



Furthermore, let the full set of training data be partitioned into subsets, S_b, for $b = 0, \cdots, K/N_b$, where N_b is the batch size, and define the partial loss function,

$$L(\theta; S_b) = \sum_{k \in S_b} \|x_k - f_\theta(y_k)\|^2 \ .$$

Also, assume you use an algorithm such as the one below.

$v \leftarrow 0$

Repeat until converged:

For $b = 1$ to N_b:

$d \leftarrow -\nabla_\theta L(\theta; S_b)$

$v \leftarrow \gamma v + \alpha(1-\gamma)d$

$\theta \leftarrow \theta + v^t$

a) Name two advantages of using a smaller batch size for this problem?

Solution: Two advantages of using a smaller batch size are:

1. Each iteration will be much faster (but the amount of time for an epoch will remain approximately the same).
2. The SGD algorithm will be more robust to local minimum since the "noise" in the gradient update will help to "knock it out" of local minimum.

b) Name a disadvantage of using a smaller batch size for this problem?

Solution: A disadvantage of using a smaller batch size is that SGD will tend to "hunt" around the exact solution. So the final estimate of θ will be a bit noisy.

c) What happens when $\gamma = 0$?

Solution: In this case, there is no momentum, and the step is exactly proportional to the negative gradient.

d) What is it called when we set $\gamma > 0$, and what is the advantage of doing this?

Solution: This is called "momentum". The advantage of using momentum is that SGD tends to do a better job of jumping out of local minimum.

e) If at every iteration, $d = \mathbf{1}$, then what is $\lim_{iterations \to \infty} v$?

Solution: We can find the solution to this by solving for the equilibrium given by

$$\begin{aligned} v &= \gamma v + \alpha(1-\gamma)d \\ v &= \gamma v + \alpha(1-\gamma)\mathbf{1} \\ (1-\gamma)v &= \alpha(1-\gamma)\mathbf{1} \\ v &= \alpha\mathbf{1} \end{aligned}$$

So asymptotically, the step size is α times the negative gradient.

Name/PUID: ______________________________

Problem 4. (25pt) Residual training of denoisers

Our goal is to train a denoiser to remove additive Gaussian white noise from images of size $N \times N$. To do this, we generate training data with the form

$$Y_k = X_k + W_k \ ,$$

where $W_k \sim N(0, \sigma^2 I)$ are i.i.d. Gaussian random vectors of dimension N^2 for $k = 0, \cdots, K-1$. Furthermore, let $W_{k,i}$ denote the i^{th} component of W_k.

a) Calculate a simple expression for $R(i,j) = E[W_{k,i} W_{k,j}]$. What does this tell you about the components of W_k?

Solution:

$$R(i,j) = E[W_{k,i} W_{k,j}] = \sigma^2 \delta(i-j)$$

---

b) How should we select the training images X_k? Why?

Solution: We should select K images that our typical of the images we plan on denoising. This will allow the neural network to learn the distribution of both the images (i.e., the prior distribution) and the noise.

---

c) If we are going to train a deep neural network, $f_\theta(y)$, to perform this task, then what is the best way to train it?

Solution: It is best to train the neural network, $f_\theta(y)$, to estimate the noise rather than to estimate the image.

---

d) Write out the loss function for training the deep neural network in terms of X_k, Y_k, W_k and f_θ.

Solution:

$$L(\theta) = \frac{1}{K} \|W_k - f_\theta(Y_k)\|^2$$

---

e) What is the advantage of estimating the noise over estimating the signal?

Solution: The advantage of estimating the noise is that it has less dynamic range, so it is easier to train the neural network to accurately estimate it. Moreover, estimating the noise can be viewed as a skipped connection, so this has the advantage of mitigating the vanishing gradient problem.

Name/PUID: ______________________________

Problem 5. (20pt) Convolution blocks and their adjoint

Consider the function $f : \Re^N \rightarrow \Re^N$ that implements 1D (true) convolution with the kernel $[w_0, \cdots, w_{2p}]$ for some p and uses a "same" boundary condition.
Also, let $g = f^t$ be the adjoint function of f.
a) Is f a linear function? Provide a proof.
Hint: A function f is linear if $\forall x, y \in \Re^N$ and $\forall \alpha, \beta \in \Re$ it is always the case that $f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$.

