Understanding Deep Learning Errata

November 21, 2024

Much gratitude to everyone who has pointed out mistakes. If you find a problem not listed here, please contact me via github or by mailing me at udlbookmail@gmail.com.

Instructions

To find which errata are relevant to your version of the book, first consult the copyright page at the start of the book just before the dedication. The printing will be stated (e.g., "Second printing") just before the line that says "Library of Congress...". If it doesn't specify the printing here, then you have the first printing.

This document is organized by printing. If you have the first printing of the book, all errata are relevant to you. If you have the second printing, then only the errata in the sections for second and third printings are relevant. If you have the third printing, then only the errata for the third printing is relevant and so on.

Errors

These are things that might genuinely confuse you. There is a list of more minor changes (e.g., math symbols that are in bold but should not be, missing brackets, slight rephrasings, changing tiny things for consistency with other chapters) at the end of this document.

Fourth printing (Jun 2024)

There have been only seven errors reported since February 2024 when the second printing was submitted. The book is very stable, so don't be put off from buying a physical copy.

- Figure 12.7 (and legend): Changed to reflect the fact that LayerNorm is applied separately to each embedding. See figure 1.1 of this document. Text in Section 12.4 changed to "This is similar to BatchNorm but normalizes each embedding in each batch element separately using statistics calculated across its D embedding dimensions."
- Figure 14.5: The precision can be computed as the proportion of ~~real examples~~ samples that lie within the approximated manifold of ~~samples~~ real examples . Similarly, the recall is computed as the proportion of ~~real examples~~ samples that lie within the approximated manifold of ~~samples~~ real examples (not shown).
- Problem 15.1: What will the loss be in equation 15.9 when $Pr(\mathbf{x}^*) = Pr(\mathbf{x})$?
- Section 16.3.1 This can be inverted in $\mathcal{O}[D^2]$, and the log determinant is just the sum of the log of the absolute values on the diagonals of $\mathbf{L}$ and $\mathbf{D}$
- Section 16.3.2 (and similar changes in problem 16.9). A simple example is a piecewise linear function with K regions (figure 16.5) which maps $[0, 1]$ to $[0, 1]$ as:

$$\text{f}[h, \boldsymbol{\phi}] = \left(\sum_{k=1}^{b-1} \phi_k\right) + (hK - b{+}1)\phi_b, \tag{1.1}$$

  where the parameters $\phi_1, \phi_2, \ldots, \phi_K$ are positive and sum to 1, and $b = \lfloor Kh \rfloor{+}1$ is the index of the bin that contains h.

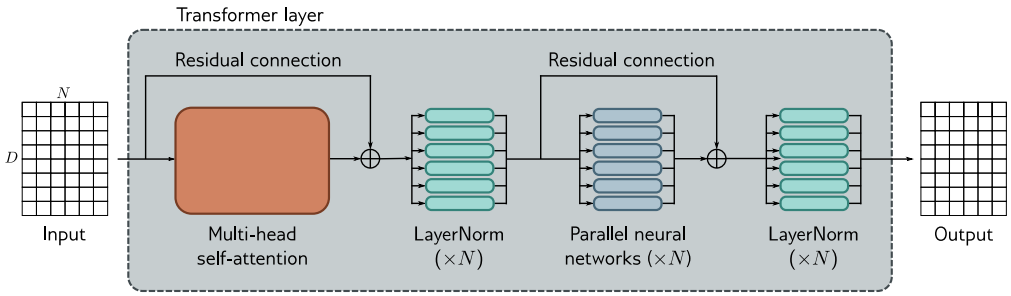


Figure 1.1 Corrected version of figure 12.7. Transformer layer. The input consists of a $D \times N$ matrix containing the D-dimensional word embeddings for each of the N input tokens. The output is a matrix of the same size. The transformer layer consists of a series of operations. First, there is a multi-head attention block, allowing the word embeddings to interact with one another. This forms the processing of a residual block, so the inputs are added back to the output. Second, a LayerNorm operation is applied separately to each embedding. Third, there is a second residual layer where the same fully connected neural network is applied separately to each of the N word representations (columns). Finally, LayerNorm is applied again.

- Equation 16.22:

$$\begin{aligned}\log\left[\left|\mathbf{I}+\frac{\partial \mathbf{f}[\mathbf{h},\boldsymbol{\phi}]}{\partial \mathbf{h}}\right|\right] &= \operatorname{trace}\left[\log\left[\mathbf{I}+\frac{\partial \mathbf{f}[\mathbf{h},\boldsymbol{\phi}]}{\partial \mathbf{h}}\right]\right] \\ &= \sum_{k=1}^{\infty}\frac{(-1)^{k-1}}{k}\operatorname{trace}\left[\frac{\partial \mathbf{f}[\mathbf{h},\boldsymbol{\phi}]}{\partial \mathbf{h}}\right]^{k},\end{aligned}$$

- Appendix B.1 Definition of surjection was wrong. Should be: A *surjection* is a function where every element in the second set receives a mapping from the first (but there may be multiple elements of the first set that are mapped to the same element of the second set).

Second and third printings (Mar. 2024)

The book was reprinted twice around March 2024.

- Equation 6.12

$$\begin{aligned}\mathbf{m}_{t+1} &\leftarrow \beta\cdot\mathbf{m}_t + (1-\beta)\sum_{i\in\mathcal{B}_t}\frac{\partial \ell_i[\boldsymbol{\phi}_t - \alpha\beta\cdot\mathbf{m}_t]}{\partial\boldsymbol{\phi}}\\ \boldsymbol{\phi}_{t+1} &\leftarrow \boldsymbol{\phi}_t - \alpha\cdot\mathbf{m}_{t+1},\end{aligned} \tag{1.2}$$

- Equation 6.18:

$$\begin{aligned}\mathbf{m}_{t+1} &\leftarrow \beta\cdot\mathbf{m}_t + (1-\beta)\sum_{i\in\mathcal{B}_t}\frac{\partial \ell_i[\boldsymbol{\phi}_t]}{\partial\boldsymbol{\phi}}\\ \mathbf{v}_{t+1} &\leftarrow \gamma\cdot\mathbf{v}_t + (1-\gamma)\left(\sum_{i\in\mathcal{B}_t}\frac{\partial \ell_i[\boldsymbol{\phi}_t]}{\partial\boldsymbol{\phi}}\right)^2,\end{aligned} \tag{1.3}$$

- Equation 15.6

$$\begin{aligned}L[\boldsymbol{\phi}] &= -\frac{1}{J}\sum_{j=1}^{J}\Big(\log\Big[1-\text{sig}[\text{f}[\mathbf{x}_j^*,\boldsymbol{\phi}]]\Big]\Big) - \frac{1}{I}\sum_{i=1}^{I}\Big(\log\Big[\text{sig}[\text{f}[\mathbf{x}_i,\boldsymbol{\phi}]]\Big]\Big)\\ &\approx -\mathbb{E}_{\mathbf{x}^*}\Big[\log\Big[1-\text{sig}[\text{f}[\mathbf{x}^*,\boldsymbol{\phi}]]\Big]\Big] - \mathbb{E}_{\mathbf{x}}\Big[\log\Big[\text{sig}[\text{f}[\mathbf{x},\boldsymbol{\phi}]]\Big]\Big]\\ &= -\int Pr(\mathbf{x}^*)\log\Big[1-\text{sig}[\text{f}[\mathbf{x}^*,\boldsymbol{\phi}]]\Big]d\mathbf{x}^* - \int Pr(\mathbf{x})\log\Big[\text{sig}[\text{f}[\mathbf{x},\boldsymbol{\phi}]]\Big]d\mathbf{x},\end{aligned} \tag{1.4}$$

- Equation 15.8

$$\begin{aligned}L[\boldsymbol{\phi}] &= -\int Pr(\mathbf{x}^*)\log\Big[1-\text{sig}[\text{f}[\mathbf{x}^*,\boldsymbol{\phi}]]\Big]d\mathbf{x}^* - \int Pr(\mathbf{x})\log\Big[\text{sig}[\text{f}[\mathbf{x},\boldsymbol{\phi}]]\Big]d\mathbf{x}\\ &= -\int Pr(\mathbf{x}^*)\log\left[1-\frac{Pr(\mathbf{x})}{Pr(\mathbf{x}^*)+Pr(\mathbf{x})}\right]d\mathbf{x}^* - \int Pr(\mathbf{x})\log\left[\frac{Pr(\mathbf{x})}{Pr(\mathbf{x}^*)+Pr(\mathbf{x})}\right]d\mathbf{x}\\ &= -\int Pr(\mathbf{x}^*)\log\left[\frac{Pr(\mathbf{x}^*)}{Pr(\mathbf{x}^*)+Pr(\mathbf{x})}\right]d\mathbf{x}^* - \int Pr(\mathbf{x})\log\left[\frac{Pr(\mathbf{x})}{Pr(\mathbf{x}^*)+Pr(\mathbf{x})}\right]d\mathbf{x}.\end{aligned} \tag{1.5}$$

- Equation 19.40:

$$r[\boldsymbol{\tau}_{it}] \approx r_{it} + \gamma\cdot v[\mathbf{s}_{i,t+1},\boldsymbol{\phi}].$$

- Section B.3.6: Definition of nullspace was ambiguous/wrong. Last line of this section changed to: Conversely, for a landscape matrix $\mathbf{A}$, the part of the input space that maps to zero (i.e., those $\mathbf{x}$ where $\mathbf{A}\mathbf{x}=\mathbf{0}$) is termed the *nullspace* of the matrix.

- Section C.5.3: The definition of the Fréchet distance was incorrect. Changed to:

$$D_{Fr}\Big[p(x)\Big|\Big|q(y)\Big] = \sqrt{\min_{\pi(x,y)}\left[\iint \pi(x,y)|x-y|^2 dxdy\right]}, \tag{1.6}$$

where $\pi(x,y)$ represents the set of joint distributions that are compatible with the marginal distributions $p(x)$ and $q(y)$. The Fréchet distance can also be formulated as a measure of the maximum distance between the cumulative probability curves.

- Equation C.32:

$$D_{KL}\Big[\text{Norm}[\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1]\Big|\Big|\text{Norm}[\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2]\Big] = \frac{1}{2}\left(\log\left[\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|}\right] - D + \text{tr}\left[\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\right] + (\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)\right). \tag{1.7}$$

First printing (Dec. 2023)

These are things that might genuinely confuse you:

- Figure 4.7b had the wrong calculated numbers in it (but pattern is same). Correct version is in figure 1.2 of this document.
- Section 6.3.1 where now the gradients are evaluated at $\boldsymbol{\phi}_t - \alpha\beta \cdot \mathbf{m}_t$.
- Section 7.5.1 The expectation (mean) $\mathbb{E}[f_i']$ of the intermediate values f_i' is:
- Equation 15.7 The optimal discriminator for an example $\tilde{\mathbf{x}}$ depends on the underlying probabilities:

$$Pr(\text{real}|\tilde{\mathbf{x}}) = \text{sig}\big[\text{f}[\tilde{\mathbf{x}}, \boldsymbol{\phi}]\big] = \frac{Pr(\tilde{\mathbf{x}}|\text{real})}{Pr(\tilde{\mathbf{x}}|\text{generated}) + Pr(\tilde{\mathbf{x}}|\text{real})} = \frac{Pr(\mathbf{x})}{Pr(\mathbf{x}^*) + Pr(\mathbf{x})}. \tag{1.8}$$

where on the right hand side, we evaluate $\tilde{\mathbf{x}}$ against the generated distribution $Pr(\mathbf{x}^*)$ and the real distribution $Pr(\mathbf{x})$.