Solution: Yes, f is linear because

$$
\begin{aligned}
f(\alpha x + \beta y) &= w * (\alpha x + \beta y) && (1)\\
&= \alpha w * x + \beta w * y && (2)\\
&= \alpha f(x) + \beta f(y) \ . && (3)
\end{aligned}
$$

---

b) For the case that $N = 5$ and $P = 1$, write out a matrix A so that $f(x) = Ax$.

Solution:

$$
A = \begin{bmatrix}
w_1 & w_0 & 0 & 0 & 0 \\
w_2 & w_1 & w_0 & 0 & 0 \\
0 & w_2 & w_1 & w_0 & 0 \\
0 & 0 & w_2 & w_1 & w_0 \\
0 & 0 & 0 & w_2 & w_1
\end{bmatrix}
$$

---

c) For the case that $N = 5$ and $P = 1$, write out a matrix B so that $g(y) = By$.

Solution:

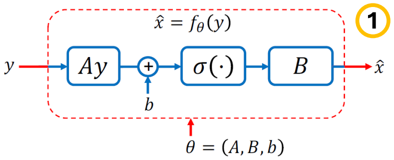
$$
B = \begin{bmatrix}
w_1 & w_2 & 0 & 0 & 0 \\
w_0 & w_1 & w_2 & 0 & 0 \\
0 & w_0 & w_1 & w_2 & 0 \\
0 & 0 & w_0 & w_1 & w_2 \\
0 & 0 & 0 & w_0 & w_1
\end{bmatrix}
$$

---

d) What is the interpretation of g?

Solution: g is convolution with the time-reversed kernel, $[w_{2p-1}, \cdots, w_0]$.

Single Layer NN Graphically: (1)



Where: $f_\theta(y) = B\sigma(Ay + b)$

M-dimensional simplex:

$$S^M = \left\{x \in \Re^M : \forall i, x_i \geq 0 \text{ and } 1 = \sum_{i=0}^{M-1} x_i\right\}$$

Softmax: $\sigma_i(z) = \dfrac{e^{z_i}}{\sum_j e^{z_j}}$

Gradient of the Softmax:

$$[\nabla\sigma(z)]_{i,j} = \frac{1}{\sum_k e^{z_k}}\left(e^{z_i}\delta_{i-j} - \frac{e^{z_i}e^{z_j}}{\sum_k e^{z_k}}\right)$$

MSE Loss: $L_{MSE}(\theta) = \dfrac{1}{K}\sum_{k=0}^{K-1}\|x_k - f_\theta(y_k)\|^2$

Gradient Descent and line search: (2)

Repeat until converged {
$d \leftarrow -\nabla L(\theta)$
$\alpha^* \leftarrow \arg\min_\alpha\{L(\theta + \alpha d^t)\}$
$\theta \leftarrow \theta + \alpha^* d^t$
}

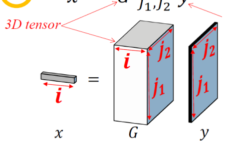
Steepest Descent Algorithm

Loss gradient (MSE):

$$-\nabla_\theta L_{MSE}(\theta) = \frac{2}{K}\sum_{k=0}^{K-1}\left(x_k - f_\theta(y_k)\right)^t \nabla_\theta f_\theta(y_k)$$

Einstein Notation: (3)

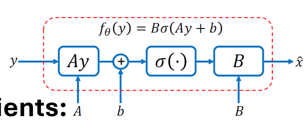
$x^i = G^i{}_{j_1,j_2}\, y^{j_1,j_2}$



Delta Functions:

$$\delta^i{}_j = \begin{cases}1 & \text{if } i = j\\ 0 & \text{if } i \neq j\end{cases}$$

$G^k{}_j = G^k{}_i\,\delta^i{}_j$



Gradients:

w.r.t. y $\left[\nabla_y f\right]^i{}_j = B^i{}_{i_1}[\nabla\sigma]^{i_1}{}_{i_2}A^{i_2}{}_j$

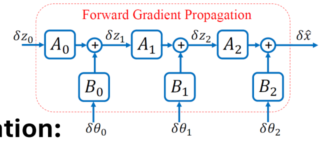
w.r.t. A $\left[\nabla_A f\right]^{i_1}{}_{j_1,j_2} = B^{i_1}{}_{i_2}[\nabla\sigma]^{i_2}{}_{j_1}\,y^{j_2}$

w.r.t. b $\left[\nabla_b f\right]^{i_1}{}_{j_1} = B^{i_1}{}_{i_2}[\nabla\sigma]^{i_2}{}_{j_1}$

w.r.t. B $\left[\nabla_B f\right]^{i_1}{}_{j_1,j_2} = \delta^{i_1}{}_{j_1}\sigma^{j_2}$