- Equation 15.9. First integrand should be with respect to $\mathbf{x}^*$. Correct version is:

$$\begin{aligned} &D_{JS}\Big[Pr(\mathbf{x}^*) \,||\, Pr(\mathbf{x})\Big] \\ &= \frac{1}{2}D_{KL}\left[Pr(\mathbf{x}^*)\left|\left|\frac{Pr(\mathbf{x}^*)+Pr(\mathbf{x})}{2}\right.\right.\right] + \frac{1}{2}D_{KL}\left[Pr(\mathbf{x})\left|\left|\frac{Pr(\mathbf{x}^*)+Pr(\mathbf{x})}{2}\right.\right.\right] \\ &= \underbrace{\frac{1}{2}\int Pr(\mathbf{x}^*)\log\left[\frac{2Pr(\mathbf{x}^*)}{Pr(\mathbf{x}^*)+Pr(\mathbf{x})}\right]d\mathbf{x}^*}_{\text{quality}} + \underbrace{\frac{1}{2}\int Pr(\mathbf{x})\log\left[\frac{2Pr(\mathbf{x})}{Pr(\mathbf{x}^*)+Pr(\mathbf{x})}\right]d\mathbf{x}}_{\text{coverage}}. \end{aligned}$$

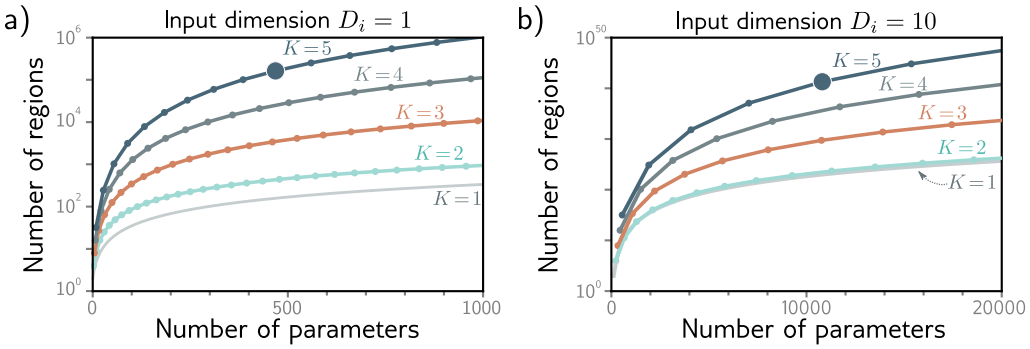


Figure 1.2 Corrected version of figure 4.7: The maximum number of linear regions for neural networks increases rapidly with the network depth. a) Network with $D_i = 1$ input. Each curve represents a fixed number of hidden layers K, as we vary the number of hidden units D per layer. For a fixed parameter budget (horizontal position), deeper networks produce more linear regions than shallower ones. A network with $K = 5$ layers and $D = 10$ hidden units per layer has 471 parameters (highlighted point) and can produce 161,051 regions. b) Network with $D_i = 10$ inputs. Each subsequent point along a curve represents ten hidden units. Here, a model with $K = 5$ layers and $D = 50$ hidden units per layer has 10,801 parameters (highlighted point) and can create more than 10^{40} linear regions.

- Equation 15.12.

$$D_w\Big[Pr(x)||q(x)\Big] = \max_{\mathbf{f}}\left[\sum_i Pr(x=i)f_i - \sum_j q(x=j)f_j\right], \tag{1.9}$$

- Section 15.2.4 Consider distributions $Pr(x = i)$ and $q(x = j)$ defined over K bins. Assume there is a cost C_{ij} associated with moving one unit of mass from bin i in the first distribution to bin j in the second;

- Equation 15.14. Missing bracket and we don't need to use $\mathbf{x}^*$ notation here. Correct version is:

$$D_w\Big[Pr(\mathbf{x}), q(\mathbf{x})\Big] = \min_{\pi[\bullet,\bullet]}\left[\int\int \pi(\mathbf{x}_1,\mathbf{x}_2)\cdot||\mathbf{x}_1-\mathbf{x}_2||d\mathbf{x}_1 d\mathbf{x}_2\right].$$

- Equation 15.15. Don't need to use $\mathbf{x}^*$ notation here, and second term on right hand side should have $q[\mathbf{x}]$ term not $Pr(\mathbf{x})$. Correct version is:

$$D_w\Big[Pr(\mathbf{x}), q(\mathbf{x})\Big] = \max_{\mathrm{f}[\mathbf{x}]}\left[\int Pr(\mathbf{x})\mathrm{f}[\mathbf{x}]d\mathbf{x} - \int q(\mathbf{x})\mathrm{f}[\mathbf{x}]d\mathbf{x}\right].$$

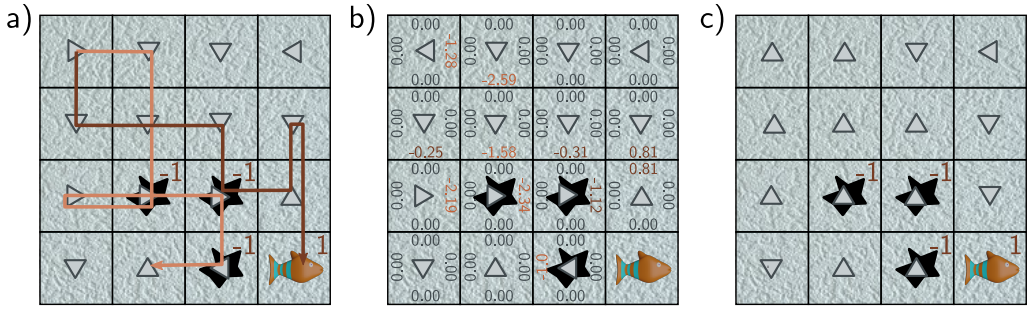


Figure 1.3 Corrected version of figure 19.11

- Equation 16.12 has a mistake in the second term. It should be:

$$\mathrm{f}[h_d, \boldsymbol{\phi}] = \left(\sum_{k=1}^{b-1} \phi_k\right) + (hK - b)\phi_b.$$

- Equation 17.34.

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\phi}} \mathbb{E}_{Pr(x|\boldsymbol{\phi})}\big[\mathrm{f}[x]\big] &= \mathbb{E}_{Pr(x|\boldsymbol{\phi})}\left[\mathrm{f}[x]\frac{\partial}{\partial \boldsymbol{\phi}}\log\big[Pr(x|\boldsymbol{\phi})\big]\right] \\ &\approx \frac{1}{I}\sum_{i=1}^{I}\mathrm{f}[x_i]\frac{\partial}{\partial \boldsymbol{\phi}}\log\big[Pr(x_i|\boldsymbol{\phi})\big].\end{aligned}$$

- Figure 19.11 is wrong in that only the state-action values corresponding to the current state-action pair should be moderated. Correct version above.

- Equation B.4. Square root sign should cover x. Correct version is:

$$x! \approx \sqrt{2\pi x}\left(\frac{x}{e}\right)^x.$$

- Appendix B.3.6. Consider a matrix $\mathbf{A} \in \mathbb{R}^{D_1 \times D_2}$. If the number of columns D_2 of the matrix is fewer than the number of rows D_1 (i.e., the matrix is "portrait"),

- Equation C.20. Erroneous minus sign on covariance matrix. Correct version is:

$$\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\mathbf{z}.$$

Minor fixes

These are mostly tiny changes that almost certainly won't affect your understanding (e.g., math symbols that are in bold but should not be, missing brackets, slight re-phrasings, changing tiny things for consistency with other chapters). You probably won't even notice them, and most people wouldn't bother making this type of change once the book was already published. I'm mainly listing them to stop people submitting duplicates, and to help translators.

Fourth printing (Jun 2024)

- Chapter 1 introduction: A *deep neural network* (or *deep network* for short) is a type of machine learning model, and when it is fitted to data, this is referred to as *deep learning*. (+ other minor changes to preserve formatting)
- Chapter 1 introduction: Hence, this chapter also contains a brief primer on AI ethics.
- Section 1.1.2 For the music classification example, the input vector might be of fixed size (perhaps a 10-second clip) but is very high-dimensional (i.e., contains many entries).
- Section 1.1.2 Moreover, ~~structure is naturally two-dimensional~~ it contains spatial structure; two pixels above and below one another are closely related, even if they are not adjacent in the input vector.
- Section 1.2.2 Similarly, real-world images are a tiny subset of the images that can be created by drawing random red, green, and blue (RGB) values for every pixel. (+ other minor changes to preserve formatting)
- Section 1.4 They may ~~contain billions of parameters~~ be enormous, and there is no way we can understand how they work based on examination. (the term parameters is not defined yet)
- Section 1.5 Why do they need ~~so many parameters~~ to be so large? (the term parameters is not defined yet).

- Section 2.1. Added marginal link to appendix for argmin function
- Figure 2.1 legend and three other places in this chapter Removed transpose from parameter vector (unnecessary and confused some people).
- Figure 2.2 Parameters and line in figure 2.2d changed to $\phi_0 = 0.82, \phi_1 = 0.52$ as this gives a slightly lower loss.
- Equation 2.5 Standard ϕ on left hand side changed to bold $\boldsymbol{\phi}$.
- Figure 3.5 The symbol D_i is not defined. Added the sentence: This idea generalizes to functions in D_i dimensions.
- Section 4.4.1 Function $\mathbf{f}[\bullet]$ should be bold. New version: A general deep network $\mathbf{y} = \mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ with K layers can now be written as
- Figure 4.6 legend: The weights are stored in matrices $\boldsymbol{\Omega}_k$ that ~~pre-~~multiply the activations from the preceding layer.
- Chapter 4 in notes. Lu et al. (2017) proved that there exists a network with ReLU activation functions and at least $D_i + 4$ hidden units in each layer that can approximate any specified D_i-dimensional Lebesgue integrable function to arbitrary accuracy given enough layers.
- Chapter 5 introduction: Changed marginal link to ~~Number~~ Sets for consistency with the appendix.
- Section 5.1.1 This shift in perspective raises the question of
- Section 5.2: To perform inference for a new test example $\mathbf{x}$, return either the full distribution $Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \hat{\boldsymbol{\phi}}])$ or the value where this distribution is maximized.
- Section 5.3.1: We see that the least squares loss function follows naturally from the assumptions that the predictions are (i) independent and (ii) drawn from a normal distribution with mean $\mu = \mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}]$.
- Section 5.3.1: We removed the denominator between the third and fourth lines, as this is just a constant positive scaling factor that does not affect the position of the minimum. (other minor changes to preserve formatting)
- Equation 5.12: Inner close square bracket in wrong place

$$\hat{y} = \underset{y}{\operatorname{argmax}} \left[Pr(y|\mathrm{f}[\mathbf{x}, \hat{\boldsymbol{\phi}}], \sigma^2 \right].$$

- Section 5.3.3 However, there is nothing to stop us from treating σ^2 as a learned parameter ~~of the model~~and minimizing equation 5.9 with...
- Figure 5.10 legend. ...any vertical slice of this plot produces three values that sum to one

- Section 5.5. Clarity improved by referring to both uses of this notation. where $\mathrm{f}_{y_i}[\mathbf{x}, \boldsymbol{\phi}]$ and $\mathrm{f}_{k'}[\mathbf{x}, \boldsymbol{\phi}]$ denote the y_i^{th} and k'^{th} outputs of the network, respectively.
- Figure 5.12: Model distribution (a normal distribution with parameters $\boldsymbol{\theta} = \{\mu, \sigma^2\}$).
- Figure 5.14: here, two ~~weighted~~ normal distributions, dashed blue and orange curves
- Section 6.1.2 More formally, they are convex, which means that every ~~no~~ chord (line segment between two points on the surface) ~~intersects the function~~ lies above the function and does not intersect it.
- Chapter 6 notes. A function is convex if colorred every ~~no~~ chord (line segment between two points on the surface) ~~intersects the function~~ lies above the function and does not intersect it.
- Chapter 6 notes. However, this is strange since SGD is a special case of Adam (when ~~$\beta = \gamma = 0$~~ $\beta = 0, \gamma = 1$) once the modification term (equation 6.16) becomes one, which happens quickly.
- Problem 6.1 A surface is guaranteed to be convex if the eigenvalues of the Hessian $\mathbf{H}[\boldsymbol{\phi}]$ are positive everywhere.
- Equation 7.7 (new label added to unlabelled equation between previous equation 7.6 and 7.7)
- Section 7.4 Now let's consider how the loss changes when the pre-activations $\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2$ change.
- Equation 7.26 (formerly, now eq 7.27). Activation should be bolded:

$$\begin{aligned} \mathbf{f}_k &= \boldsymbol{\beta}_k + \boldsymbol{\Omega}_k \mathbf{h}_k \\ &= \boldsymbol{\beta}_k + \boldsymbol{\Omega}_k \mathbf{a}[\mathbf{f}_{k-1}], \end{aligned}$$

- Section 7.5.1 where ~~$\mathbf{f}$~~ $\mathbf{h}$ represents the ~~pre-~~activations
- Section 7.5.1 where $\mathbf{h}$ represents the activations, $\boldsymbol{\Omega}$~~,~~ and $\boldsymbol{\beta}$ represent the weights and biases, and $\mathbf{a}[\bullet]$ is the activation function.
- Section 7.5.1 Assuming that the ~~input~~ distribution of pre-activations f_j at the previous layer is symmetric about zero
- Equations 7.29 and 7.30 + nearby text. Added subscript to variance so $\sigma^2_{f'} \rightarrow \sigma^2_{f'_i}$.
- Problem 7.6. Equation number added.
- Section 8.1 This is better than the chance error rate of 90% ~~error rate~~ but far worse than for the training set;

- Section 8.2 To make this easier to visualize, we revert to a 1D ~~linear~~ least squares regression problem
- Section 8.2.2 They combine additively for ~~linear~~ regression tasks with a least squares loss. However, their interaction can be more complex for other types of problems
- Section 8.3 there is nothing we can do to circumvent this, and it represents a fundamental limit on expected model performance
- Section 8.3.3: For a fixed-size training dataset, the variance term typically increases as the model capacity increases.
- Section 8.4.1: Rephrased sentence to avoid potential ambiguity: Second, the test performance continues to improve with capacity even when this exceeds the point where the training data are all classified correctly.
- Figure 8.10. Legend. Training and test ~~loss~~ erroron MNIST-1D... The training and test ~~loss~~ errordecreases to zero... Depending on the loss function, the model, and the amount of noise in the data
- Section 9.1 We seek the parameters $\hat{\boldsymbol{\phi}}$ that minimize ~~minimum of~~ the loss function $L[\boldsymbol{\phi}]$:
- Section 9.1 where g$[\boldsymbol{\phi}]$ is a function which ~~that~~ returns a scalar that takes ~~a~~larger value~~s~~ when the parameters are less preferred
- Figure 9.2 legend: L2 regularization in simplified network with 14 hidden units.
- Figure 9.2 legend: For large λ (panels e–f), the regularization term overpowers the likelihood term, so the fitted function is too smooth and the overall ~~the fitted function is smoother than the ground truth, so the~~ fit is worse.
- Section 9.2.1: (see notes "Implicit regularization in gradient descent" at end of chapter)
- Section 9.2.2: if the model is over-parameterized, then it may fit all the training data exactly, so each ~~all~~ of these gradient terms will ~~all~~ be zero at the global minimum.
- Section 9.3: We've seen that ~~adding~~ explicit regularization ~~terms~~ encourages the training algorithm to find a good solution by adding extra terms to the loss function.
- Section 9.3.1: The model is trained once, the performance on the validation set is monitored every T iterations, and the associated parameters ~~models~~ are stored. The stored parameters ~~models~~ where the validation performance was best are ~~is~~ selected.
- Section 9.3.3 This makes the network less dependent on any given hidden unit and encourages the weights to have smaller magnitudes so that the change in the function due to the presence or absence of any specific hidden ~~the~~ unit is reduced.

- 9.3.3 *Dropout* ~~randomly~~ clamps a random subset (typically 50%) of hidden units to zero at each iteration of SGD.

- 9.3.3. When several units conspire in this way, eliminating one (as would happen in dropout) causes a considerable change to the output function in ~~that is propagated to~~ the half-space where that unit was active

- Section 9.3.5 The maximum likelihood approach is generally overconfident; ~~in the training phase~~ it selects the most likely parameters during training and uses these to make predictions ~~and bases its predictions on the model defined by these.~~

- Chapter 9 Notes Notably missing from the discussion in this chapter is BatchNorm ~~at~~ and its variants

- Chapter 10 and 11 (multiple places) zero padding → zero-padding

- Section 10.2.5 If we only apply a single convolution, information will likely ~~inevitably~~ be lost;

- Section 10.2.5 If the incoming layer has C_i channels and we select a kernel size K per channel, the hidden units in each output channel are computed as a weighted sum over all C_i channels and K kernel entries ~~positions~~ using a weight matrix $\boldsymbol{\Omega} \in \mathbb{R}^{C_i \times K}$ and one bias. Hence, if there are C_o channels in the next layer, then we need $\boldsymbol{\Omega} \in \mathbb{R}^{C_i \times C_o \times K}$ weights and $\boldsymbol{\beta} \in \mathbb{R}^{C_o}$ biases.

- Figure 10.7 legend: The MNIST-1D input has dimension $D_i = 40$. The first convolutional layer has fifteen channels, kernel size three, stride two, and only retains "valid" positions to make a ~~representation~~ hidden layer with nineteen positions and fifteen channels. The following two convolutional layers have the same settings, gradually reducing the representation size at each subsequent hidden layer.

- Section 10.4.3 Combined with a bias and activation function, it is equivalent to running the same fully connected network on the input channels at every position.

- Section 10.6 As ~~the network progresses~~ a data example passes through the network, the spatial dimensions usually decrease by factors of two, and the ~~number of~~ channels increase~~s~~by factors of two.

- Chapter 10 Notes p183. However, the ConvolutionOrthogonal initializer (Xiao et al., 2018a) is specialized for convolutional networks ~~(Xiao et al., 2018a)~~.

- Chapter 10 Notes p184 (Lin et al., 2017b) ~~Lin et al. (2017b)~~, (Redmon et al., 2016) ~~Redmon et al. (2016)~~.

- Problem 10.4 Write out the equation for a 1D convolution with kernel size seven, dilation rate of three, and stride of three. ~~You may assume that the input is padded with zeros at positions x_{-2}, x_{-1} and x_0~~

- Problem 10.6. Draw a 12×6 ~~6×12~~ weight matrix in the style of figure 10.4d relating inputs $x_1, \ldots, x_6$ to outputs $h_1, \ldots, h_{12}$ in the multi-channel convolution as depicted in figures 10.5a–b.

- Problem 10.7. Draw a 6×12 ~~12× 6~~ weight matrix in the style of figure 10.4d relating inputs $h_1, \ldots, h_{12}$ to outputs $h'_1, \ldots, h'_6$ in the multi-channel convolution in figure 10.5c.

- Problem 10.15 Show that the weight matrix for 1D convolution with kernel size three and stride two is equivalent to composing the matrices for 1D convolution with kernel size three and stride one and this sampling matrix.

- Problem 10.17 What is the receptive field size at each of the first three layers of AlexNet (i.e., the first three orange blocks in figure 10.16)?

- Chapter 11 Introduction: The previous chapter described how image classification performance improved as the depth of convolutional networks was extended from eight layers (AlexNet) to ~~eighteen~~ nineteen layers (VGG).

- Section 11.1.1 In principle, we can add as many layers as we want, and in the previous chapter, we saw that adding more layers to a convolutional network does improve performance; the VGG network (figure 10.17), which has ~~eighteen~~ nineteen layers, outperforms AlexNet (figure 10.16)...

- Section 11.1.1 When we change the parameters that determine $\mathbf{f}_1$, *all* of the derivatives in this sequence ~~can change~~ are evaluated at slightly different locations since layers $\mathbf{f}_2, \mathbf{f}_3$, and $\mathbf{f}_4$ are themselves computed from $\mathbf{f}_1$.

- Section 11.2 Rephrased for clarity. A complementary way of thinking about this residual network is that it creates sixteen paths with differing numbers of transformations between input and output ~~of different lengths from input to output~~.

- Section 11.4.1 Hence, it both creates redundancy in the ~~weight parameters~~ weights and biases and

- Section 11.5.1 Rephrasing of fourth and fifth sentences to emphasize that these changes happen between ResNet blocks. New version: The resolution is decreased between adjacent ResNet blocks using convolutions with stride two. Channels are similarly added by either appending zeros to the representation or applying an extra 1×1 convolution.

- Section 11.5.2 Concatenation across the downsampling makes no sense since the representations have different spatial sizes.