Adjoint of the Loss Gradient: (4)

$$[\nabla L_{MSE}(\theta)]^t = \frac{-2}{K}\sum_{k=0}^{K-1}[\nabla f_\theta(y_k)]^t\left(x_k - f_\theta(y_k)\right)$$



Back Propagation:

$g_{m,k} = \nabla_{\theta_m}\|x_k - f_\theta(y_k)\|^2 = -2\left(x_k - f_\theta(y_k)\right)^t\nabla f_\theta(y_k)$

$\nabla_{\theta_m}L(\theta) = g_m = \dfrac{1}{K}\sum_{k=0}^{K-1} g_{m,k}$ → Gradient of Loss Function

MAE: $\|a - b\|_1^1 = \sum_i |a_i - b_i|$

MC Cross Entropy: $-\sum_i a_i \log b_i$

Back Prop for Loss Functions:

MSE: $\epsilon_k^t = -2\left(x_k - f_\theta(y_k)\right)$

MAE: $\epsilon_k^t = -sign\left(x_k - f_\theta(y_k)\right)$

CE: $\epsilon_k^t = -\dfrac{x_k}{f_\theta(y_k)}$

Convolution "Same" Boundary: (5)

$$x(j_1,j_2) = \sum_{k_1=-1}^{1}\sum_{k_2=-1}^{1} z(j_1 - k_1, j_2 - k_2)\,w(k_1,k_2)$$

or $x^{j_1,j_2} = w_{(j_1,j_2)} * z^{(j_1,j_2)}$

Convolution "Valid" Boundary:

$$x(j_1,j_2) = \sum_{k_1=-1}^{1}\sum_{k_2=-1}^{1} z(j_1 + 1 - k_1, j_2 + 1 - k_2)\,w(k_1,k_2)$$

or $x^{j_1,j_2} = w_{(j_1,j_2)} * z^{(j_1,j_2)}$

2D Convo w/ vector input: D_i=2, D_o=1

$$x(j_1,j_2) = \sum_{i=0}^{d-1}\sum_{k_1=-p}^{p}\sum_{k_2=-p}^{p} z(j_1 - k_1, j_2 - k_2, i)\,w(k_1,k_2,i)$$

or $x^{j_1,j_2} = w_{(j_1,j_2),i} * z^{(j_1,j_2),i}$

2D Convo w/ vector input/output: D_i=2, D_o=2

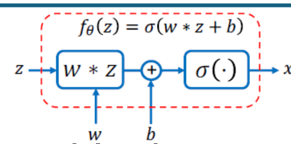
$x^{j_1,j_2,j_3} = w_{(j_1,j_2),i}{}^{j_3} * z^{(j_1,j_2),i}$

CNN Average Pooling: d=stride

$$x(j_1,j_2) = \sum_{k_1=0}^{p-1}\sum_{k_2=0}^{p-1}\frac{1}{p^2}z(d_1 j_1 + k_1, d_2 j_2 + k_2)$$

CNN Max Pooling:

$$x(j_1,j_2) = \max_{0\leq k_1<p}\max_{0\leq k_2<p} z(d_1 j_1 + k_1, d_2 j_2 + k_2)$$



w.r.t. Input:

Forward Function

$f(y) = \sigma\left(w_{(j_1,j_2),i}{}^{j_3} * z^{(j_1,j_2),i} + b^{j_3}\right)$

Adjoint Gradient

$\delta_{j_1,j_2,i} = w_{(-j_1,-j_2),i}{}^{j_3}[\nabla\sigma]^{k_1,k_2,k_3}{}_{(j_1,j_2),j_3}\epsilon_{k_1,k_2,k_3}$

w.r.t. b:

Forward Function

$f(y) = \sigma\left(w_{(j_1,j_2),i}{}^{j_3} * z^{(j_1,j_2),i} + b^{j_3}\right)$

Adjoint Gradient

$[g_b]_{j_3} = \mathbf{1}^{j_1,j_2}[\nabla\sigma]^{k_1,k_2,k_3}{}_{j_1,j_2,j_3}\epsilon_{k_1,k_2,k_3}$

w.r.t. w:

Forward Function

$f(y) = \sigma\left(w_{(j_1,j_2),i}{}^{j_3} * z^{(j_1,j_2),i} + b^{j_3}\right)$

Adjoint Gradient

$[g_w]_{j_1,j_2,i,j_3} = z^{(-j_1,-j_2),i}[\nabla\sigma]^{k_1,k_2,k_3}{}_{(j_1,j_2),j_3}\epsilon_{k_1,k_2,k_3}$

Update Directions:

Gradient step: $A \leftarrow A + \alpha d^t$ → *Replace A with b or B*

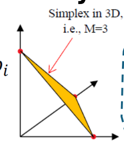
For A: $d_{j_1,j_2} = -\nabla_A L(\theta) = \dfrac{2}{K}\sum_{k=0}^{K-1}\epsilon_{k,i_1}B^{i_1}{}_{i_2}[\nabla\sigma_k]^{i_2}{}_{j_1}\,y_k{}^{j_2}$

For b: $d_{j_1} = -\nabla_b L(\theta) = \dfrac{2}{K}\sum_{k=0}^{K-1}\epsilon_{k,i_1}B^{i_1}{}_{i_2}[\nabla\sigma_k]^{i_2}{}_{j_1}$

For B: $d_{j_1,j_2} = -\nabla_B L(\theta) = \dfrac{2}{K}\sum_{k=0}^{K-1}\epsilon_{k,j_1}\sigma_k{}^{j_2}$

Discrete Probability Density:

$P\{X \in A\} = \sum_{i\in A} p_i$



Expected, Mean, Variance:

$\mu = E[X] = \sum_{n=0}^{M-1} n\,p_n$ *Expected Value of X*

$E[f(X)] = \sum_{n=0}^{M-1} f(n)\,p_n$ *Expected Value of f(X)*

$\sigma^2 = E[(X-\mu)^2] = \sum_{n=0}^{M-1}(n-\mu)^2 p_n$ *Variance X*

Conditional Density:

$$p_{x|y}(i|j) = P\{X = i|Y = j\} = \frac{P\{X = i, Y = j\}}{P\{Y = j\}} = \frac{p(i,j)}{p_y(j)} = \frac{p(i,j)}{\sum_{i=0}^{M-1}p(i,j)}$$

$$p_{y|x}(j|i) = P\{Y = j|X = i\} = \frac{P\{X = i, Y = j\}}{P\{X = i\}} = \frac{p(i,j)}{p_x(i)} = \frac{p(i,j)}{\sum_{j=0}^{M-1}p(i,j)}$$

Conditional Expectation:

$$E[f(X)|Y] = \sum_{n=0}^{M-1} f(n)\,p_{x|y}(n|Y)\quad E[f(Y)|X] = \sum_{n=0}^{M-1} f(n)\,p_{y|x}(n|X)$$

Frequentist View: (6)

MLE: $\hat{\theta} = \arg\max_\theta\{p_\theta(X,Y)\}$
$= \arg\min_\theta\{-\log p_\theta(X,Y)\}$

Regression

$\hat{\theta} = \arg\min_\theta\{-\log p_\theta(x,y)\}$
$= \arg\min_\theta\{-\log p_\theta(x|y) - \log p(y)\}$
$= \arg\min_\theta L_{MSE}(\theta; x, y)$

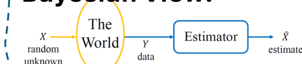
Classification (w/ 1-hot encoding)

$\hat{\theta} = \arg\min_\theta\{-\log p_\theta(x,y)\}$
$= \arg\min_\theta\{-\log p_\theta(x|y) - \log p(y)\}$
$= \arg\min_\theta\{L_{CE}(\theta; x, y)\}$

Multinomial Distro

$\hat{\theta} = \arg\min_\theta\left\{-\sum_{i=0}^{M-1}\frac{N_i}{N}\log\theta_i\right\} = \frac{N_i}{N}$

Bayesian View:



Bayes Law:

$$p_{x|y}(x|y) = \frac{p_{y|x}(y|x)\,p_x(x)}{p_y(y)} = \frac{p_{y|x}(y|x)\,p_x(x)}{\sum_{k=0}^{M-1}p_{y|x}(y|k)\,p_x(k)}$$