- Chapter 11 Notes p202. A second ReLU function was applied after this block was added back to the main representation. This architecture was termed *ResNet v1.*

- Chapter 11 Notes p203. (networks for processing sequences, in which the previous output is fed back as an additional input as we move through the sequence ~~(~~see figure 12.19)

- Problem 11.2 How many paths of each length would there be ~~if~~ with (i) three residual blocks and (ii) five residual blocks? Deduce the rule for K residual blocks.

- Section 12.2 Rephrasing to avoid potential ambiguity. A self-attention block $\mathbf{sa}[\bullet]$ takes N inputs $\mathbf{x}_1, \ldots, \mathbf{x}_N$, each of dimension $D \times 1$, and returns N ~~output vectors of the same size~~, outputs each of which is also of size $D \times 1$.

- Section 12.2.1 This computation scales linearly with the sequence length N, so it needs fewer parameters than a fully connected network relating all DN inputs to all DN ~~outputs. T~~ values. In fact, the value computation can be viewed as a sparse matrix operation with shared parameters that relates these DN quantities (figure 10.2b)

- Figure 12.2 Legend. Each output is a linear combination of the values, with ~~a shared~~ the attention weight a[$\mathbf{x}_m$ ...

- Section 12.3.1 Observant readers will have noticed that the self-attention mechanism ~~discards~~ overlooks important information: the computation ~~is the same regardless~~ does not take into account the order of the inputs $\mathbf{x}_n$.

- Section 12.3.1 Changed word to avoid ambiguity: The input to a self-attention mechanism may be an entire sentence, many sentences, or just a fragment of a sentence, and the absolute position of a word is much less important than the relative position between two ~~inputs~~ words. Figure 12.8. Moved the following sentence from descripton of panel c to the description of panel b to avoid ambiguity: Note that the last character of the first token to be merged cannot be whitespace, which prevents merging across words.

- Section 12.6: i.e., the matrices $\boldsymbol{\Omega}_{vh}, \boldsymbol{\Omega}_{qh}, \boldsymbol{\Omega}_{kh}$ are ~~1024×64~~ 64×1024

- Section 12.6. In the *fine-tuning stage*, the resulting network is adapted to solve a particular task using a smaller body of ~~supervised~~ labelled training data.

- Figure 12.10 legend: Minor rephrasing to improve clarity. As such, the output embeddings are passed through a softmax function, and the multiclass classification loss (section 5.24) is used.→ To this end, the outputs corresponding to the masked tokens are passed through softmax functions, and a multiclass classification loss (section 5.24) is applied to each

- Section 12.7.1 Rephrased to avoid ambiguity: GPT3 ~~constructs~~ is an autoregressive language model.

- Section 12.7.1 Original text is true, but this is a more useful statement given where we are in the text: The autoregressive formulation demonstrates the connection between maximizing the ~~log~~ joint probability of the tokens ~~in the loss function~~ and the next token prediction task

- Section 12.7.1 Slight rephrasing of sentence before equation 12.14. An autoregressive model predicts the conditional distributions $Pr(t_n|t_1, \ldots, t_{n-1})$ of each

token given all the prior tokens, and hence indirectly computes the joint probability $Pr(t_1, t_2, \ldots, t_N)$ of all N tokens:

- Section 12.7.2 Added some constructive redundancy: Ideally, we would pass in the whole sentence and compute all the log probabilities and gradients ~~simultaneously~~ in the same forward pass rather than doing a forward pass for each token in the sentence.
- Section 12.7.2 Slight rewording to improve clarity: To train a decoder, we seek parameters that maximize the log probability of the input text under the autoregressive model (i.e., that maximize the sum of the log conditional probability terms).
- Section 12.7.2: Slightly better: Consequently, after the transformer layers, a single linear layer maps each ~~word~~ output embedding to the size of the vocabulary, followed by a $\textbf{softmax}[\bullet]$ function that converts these values to probabilities.
- Section 12.7.3: The new extended sequence can be fed back into the decoder network ~~that outputs~~ to yield the probability distribution over the next token.
- Section 12.7.3: For example, *beam search* keeps track of multiple possible sentence completions to find the overall most likely sequence of words (which is not necessarily found by greedily choosing the most likely ~~next~~ word at each step).
- Section 12.8 Improved first two sentences slightly: Translation between languages is an example of a *sequence-to-sequence* task. One common approach uses both an encoder (to compute a good representation of the source sentence) and a decoder (to generate the sentence in the target language). This is aptly called an *encoder-decoder* model.
- Figure 12.14 Legend. The flow of computation is the same as in standard self-attention ~~. However~~, but the queries are calculated from the decoder embeddings $\mathbf{X}_{dec}$, and the keys and values from the encoder embeddings $\mathbf{X}_{enc}$. ~~In the context of translation~~ For translation tasks, the encoder contains information about the source language statistics, and the decoder contains information about the target language statistics.
- Section 12.10.1 Third paragraph, sentence phrasing changed for improved clarity: Each pixel's final embedding is averaged, and a linear layer maps these values to activations which are passed through a softmax layer to predict class probabilities.
- Section 12.10.2 Fixed minor inaccuracy Each patch is mapped to ~~a lower dimension~~ an input embedding via a learned linear transformation,
- Section 12.10.3. Reworded paragraph to include references to receptive field: The Vision Transformer differs from convolutional architectures in that it operates on a single scale and has a receptive field that covers the whole image. Several transformer models that process the image at multiple scales have been proposed. Similarly to convolutional networks, these generally start with small high resolution patches and few channels and gradually enlarge the receptive field, decrease the spatial resolution and increase the number of channels.

- Figure 12.18 legend. Added clarification about windows: The transformer network applies self-attention to the patches within each window independently (i.e., patches only attend to other patches in the same window).
- Chapter 12 Notes. The *synthesizer* simplifies this idea by ~~simply~~
- Figure 13.7 Legend. (i) aggregating the neighboring nodes to form a single vector, (ii) applying a linear transformation $\boldsymbol{\Omega}_0$ to the aggregated ~~nodes~~ vector.
- Figure 13.12 The outputs of the dot-product self-attention mechanism in the transformer are also weighted sums of the transformed inputs
- Figure 13.9 Legend. Each node depends on its neighbors in the previous layer. These, in turn, depend on their neighbors in the layer before, so (similarly to convolutional networks) each node has a receptive field (figure 13.9). The ~~size of the~~receptive field size is termed the *k-hop neighborhood*.
- Section 13.8.6 This is very similar to the dot-product self-attention computation in transformers
- Figure 14.1 Legend. Latent variable models define a mapping between an underlying explanatory (latent) variable and the data~~. They~~and may fall into any of the above categories.
- Section 14.3. Slightly improvement in clarity to distinguish random variable from its instantiation. Changed all mentions of $Pr(y_i|\mathbf{x}_i^*)$ to $Pr(y|\mathbf{x}_i^*)$.
- Section 15.2.4: It is a *linear programming problem* in its *primal form*~~.~~
- Section 15.2.6: Changed to stop nasty line wrap-over: One way to achieve this is to clip the discriminator weights to a small range (e.g., ± 0.01)
- Figure 15.9: Removed brackets around image sizes for consistency with text.
- Figure 15.11: Changed caption to reflect the fact that multiple tricks have been used to get good performance here, not just progressive growing. New caption: "Combining methods. GANs can generate realistic images of faces when trained on CELEBA-HQ dataset and more complex, variable objects when trained on LSUN categories."
- Section 15.5.3 Added marginal reference to ℓ_1-norm
- Section 15.6: Figure 15.20 shows examples of manipulating the style and noise vectors ~~are~~ at different scales
- Chapter 15 Notes on Conditional GANS. Images generated by GANs have variously been conditioned on classes (~~e.g.,~~Odena et al., 2017), input text (Reed et al., 2016a; Zhang et al., 2017d), attributes (Yan et al., 2016; Donahue et al., 2018a; Xiao et al., 2018b), bounding boxes and keypoints (Reed et al., 2016b), and images (~~e.g.,~~Isola et al., 2017)~~)~~

- Chapter 15. Notes on Adversarial loss. In many image translation tasks, there is no "generator"; ~~such models~~ these can be considered supervised learning tasks with an adversarial loss that encourages realism.

- Problem 15.2: Write an equation relating the loss L in equation 15.8 to the Jensen-Shannon distance $D_{JS}\left[Pr(\mathbf{x}^*) \,||\, Pr(\mathbf{x})\right]$ in equation 15.9.

- Section 16.3.3: If the function $\mathbf{g}[\bullet, \boldsymbol{\phi}]$ is an elementwise flow, the Jacobian will be lower triangular ~~diagonal~~with the identity matrix in the top-left quadrant and the derivatives of the elementwise transformation in the bottom-right.

- Section 17.1.1 As in equation 17.2, the probability ~~likelihood~~ $Pr(x)$ is given by the marginalization over the latent variable z

- Section 17.4.2 This is termed the reconstruction loss → This measures the reconstruction accuracy. (Technically, it's not a loss since we are maximizing it).

- Chapter 17 Notes. Missing period: Other authors have investigated using a discrete latent space (Van Den Oord et al., 2017; Razavi et al., 2019b; Rolfe, 2017; Vahdat et al., 2018a,b).

- Problem 17.7 Derive the EM algorithm for fitting the 1D mixture of Gaussians ~~algorithm~~ model with N components.

- Equation 18.34 Added brackets to equation:

$$\begin{aligned} L[\boldsymbol{\phi}_{1\ldots T}] &= \sum_{i=1}^{I}\Bigg(-\log\Big[\text{Norm}_{\mathbf{x}_i}\left[\mathbf{f}_1[\mathbf{z}_{i1}, \boldsymbol{\phi}_1], \sigma_1^2\mathbf{I}\right]\Big] \\ &\quad + \sum_{t=2}^{T}\frac{1}{2\sigma_t^2}\left\|\left(\frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_{it} - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon}_{it}\right) - \mathbf{f}_t[\mathbf{z}_{it}, \boldsymbol{\phi}_t]\right\|^2\Bigg). \end{aligned}$$

- Under Equation 18.36: Added missing equation number in equation after 18.36.

- Figure 18.10 Caption: ...including the denoising diffusion implicit ~~(DDIM)~~ model (DDIM)

- Chapter 18 notes. One of these models is the denoising diffusion implicit model (DDIM), in which the updates are not stochastic (figure 10.8~~bc~~). This model is amenable to taking larger steps (figure 10.8~~de~~) without inducing large errors.

- Problem 18.1 Show that if $\text{Cov}[\mathbf{x}_{t-1}] = \mathbf{I}$ and we use the update:

$$\mathbf{x}_t = \sqrt{1-\beta_t}\cdot\mathbf{x}_{t-1} + \sqrt{\beta_t}\cdot\boldsymbol{\epsilon}_t,$$

  then $\text{Cov}[\mathbf{x}_t] = \mathbf{I}$, so the variance stays the same.