Min Mean Square Error

$\hat{X} = E[X|Y] = \sum_{x=0}^{M-1} x\,p_{x|y}(x|Y)$

MAP Estimate

$= \arg\min_x\{-\log p_{y|x}(y|x) - \log p_x(x)\}$

Bias and Variance

Frequentist

$\text{bias}_\theta = E[\hat{\theta}|\theta] - \theta$

$\text{Var}_\theta = E\left[\|\hat{\theta} - E[\hat{\theta}|\theta]\|^2\,\middle|\,\theta\right]$

$\text{MSE}_\theta = \text{Var}_\theta + (\text{bias}_\theta)^2$

Bayesian

$\text{bias}_x = E[\hat{X}|X = x] - x$

$\overline{\text{bias}^2} = E\left[\|E[\hat{X}|X] - X\|^2\right]$

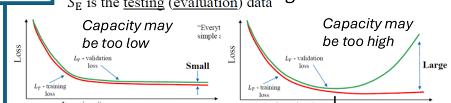
$\text{Var} = E\left[\|\hat{X} - E[\hat{X}|X]\|^2\right]$

$\text{MSE} = \text{Var} + \overline{\text{bias}^2}$

Data partitions:

S_T is the training data
S_V is the validation data
S_E is the testing (evaluation) data

Each partition has the same loss function but different "ground truth" data



Overfitting Fixes: (7)

Regularization – adds penalties to weights, encouraging them to be small or collapse to 0

$S(\theta) = -\log p(\theta)$ — *MAP estimate*

$S(\theta) = \|\theta\|^2$ — *Like a Gaussian Prior. Reduces amplitude of weights*

$S(\theta) = \|\theta\|_1$ — *Like a Laplacian Prior. Encourages weights to go to zero*

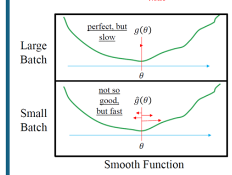
Dropout – randomly drop nodes in network

For each batch{
For each layer l {
$r^{(l)} \leftarrow Bernoulli(p, shape(y^{(l)}))$

For $n = 0$ to $K_b - 1$ {
For each layer l {
$\tilde{y}^{(l)} \leftarrow r^{(l)}.*\,y^{(l)}$ Dropout
$z^{(l+1)} \leftarrow w^{(l+1)} * \tilde{y}^{(l)}$
$y^{(l+1)} \leftarrow f(z^{(l+1)})$

Stochastic Gradient Descent: (8)

$\hat{g} = g + \dfrac{w}{\sqrt{K_b}}$ (Batch Gradient = True Gradient + Noise)



SGD w/ Momentum:

init $v \leftarrow 0$
Repeat until converged {
Repeat $b = 1$ to B {
$d \leftarrow -\nabla L(\theta; S_b)$
$v \leftarrow \gamma v + \alpha(1-\gamma)d$
$\theta \leftarrow \theta + v^t$

ADAM Optimizer: *Momentum+Preconditioning*

init $v \leftarrow 0$; $r \leftarrow 0$;
init $t \leftarrow 0$
Repeat until converged {
Repeat $b = 1$ to B {
$t \leftarrow t + 1$
$d \leftarrow -\nabla L(\theta; S_b)$
$v \leftarrow \beta_1 v + (1-\beta_1)d$
$r \leftarrow \beta_2 r + (1-\beta_2)d^2$
$\hat{v} \leftarrow v/(1-\beta_1^t)$
$\hat{r} \leftarrow r/(1-\beta_2^t)$
$\theta \leftarrow \theta + \alpha(\sqrt{\hat{r}} + \epsilon)^{-1}\hat{v}$

Batch (9)

Input parameters:
ϵ – small number to regularize division
α – momentum parameter
Learned parameters:
γ, β – scaling and offset parameters
Estimated parameters:
μ_l, σ_l^2 – learned mean and variance parameters
Parameters saved for inference:
μ_l, σ_l^2 – learned mean and variance parameters
γ, β, ϵ – same as above

Training

Initialize $\mu_l \leftarrow 0$; $\sigma_l^2 \leftarrow$
For each b (batch) {
$\mu_B \leftarrow \dfrac{1}{K_b}\sum_{k\in S_b} z_k$
$\sigma_B^2 \leftarrow \dfrac{1}{K_b - 1}\sum_{k\in S_b}(z_k - \mu_B)^2$
For each $k \in S_b$ {
$\hat{z}_k \leftarrow \dfrac{z_k - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$
$y_k \leftarrow \gamma\hat{z}_k + \beta$
}
$\mu_l \leftarrow \alpha\,\mu_l + (1-\alpha)\mu_B$
$\sigma_l^2 \leftarrow \alpha\,\sigma_l^2 + (1-\alpha)\sigma_B^2$

Inference

puts:
ϵ – same constant as used in training
γ, β – learned from training
μ_l, σ_l^2 – estimated during training
For each input {
$\hat{z} \leftarrow \dfrac{z - \mu_l}{\sqrt{\sigma_l^2 + \epsilon}}$
$y \leftarrow \gamma\hat{z} + \beta$

Denoising:

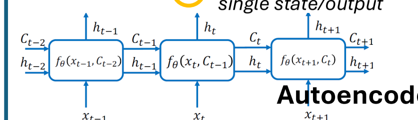
$\hat{x} = E[x|y]$
$= E[y - w|y]$
$= y - E[w|y]$
$= y - f_\theta(y)$
where $\hat{w} = f_{\hat{\theta}}(y)$

Positional Encoding:

$P(t, 2i) = \sin\left(\dfrac{2\pi}{\lambda^i}t\right)$

$P(t, 2i+1) = \cos\left(\dfrac{2\pi}{\lambda^i}t\right)$

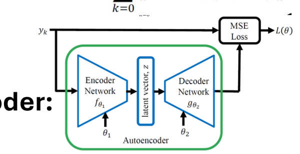
RNN - LSTM: (10)



GRU: *Combines C and h into single state/output*

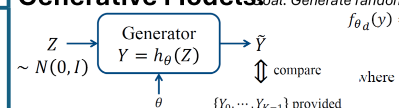
Autoencoder:

$Loss(\theta) = \sum_{k=0}^{K-1}\left\|y_k - g_{\theta_2}\left(f_{\theta_1}(y_k)\right)\right\|^2$



Generative Models:

Goal: Generate random vectors that "look" like ground truth



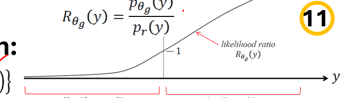
$\{Y_0, \cdots, Y_{K-1}\}$ provided as training samples

Log Likelihood Ratio: (11)

$f_{\theta_d}(y) \approx P\{Class = R|y\}$

$= \dfrac{p_r(y)}{p_r(y) + p_{\theta_g}(y)} = \dfrac{1}{1 + R_{\theta_g}(y)}$

where $R_{\theta_g}(y) = \dfrac{p_{\theta_g}(y)}{p_r(y)}$



Discriminator Loss Function:

$$\hat{d}(\theta_g, \theta_d) = \frac{1}{K}\sum_{k=0}^{K-1}\left\{-\log f_{\theta_d}(y_k) - \log\left(1 - f_{\theta_d}(\tilde{y}_k)\right)\right\}$$

G/D Loss Function $g(\theta_g, \theta_d) = E\left[-\log f_{\theta_d}\left(\tilde{Y}_{\theta_g}\right)\right]$

$d(\theta_g, \theta_d) = E[-\log f_{\theta_d}(Y)] + E\left[-\log\left(1 - f_{\theta_d}\left(\tilde{Y}_{\theta_g}\right)\right)\right]$

$\theta_g^* = \arg\min_{\theta_g} g(\theta_g, \theta_d^*)$

$= \arg\min_{\theta_g} E\left[-\log\left(f_{\theta_d}\left(\tilde{Y}_{\theta_g}\right)\right)\right]$

$\theta_d^* = \arg\min_{\theta_d} d(\theta_g^*, \theta_d)$

$= \arg\min_{\theta_d}\left\{E[-\log f_{\theta_d}(Y)] + E\left[-\log\left(1 - f_{\theta_d}\left(\tilde{Y}_{\theta_g}\right)\right)\right]\right\}$

Nash Equilibrium:

$g(\theta_g^*, \theta_d^*) = \min_{\theta_g} g(\theta_g, \theta_d^*)$

$d(\theta_g^*, \theta_d^*) = \min_{\theta_d} d(\theta_g^*, \theta_d)$

2 Agents:

Zero Sum:

$d(\theta_g^*, \theta_d^*) = \max_{\theta_g} d(\theta_g, \theta_d^*)$

$d(\theta_g^*, \theta_d^*) = \min_{\theta_d} d(\theta_g^*, \theta_d)$