- Chapter 19 introduction: In finance, an RL algorithm might control a virtual trader (the agent) who buys or sells assets (the actions) on a ~~trading platform~~ financial exchange (the environment) to maximize profit (the reward).
- Section 19.1.5 Minor reprhrasing: The environment then ~~assigns~~ advances to the next state according to $Pr(s_{t+1}|s_t, a_t)$ and issues a reward according to $Pr(r_{t+1}|s_t, a_t)$.
- Figure 19.3 legend. Slight rephrasing to improve clarity: The agent (penguin) can perform one of a set of actions in each state. ~~The action influences both the probability of moving to the successor state and the probability of receiving rewards.~~ b) Here, the four actions correspond to moving up, right, down, and left. c) For any state (here, state 6), the action changes the probability of moving to the next state. The penguin moves in the intended direction with 50% probability, but the ice is slippery, so it may slide to one of the other adjacent positions with equal probability. Accordingly, in panel (a), the action taken (gray arrows) doesn't always line up with the trajectory (orange line). ~~Here,~~ In general, the action can also influence the probability of receiving rewards, but in this example the action does not affect the reward...
- Figure 19.4. Slight rephrasing for clarity: Here, the penguin is in state three and can only see ~~the region in the dashed box~~ tiles in the vicinity (dashed box).
- Section 19.1.3 Rephrase for clarity: An MDP produces a sequence $(s_1, a_1, r_2), (s_2, a_2, r_3), (s_3, a_3, r_4) \ldots$ of states s_t, actions a_t, and rewards r_{t+1} which are received at the subsequent time-step (figure 19.3).
- Section 19.3. Added footnote for "Model-based methods": The term model refers here to the MDP and not a machine learning model.
- Section 19.3 Reworded to avoid ambiguity. Conversely, model-free methods ~~eschew a model of the MDP and fall into two classes~~ assume that the transition matrix and reward structure of the underlying MDP are unknown. These methods fall into two families:
- Section 19.3.2. Not wrong, but clearer with these tweaks: Each starts with a given state and action and thereafter follows the current policy, producing a series of actions, states, and ~~returns~~ rewards (figure 2.11a). The action value for a given state-action pair under the current policy is estimated as the average of the empirical returns (i.e., cumulative sums of time-discounted rewards) that follow each time this pair occurs (figure 2.11b).
- Figure 19.10b: Tweaked four of the values as had rounded oddly in computation.
- Figure 19.11c: Policy in cell 12 should not be updated (arrow should still point upwards). We have not performed the "down arrow" action, so we can't change this.
- Section 19.3.3 Changed tense for internal consistency: Monte Carlo methods sample~~d~~ the MDP to acquire information.

- Figure 19.14: Input should be tagged as having size $84 \times 84 \times 4$.
- Section 19.4.2: ...leads to a systematic bias in the estimated action values $q[s_t, a_t]$.
- Equation 19.30: Changed for compatibility with earlier definition of reward:

$$r[\boldsymbol{\tau}_i] = \sum_{t=1}^{T} r_{i,t+1} = \sum_{k=1}^{t} r_{i,k+1} + \sum_{k=t}^{T} r_{i,k+1},$$

- Equation 19.31: Changed for compatibility with earlier definition of reward:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \cdot \frac{1}{I} \sum_{i=1}^{I} \sum_{t=1}^{T} \frac{\partial \log\left[\pi[a_{it}|\mathbf{s}_{it}, \boldsymbol{\theta}]\right]}{\partial \boldsymbol{\theta}} \sum_{k=t}^{T} r_{i,k+1}.$$

- Equation 19.32: Changed for compatibility with earlier definition of reward:

$$r[\boldsymbol{\tau}_{it}] = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_{i,k+1},$$

- Section 19.5.3 One way to reduce this variance is to subtract ~~the trajectory returns $r[\boldsymbol{\tau}]$ from a baseline b~~ a baseline b from the trajectory returns $r[\boldsymbol{\tau}]$:
- Equation 19.39: Changed for compatibility with earlier definition of reward:

$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} \sum_{t=1}^{T} \left(v[\mathbf{s}_{it}, \boldsymbol{\phi}] - \sum_{j=t}^{T} r_{i,j+1} \right)^2.$$

- Equation 19.40: Changed for compatibility with earlier definition of reward:

$$r[\boldsymbol{\tau}_{it}] \approx r_{i,t+1} + \gamma \cdot v[\mathbf{s}_{i,t+1}, \boldsymbol{\phi}].$$

- Equation 19.41: Changed for compatibility with earlier definition of reward:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \cdot \frac{1}{I} \sum_{i=1}^{I} \sum_{t=1}^{T} \frac{\partial \log\left[Pr(a_{it}|\mathbf{s}_{it}, \boldsymbol{\theta})\right]}{\partial \boldsymbol{\theta}} \Big(r_{i,t+1} + \gamma \cdot v[\mathbf{s}_{i,t+1}, \boldsymbol{\phi}] - v[\mathbf{s}_{i,t}, \boldsymbol{\phi}] \Big).$$

- Equation 19.42: Changed for compatibility with earlier definition of reward:

$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} \sum_{t=1}^{T} \left(r_{i,t+1} + \gamma \cdot v[\mathbf{s}_{i,t+1}, \boldsymbol{\phi}] - v[\mathbf{s}_{i,t}, \boldsymbol{\phi}] \right)^2.$$

- Figure 19.18 Legend. One trajectory through an ~~MDP~~ MRP

- Problem 19.1 Figure 19.18 shows a single trajectory through ~~the example MDP~~ an example MRP
- Problem 19.3 Added clarification that discount value is 0.9
- Section 20.12 if each of the 40 inputs of the MNIST-1D data were quantized into 10 possible values, there would be 10^{40} possible inputs, which is a factor of ~~10^{35}~~ 10^{36} more than the number of training examples.
- Section 20.12 A fully connected network for MNIST-1D with two hidden layers of width 400 can create mappings with up to 10^{42} linear regions. That's roughly ~~10^{37}~~ 10^{38} regions per training example, so very few of these regions contain data at any stage during training;
- Figure 21.1 Caption Structural description of the value alignment problem. ~~a)~~ Problems arise from a) misaligned objectives (e.g., bias) or b) informational asymmetries between a (human) principal and an (artificial) agent (e.g., lack of explainability).
- Section B.3.2 Minor clarification: For a vector $\mathbf{z}$, the ℓ_p norm is defined as:

$$||\mathbf{z}||_p = \left(\sum_{d=1}^{D} |z_d|^p\right)^{1/p}, \tag{1.10}$$

  for real-valued $p > 1$.
- Section C.2.1 Added missing equation numbers
- Section C.3.3 Added missing equation number
- Figure C.4 When the covariance matrix is a multiple of the ~~diagonal~~ identity matrix, the isocontours are circles, and we refer to this as spherical covariance.
- Section C.4.2. Consider a joint distribution $Pr(x, y, z)$ over three variables, x, y, and z, which in this particular case factors as:

Second and third printings (Mar 2024)

These are things that are wrong and need to be fixed, but that will probably not affect your understanding (e.g., math symbols that are in bold but should not be).

- Section 1.1: ...and what is meant by "training" a model.
- Figure 1.13: ~~Adapted from Pablok (2017).~~

- Figure 2.3 legend: Each combination of parameters $\boldsymbol{\phi} = [\phi_0,\phi_1]^T$.
- Section 2.3: 1D linear regression has the obvious drawback
- Figure 3.5 legend: The universal approximation theorem proves that, with enough hidden units, there exists a shallow neural network that can describe any given continuous function defined on a compact subset of $\mathbb{R}^{D_i}$ to arbitrary precision.
- Notes page 38 Most of these are attempts to avoid the dying ReLU problem while limiting the gradient for negative values.
- Figure 4.1 legend: The first network maps inputs $x \in [-1, 1]$ to outputs $y \in [-1, 1]$ using a function comprising three linear regions that are chosen so that they alternate the sign of their slope (fourth linear region is outside range of graph).
- Figure 4.2: Colors changed to avoid ambiguity
- Equation 4.13 is missing a prime sign:

$$\begin{aligned}\mathbf{h} &= \mathbf{a}\left[\boldsymbol{\theta}_0 + \boldsymbol{\theta} x\right] \\ \mathbf{h}' &= \mathbf{a}\left[\boldsymbol{\psi}_0 + \boldsymbol{\Psi}\mathbf{h}\right] \\ y' &= \phi_0' + \boldsymbol{\phi}'\mathbf{h}',\end{aligned}$$

- Equation 4.14: ϕ_0' should not be bold.

$$y = \phi_0' + \boldsymbol{\phi}'\mathbf{h}'$$

- Equation 4.17 is not technically wrong, but the product is unnecessary and it's unclear if the last term should be included in it (no). Better written as:

$$N_r = \left(\frac{D}{D_i} + 1\right)^{D_i(K-1)} \cdot \sum_{j=0}^{D_i} \binom{D}{j}.$$

- Equation 5.10. Second line is disambiguated by adding brackets:

$$\begin{aligned}\hat{\boldsymbol{\phi}} &= \underset{\boldsymbol{\phi}}{\text{argmin}}\left[-\sum_{i=1}^{I}\log\left[\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{(y_i - \text{f}[\mathbf{x}_i, \boldsymbol{\phi}])^2}{2\sigma^2}\right]\right]\right] \\ &= \underset{\boldsymbol{\phi}}{\text{argmin}}\left[-\sum_{i=1}^{I}\left(\log\left[\frac{1}{\sqrt{2\pi\sigma^2}}\right] - \frac{(y_i - \text{f}[\mathbf{x}_i, \boldsymbol{\phi}])^2}{2\sigma^2}\right)\right] \\ &= \underset{\boldsymbol{\phi}}{\text{argmin}}\left[-\sum_{i=1}^{I} -\frac{(y_i - \text{f}[\mathbf{x}_i, \boldsymbol{\phi}])^2}{2\sigma^2}\right] \\ &= \underset{\boldsymbol{\phi}}{\text{argmin}}\left[\sum_{i=1}^{I}(y_i - \text{f}[\mathbf{x}_i, \boldsymbol{\phi}])^2\right],\end{aligned}$$

- Equation 5.12. More properly written as:

$$\hat{y} = \underset{y}{\text{argmax}} \left[Pr(y|\text{f}[\mathbf{x}, \hat{\boldsymbol{\phi}}, \sigma^2]) \right]. \tag{1.11}$$

  although the value of σ^2 does not actually matter or change the position of the maximum.

- Equation 5.15. Disambiguated by adding brackets:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\text{argmin}} \left[-\sum_{i=1}^{I} \left(\log \left[\frac{1}{\sqrt{2\pi \text{f}_2[\mathbf{x}_i, \boldsymbol{\phi}]^2}} \right] - \frac{(y_i - \text{f}_1[\mathbf{x}_i, \boldsymbol{\phi}])^2}{2\text{f}_2[\mathbf{x}_i, \boldsymbol{\phi}]^2} \right) \right].$$

- Section 5.5 The likelihood that input $\mathbf{x}$ has label $y = k$ (figure 5.10) is hence:

- Section 5.6 Removed i index from this paragraph for consistency. Independence implies that we treat the probability $Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}])$ as a product of univariate terms for each element $y_d \in \mathbf{y}$:

$$Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]) = \prod_{d} Pr(y_d|\mathbf{f}_d[\mathbf{x}, \boldsymbol{\phi}]),$$

  where $\mathbf{f}_d[\mathbf{x}, \boldsymbol{\phi}]$ is the d^{th} set of network outputs, which describe the parameters of the distribution over y_d. For example, to predict multiple continuous variables $y_d \in \mathbb{R}$, we use a normal distribution for each y_d, and the network outputs $\text{f}_d[\mathbf{x}, \boldsymbol{\phi}]$ predict the means of these distributions. To predict multiple discrete variables $y_d \in \{1, 2, \ldots, K\}$, we use a categorical distribution for each y_d. Here, each set of network outputs $\mathbf{f}_d[\mathbf{x}, \boldsymbol{\phi}]$ predicts the K values that contribute to the categorical distribution for y_d.

- Problem 5.8. Construct a loss function for making multivariate predictions $\mathbf{y} \in \mathbb{R}^{D_i}$ based on independent normal distributions...

- Notes page 94. However, this is strange since SGD is a special case of Adam (when $\beta = \gamma = 0$)

- Section 7.3. The final derivatives from the term $f_0 = \beta_0 + \omega_0 \cdot x_i$ are:

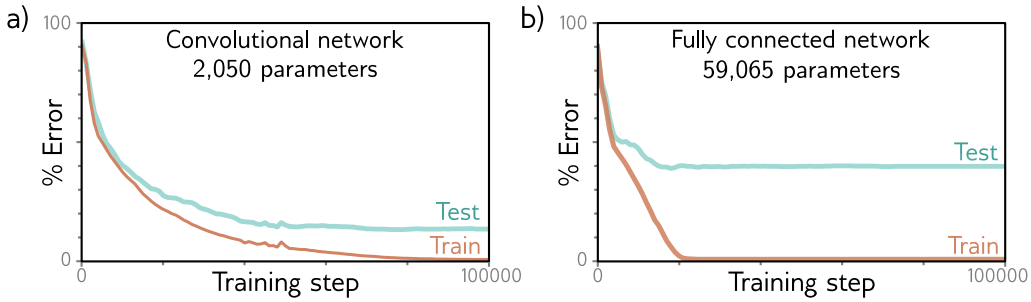
- Section 7.4. Similarly, the derivative for the weights matrix $\boldsymbol{\Omega}_k$, is given by

- Section 7.5.1 and the second moment $\mathbb{E}[h_j^2]$ will be half the variance σ_f^2

- Figure 7.8. Not wrong, but changed to "nn.init.kaiming_normal_(layer_in.weight)" for compatability with text and to avoid deprecated warning.

- Section 8.3.3 (i.e., with four hidden units and four linear regions in the range of the data) + minor changes in text to accommodate extra words

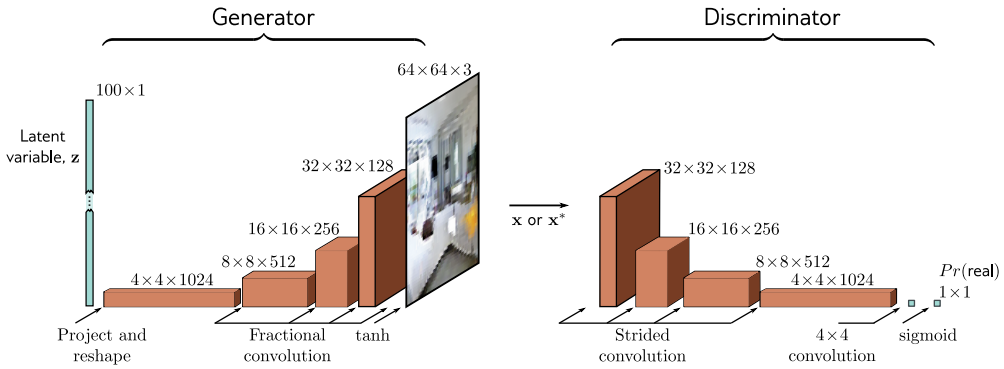
- Figure 8.9 number of hidden units / linear regions in range of data



Corrected version of figure 10.8

- Section 8.4.1 When the number of parameters is very close to the number of training data examples (figure 8.11b)
- Figure 9.5 legend: Effect of learning rate (LR) and batch size for 4000 training and 4000 test examples from MNIST-1D (see figure 8.1) for a neural network with two hidden layers. a) Performance is better for large learning rates than for intermediate or small ones. In each case, the number of iterations is 6000/LR, so each solution has the opportunity to move the same distance.
- Figure 10.3. The dilation rates are wrong by one, so should be 1,1,1, and 2 in panels a,b,c,d, respectively.
- Section 10.2.1 Not wrong, but could be disambiguated: The size of the region over which inputs are combined is termed the *kernel size*.
- Section 10.2.3 The number of zeros we intersperse between the weights determines the dilation rate.
- Section 10.2.4 With kernel size three, stride one, and dilation rate one.
- Section 10.2.7 The convolutional network has 2,050 parameters, and the fully connected network has 59,065 parameters.
- Figure 10.8 Number of parameters also wrong in figure 10.8 (correct version in this document). Recalculated curve is slightly different.
- Section 10.5.3 The first part of the network is a smaller version of VGG (figure 10.17) that contains thirteen rather than sixteen convolutional layers.
- Section 10.6 The weights and the bias are the same at every spatial position, so there are far fewer parameters than in a fully connected network, and the number of parameters doesn't increase with the input image size.
- Problem 10.1 Show that the operation in equation 10.3 is equivariant with respect to translation.
- Problem 10.2 Equation 10.3 defines 1D convolution with a kernel size of three, stride of one, and dilation one.

- Problem 10.3 Write out the equation for the 1D dilated convolution with a kernel size of three and a dilation rate of two.
- Problem 10.4 Write out the equation for a 1D convolution with kernel size of seven, a dilation rate of three, and a stride of three.
- Problem 10.9 A network consists of three 1D convolutional layers. At each layer, a zero-padded convolution with kernel size three, stride one, and dilation one is applied.
- Problem 10.10 A network consists of three 1D convolutional layers. At each layer, a zero-padded convolution with kernel size seven, stride one, and dilation one is applied.
- Problem 10.11 Consider a convolutional network with 1D input $\mathbf{x}$. The first hidden layer $\mathbf{H}_1$ is computed using a convolution with kernel size five, stride two, and a dilation rate of one. The second hidden layer $\mathbf{H}_2$ is computed using a convolution with kernel size three, stride one, and a dilation rate of one. The third hidden layer $\mathbf{H}_3$ is computed using a convolution with kernel size five, stride one, and a dilation rate of two. What are the receptive field sizes at each hidden layer?
- Legend to figure 11.15. Computational graph for batch normalization (see problem 11.5).
- Section 12.2: Not a mistake, but this is clearer: where $\boldsymbol{\beta}_v \in \mathbb{R}^D \times 1$ and $\boldsymbol{\Omega}_v \in \mathbb{R}^{D \times D}$ represent biases and weights, respectively.
- Section 12.3.3 to make self-attention work well
- Section 12.4 Title changed to Transformer layers
- Section 12.4 a larger transformer mechanism
- Section 12.4 a series of these transformer layers ...
- Section 12.5 The previous section described the transformer layer... a series of transformer layers...
- Figure 12.8 legend: The transformer → Transformer layer...The transformer layer consists
- Figure 12.8 has some minor mistakes in the calculation. The corrected version is shown at the end of this document.
- Figure 12.8 legend. At each iteration, the sub-word tokenizer looks for the most commonly occurring adjacent pair of tokens
- Section 12.5.3 a series of K transformer layers
- Section 12.6 through 24 transformer layers



Corrected version of figure 15.3

- Section 12.6 in the fully connected networks ~~in the transformer~~ is 4096
- Figure 12.10 legend: a series of transformer layers
- Section 12.7.2 the transformer layers use masked...
- Figure 12.12 are passed through a series of transformer layers... and those of tokens earlier
- Section 12.7.4 There are 96 transformer layers
- Section 12.7 comprises a series of transformer layers
- Section 12.8 Originally, these
- Section 12.8 a series of transformer layers... a series of transformer layers
- Section 13.5.1 Given I training graphs $\{\mathbf{X}_i, \mathbf{A}_i\}$ and their labels y_i, the parameters $\boldsymbol{\Phi} = \{\boldsymbol{\beta}_k, \boldsymbol{\Omega}_k\}_{k=0}^{K}$ can be learned using SGD...
- Figure 15.3 legend: At the end is a tanh function that maps the...
- Figure 15.3 arctan → tanh. Corrected version nearby in this document.
- Section 15.1.3: At the final layer, the 64×64×3 signal is passed through a tanh function to generate an image $\mathbf{x}^*$
- Equation 15.6. Minor problems with brackets in this equation. Should be:

$$\begin{aligned} L[\boldsymbol{\phi}] &= \frac{1}{J}\sum_{j=1}^{J}\Big(\log\Big[1-\text{sig}[\text{f}[\mathbf{x}_j^*,\boldsymbol{\phi}]]\Big]\Big) + \frac{1}{I}\sum_{i=1}^{I}\Big(\log\Big[\text{sig}[\text{f}[\mathbf{x}_i,\boldsymbol{\phi}]]\Big]\Big) \\ &\approx \mathbb{E}_{\mathbf{x}^*}\Big[\log\Big[1-\text{sig}[\text{f}[\mathbf{x}^*,\boldsymbol{\phi}]]\Big]\Big] + \mathbb{E}_{\mathbf{x}}\Big[\log\Big[\text{sig}[\text{f}[\mathbf{x},\boldsymbol{\phi}]]\Big]\Big] \\ &= \int Pr(\mathbf{x}^*)\log\Big[1-\text{sig}[\text{f}[\mathbf{x}^*,\boldsymbol{\phi}]]\Big]d\mathbf{x}^* + \int Pr(\mathbf{x})\log\Big[\text{sig}[\text{f}[\mathbf{x},\boldsymbol{\phi}]]\Big]d\mathbf{x}. \end{aligned}$$

- Equation 16.2 (last line). For some reason, this didn't print properly, although it looks fine in my original pdf. Should be:

$$\begin{aligned}\hat{\boldsymbol{\phi}} &= \underset{\boldsymbol{\phi}}{\text{argmax}}\left[\prod_{i=1}^{I} Pr(x_i|\boldsymbol{\phi})\right] \\ &= \underset{\boldsymbol{\phi}}{\text{argmin}}\left[\sum_{i=1}^{I} -\log\Big[Pr(x_i|\boldsymbol{\phi})\Big]\right] \\ &= \underset{\boldsymbol{\phi}}{\text{argmin}}\left[\sum_{i=1}^{I} \log\left[\left|\frac{\partial \mathbf{f}[z_i,\boldsymbol{\phi}]}{\partial z_i}\right|\right] - \log\Big[Pr(z_i)\Big]\right],\end{aligned}$$

- Equation 16.25. $\boldsymbol{\phi}$ should change to $\hat{\boldsymbol{\phi}}$ on left hand side. Correct version is:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\text{argmin}}\left[\text{KL}\left[\sum_{i=1}^{I}\delta\big[\mathbf{x}-\mathbf{f}[\mathbf{z}_i,\boldsymbol{\phi}]\big]\middle\|q(\mathbf{x})\right]\right].$$

- Equation 16.26. $\boldsymbol{\phi}$ should change to $\hat{\boldsymbol{\phi}}$ on left hand side. Correct version is:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\text{argmin}}\left[\text{KL}\left[\frac{1}{I}\sum_{i=1}^{I}\delta[\mathbf{x}-\mathbf{x}_i]\middle\|Pr(\mathbf{x}_i,\boldsymbol{\phi})\right]\right].$$

- Equation 18.24 has a minor formatting mistake. Better written as:

$$\begin{aligned}&\log\left[\frac{Pr(\mathbf{x},\mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T})}{q(\mathbf{z}_{1\ldots T}|\mathbf{x})}\right] \\ &\quad= \log\left[\frac{Pr(\mathbf{x}|\mathbf{z}_1,\boldsymbol{\phi}_1)}{q(\mathbf{z}_1|\mathbf{x})}\right] + \log\left[\frac{\prod_{t=2}^{T} Pr(\mathbf{z}_{t-1}|\mathbf{z}_t,\boldsymbol{\phi}_t)\cdot q(\mathbf{z}_{t-1}|\mathbf{x})}{\prod_{t=2}^{T} q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x})\cdot q(\mathbf{z}_t|\mathbf{x})}\right] + \log\Big[Pr(\mathbf{z}_T)\Big] \\ &\quad= \log\left[Pr(\mathbf{x}|\mathbf{z}_1,\boldsymbol{\phi}_1)\right] + \log\left[\frac{\prod_{t=2}^{T} Pr(\mathbf{z}_{t-1}|\mathbf{z}_t,\boldsymbol{\phi}_t)}{\prod_{t=2}^{T} q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x})}\right] + \log\left[\frac{Pr(\mathbf{z}_T)}{q(\mathbf{z}_T|\mathbf{x})}\right] \\ &\quad\approx \log\left[Pr(\mathbf{x}|\mathbf{z}_1,\boldsymbol{\phi}_1)\right] + \sum_{t=2}^{T}\log\left[\frac{Pr(\mathbf{z}_{t-1}|\mathbf{z}_t,\boldsymbol{\phi}_t)}{q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x})}\right],\end{aligned}$$

- Equation 18.34 missing indices on noise term:

$$\begin{aligned}L[\boldsymbol{\phi}_{1\ldots T}] &= \sum_{i=1}^{I} -\log\Big[\text{Norm}_{\mathbf{x}_i}\left[\mathbf{f}_1[\mathbf{z}_{i1},\boldsymbol{\phi}_1],\sigma_1^2\mathbf{I}\right]\Big] \\ &\quad+\sum_{t=2}^{T}\frac{1}{2\sigma_t^2}\left\|\left(\frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_{it}-\frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon}_{it}\right)-\mathbf{f}_t[\mathbf{z}_{it},\boldsymbol{\phi}_t]\right\|^2.\end{aligned} \tag{1.12}$$

- Section 20.2.2 Another possible explanation for the ease with which models are trained is that some regularization methods like L2 regularization (weight decay) make the loss surface flatter and more convex.
- Section 20.2.4 For example, Du et al. (2019a) show that residual networks converge to zero training loss when the width of the network D (i.e., the number of hidden units) is $\Omega[I^4K^2]$ where I is the amount of training data, and K is the depth of the network.
- Section 21.7 the Conference on AI, Ethics, and Society
- Appendix A. The notation $\{0, 1, 2, \ldots\}$ denotes the set of non-negative integers.
- Appendix A ...*big-O* notation, which represents an upper bound...
- Appendix A. f$[n] < c\cdot$g$[n]$ for all $n > n_0$
- Equation B. 18

$$\begin{aligned} y_1 &= \phi_{10} + \phi_{11}z_1 + \phi_{12}z_2 + \phi_{13}z_3 \\ y_2 &= \phi_{20} + \phi_{21}z_1 + \phi_{22}z_2 + \phi_{23}z_3 \\ y_3 &= \phi_{30} + \phi_{31}z_1 + \phi_{32}z_2 + \phi_{33}z_3. \end{aligned} \tag{1.13}$$

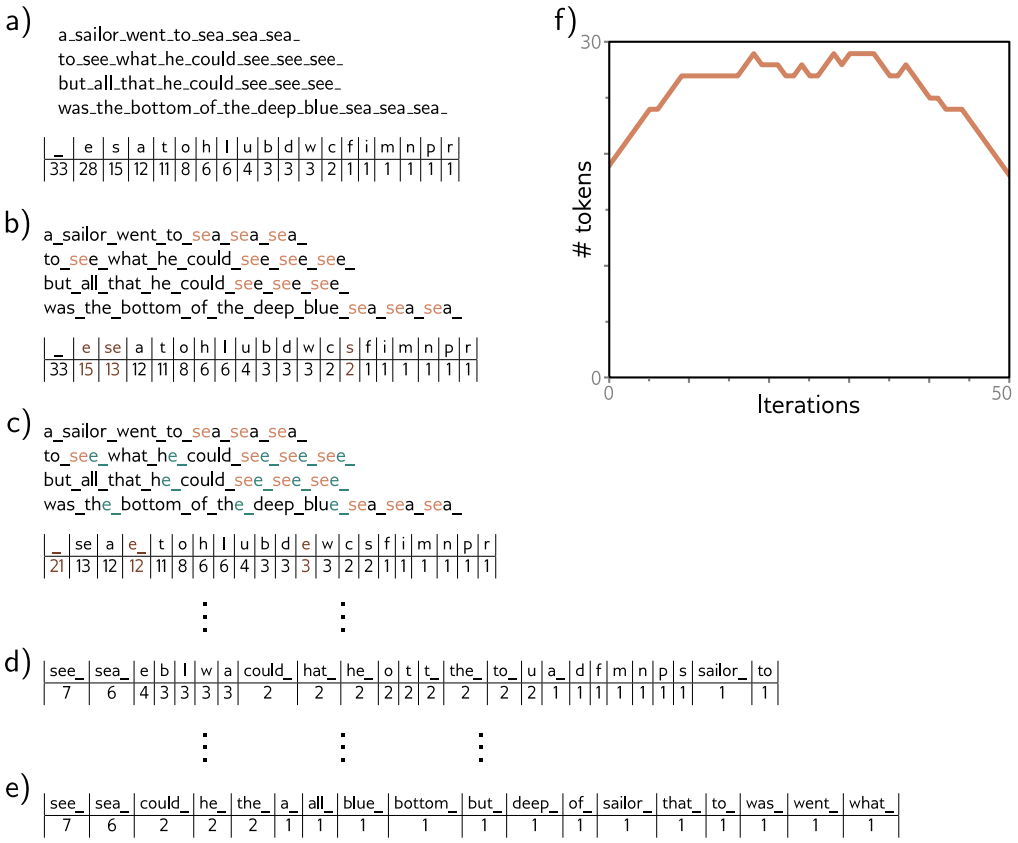
- Appendix C.5.4 Accent in wrong place: The Fréchet and Wasserstein distances...
- Equation C.32.

$$D_{KL}\Big[\text{Norm}[\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1]\Big|\Big|\text{Norm}[\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2]\Big] = \frac{1}{2}\left(\log\left[\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|}\right] - D + \text{tr}\left[\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\right] + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)\right).$$

First printing (Dec 2023)

These are things that are wrong and need to be fixed, but that will probably not affect your understanding (e.g., math symbols that are in bold but should not be).

- Section 1.1.1: In contrast, the model in figure 1.2b takes the chemical structure of a molecule as an input and predicts both the freezing and boiling points.
- Figure 1.2 legend: This *multivariate regression* model takes the structure of a chemical molecule and predicts its freezing and boiling points.



Corrected version of figure 12.8

- Section 1.1.2: Finally, consider the input for the model that predicts the freezing and boiling points of the molecule. A molecule may contain varying numbers of atoms that can
- Multiple places in Chapters 2-9. Loss functions $L[\boldsymbol{\phi}]$ sometimes written as $\mathrm{L}[\boldsymbol{\phi}]$. Have now all been converted to italic for consistency.
- Section 3.1.1 The slope of each linear region is determined by ~~(i)~~ the original slopes $\theta_{\bullet 1}$ of the active inputs for this region and ~~(ii)~~ the weights $\phi_{\bullet}$ that were subsequently applied.
- Section 4.5.5 Added missing definition of over-parameterization and other minor changes for typesetting consistence: It may be that over-parameterized deep models (i.e., those with more parameters than training examples) have a large family...
- Chapter 4 Notes, page 52. Montúfar
- Problem 4.7: Choose values for the parameters $\boldsymbol{\phi} = \{\phi_0, \phi_1, \phi_2, \phi_3, \theta_{10}, \theta_{11}, \theta_{20}, \theta_{21}, \theta_{30}, \theta_{31}\}$ for the shallow neural network in equation 3.1 (with ReLU activation functions)

that will define an identity function over a finite range $x \in [a, b]$.

- Problem 4.11: How many parameters does each network have? How many linear regions can each network make (see equation 4.17)?
- Problem 5.1 Reworded to be more precise with limits: Show that the logistic sigmoid function $\text{sig}[z]$ becomes 0 as $z \to -\infty$, is 0.5 when $z = 0$, and becomes 1 when $z \to \infty$.
- Problem 5.3: The term $\text{Bessel}_0[\kappa]$ is a modified Bessel function of the first kind of order 0.
- Problem 5.8: Construct a loss function for making multivariate predictions $\mathbf{y} \in \mathbb{R}^{D_o}$ based on...
- Figure 7.4 legend: We work backward from the end of the function computing the derivatives $\partial \ell_i / \partial f_k$ and $\partial \ell_i / \partial h_k$ of the loss with respect to the intermediate quantities.
- Section 7.3 Finally, we consider how the loss ℓ_i changes when we change the parameters $\{\beta_k\}$ and $\{\omega_k\}$.
- Figure 7.5 legend: Finally, we compute the derivatives $\partial \ell_i / \partial \beta_k$ and $\partial \ell_i / \partial \omega_k$
- Problem 7.13 For the same function as in problem 7.12, compute the derivative...
- Equation 8.2:

$$\begin{aligned} L[x] &= \big(\text{f}[x, \boldsymbol{\phi}] - y[x]\big)^2 \\ &= \Big(\big(\text{f}[x, \boldsymbol{\phi}] - \mu[x]\big) + \big(\mu[x] - y[x]\big)\Big)^2 \\ &= \big(\text{f}[x, \boldsymbol{\phi}] - \mu[x]\big)^2 + 2\big(\text{f}[x, \boldsymbol{\phi}] - \mu[x]\big)\big(\mu[x] - y[x]\big) + \big(\mu[x] - y[x]\big)^2, \end{aligned} \tag{1.14}$$

- Section 8.4.1 There would be 10^{40} bins in total, constrained by only 10^4 examples.
- Equation 9.6 LHS should be total derivative, not partial derivative:

$$\frac{d\boldsymbol{\phi}}{dt} = -\frac{\partial L}{\partial \boldsymbol{\phi}}. \tag{1.15}$$

- Figure 9.4 Added to legend for panel (a) Blue point represents global minimum. Added to legend for panel (d) Blue point represents global minimum which may now be in a different place from panel (a).
- Figure 9.9 legend Here we are using full-batch gradient descent, and the model (from figure 8.4) fits the data as well as possible, so further training won't remove the kink
- Figure 9.9 legend Consider what happens if we remove the eighth hidden unit

- Figure 9.11 legend: a-c) Two sets of parameters (cyan and gray curves) sampled from the posterior
- Figure 9.11 legend When the prior variance σ_ϕ^2 is small
- Equation 9.15 Should be total derivative not partial:

$$\frac{d\boldsymbol{\phi}}{dt} = \mathbf{g}[\boldsymbol{\phi}]. \tag{1.16}$$

- Equation 9.16 Should be total derivative not partial:

$$\frac{d\boldsymbol{\phi}}{dt} \approx \mathbf{g}[\boldsymbol{\phi}] + \alpha\mathbf{g}_1[\boldsymbol{\phi}] + \ldots, \tag{1.17}$$

- Equation 9.17 Should be total derivative not partial:

$$\begin{aligned}
\boldsymbol{\phi}[\alpha] &\approx \boldsymbol{\phi} + \alpha\frac{d\boldsymbol{\phi}}{dt} + \frac{\alpha^2}{2}\frac{d^2\boldsymbol{\phi}}{dt^2}\Big|_{\boldsymbol{\phi}=\boldsymbol{\phi}_0} \\
&\approx \boldsymbol{\phi} + \alpha\left(\mathbf{g}[\boldsymbol{\phi}] + \alpha\mathbf{g}_1[\boldsymbol{\phi}]\right) + \frac{\alpha^2}{2}\left(\frac{\partial\mathbf{g}[\boldsymbol{\phi}]}{\partial\boldsymbol{\phi}}\frac{d\boldsymbol{\phi}}{dt} + \alpha\frac{\partial\mathbf{g}_1[\boldsymbol{\phi}]}{\partial\boldsymbol{\phi}}\frac{d\boldsymbol{\phi}}{dt}\right)\Big|_{\boldsymbol{\phi}=\boldsymbol{\phi}_0} \\
&= \boldsymbol{\phi} + \alpha\left(\mathbf{g}[\boldsymbol{\phi}] + \alpha\mathbf{g}_1[\boldsymbol{\phi}]\right) + \frac{\alpha^2}{2}\left(\frac{\partial\mathbf{g}[\boldsymbol{\phi}]}{\partial\boldsymbol{\phi}}\mathbf{g}[\boldsymbol{\phi}] + \alpha\frac{\partial\mathbf{g}_1[\boldsymbol{\phi}]}{\partial\boldsymbol{\phi}}\mathbf{g}[\boldsymbol{\phi}]\right)\Big|_{\boldsymbol{\phi}=\boldsymbol{\phi}_0} \\
&\approx \boldsymbol{\phi} + \alpha\mathbf{g}[\boldsymbol{\phi}] + \alpha^2\left(\mathbf{g}_1[\boldsymbol{\phi}] + \frac{1}{2}\frac{\partial\mathbf{g}[\boldsymbol{\phi}]}{\partial\boldsymbol{\phi}}\mathbf{g}[\boldsymbol{\phi}]\right)\Big|_{\boldsymbol{\phi}=\boldsymbol{\phi}_0},
\end{aligned} \tag{1.18}$$

- Equation 9.19 Should be total derivative not partial:

$$\begin{aligned}
\frac{d\boldsymbol{\phi}}{dt} &\approx \mathbf{g}[\boldsymbol{\phi}] + \alpha\mathbf{g}_1[\boldsymbol{\phi}] \\
&= -\frac{\partial L}{\partial\boldsymbol{\phi}} - \frac{\alpha}{2}\left(\frac{\partial^2 L}{\partial\boldsymbol{\phi}^2}\right)\frac{\partial L}{\partial\boldsymbol{\phi}}.
\end{aligned} \tag{1.19}$$

- Page 156 Notes: Wrong marginal reference — Appendix B.3.7 Spectral Norm
- Figure 10.3 legend – In dilated or atrous convolution (from the French "à trous" – with holes), we intersperse zeros in the weight vector...
- Section 10.5.1 A final max-pooling layer yields a 6×6 representation with 256 channels which is resized into a vector of length 9, 216 and passed through three fully connected layers containing 4096, 4096, and 1000 hidden units, respectively.
- Section 10.5.1 The complete network contains ∼60 million parameters, most of which are in the fully connected layers ~~and the end of the network~~.

- Problem 10.4 Write out the equation for a 1D convolution with kernel size of seven, a dilation rate of three, and a stride of three. You may assume that the input is padded with zeros at positions x_{-2}, x_{-1} and x_0.

- Equation 11.3: Terms should be reordered to be consistent with definition of vector derivative in appendix:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{f}_1} = \frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1}\frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_2}\frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_3}. \tag{1.20}$$

- Equation 11.6: Terms should be reordered to be consistent with definition of vector derivative in appendix:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{f}_1} = \mathbf{I} + \frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1} + \left(\frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_1} + \frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1}\frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_2}\right) + \left(\frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_1} + \frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1}\frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_2} + \frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_1}\frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_3} + \frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1}\frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_2}\frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_3}\right), \tag{1.21}$$

- Equation 11.10:

$$\begin{aligned} f_1 &= \mathbb{E}[z_i] & f_5 &= \sqrt{f_4 + \epsilon} \\ f_{2i} &= z_i - f_1 & f_6 &= 1/f_5 \\ f_{3i} &= f_{2i}^2 & f_{7i} &= f_{2i} \times f_6 \\ f_4 &= \mathbb{E}[f_{3i}] & z_i' &= f_{7i} \times \gamma + \delta, \end{aligned} \tag{1.22}$$

- Figure 12.10 legend: A small fraction of the input tokens are randomly replaced with...

- Section 12.2.1 Not technically wrong, but for consistency with previous and following sections: Equation 12.2 shows that the same weights $\boldsymbol{\Omega}_v \in \mathbb{R}^{D\times D}$ and biases $\boldsymbol{\beta}_v \in \mathbb{R}^D$ are applied to each input $\mathbf{x}_\bullet \in \mathbb{R}^D$...

- Section 12.2.1 Not technically, wrong, but for consistency with previous and following sections: It follows that the number of attention weights has a quadratic dependence on the sequence length N, but is independent of the length D of each input ~~$\mathbf{x}_\bullet$~~.

- Section 12.3.1 Each element of the attention matrix corresponds to a particular offset between key position a and query position b.

- Section 12.3.2 title: Scaled dot-product self-attention

- Section 12.3.2 This is known as *scaled dot-product self-attention*.

- Problem 12.1 How many weights and biases would there be in a fully connected shallow network relating all DN inputs to all DN outputs?

- Notes Page 236 Much subsequent work has modified just the attention matrix so that in the scaled dot-product self-attention equation:

- Problem 12.10: Extra bracket removed:

$$a[\mathbf{x}_m, \mathbf{x}_n] = \text{softmax}_m\left[\mathbf{k}_\bullet^T \mathbf{q}_n\right] = \frac{\exp\left[\mathbf{k}_m^T \mathbf{q}_n\right]}{\sum_{m'=1}^{N} \exp\left[\mathbf{k}_{m'}^T \mathbf{q}_n\right]}. \tag{1.23}$$

- Figure 13.10 Right hand column should be labelled as "output" not "hidden layer two". item Equation 13.22

$$\mathbf{H}'_k = \boldsymbol{\beta}_k \mathbf{1}^T + \boldsymbol{\Omega}_k \mathbf{H}_k. \tag{1.24}$$

- Section 15.1 A single new sample $\mathbf{x}_j^*$ is generated by (i) choosing a *latent variable* $\mathbf{z}_j$ from a simple base distribution (e.g., a standard normal) and then (ii) passing this data through a network $\mathbf{x}_j{}^* = \mathbf{g}[\mathbf{z}_j, \boldsymbol{\theta}]$ with parameters $\boldsymbol{\theta}$

- Section 15.2.1 If we divide the two sums in the first line of equation 15.5 by the numbers...

- Section 15.2.1 When $I = J$, the optimal discriminator for an example $\tilde{\mathbf{x}}$ of unknown origin is:

- Equation 16.12:

$$\text{f}[h, \boldsymbol{\phi}] = \left(\sum_{k=1}^{b-1} \phi_k\right) + (hK - b)\phi_b, \tag{1.25}$$

- Equation 16.25:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\text{argmin}} \left[\text{KL}\left[\frac{1}{I}\sum_{i=1}^{I} \delta\left[\mathbf{x} - \text{f}[\mathbf{z}_i, \boldsymbol{\phi}]\right] \middle|\middle| q(\mathbf{x})\right]\right]. \tag{1.26}$$

- Section 16.2 The first term is the inverse of the determinant of the $D \times D$ Jacobian matrix $\partial \mathbf{f}[\mathbf{z}, \boldsymbol{\phi}]/\partial \mathbf{z}$, which contains elements $\partial \text{f}_i[\mathbf{z}, \boldsymbol{\phi}]/\partial z_j$ at position (i, j).

- Section 17.5 we can't evaluate the evidence term $Pr(\mathbf{x}|\boldsymbol{\phi})$ in the denominator (see section 17.3).

- Equation 17.28 Brackets in third line now larger.

- Section 18.2 This is a *Markov chain* because the probability of $\mathbf{z}_t$ is determined entirely by the value of the immediately preceding variable $\mathbf{z}_{t-1}$.

- Section 18.6.1 The obvious architectural choice for this image-to-image mapping is the U-Net (figure 11.10).

- Section 19.7 Hence, the decision transformer replaces the reward r_t with the *returns-to-go* $R_{t:T} = \sum_{t'=t}^{T} r_{t'}$ (i.e., the sum of the empirically observed future rewards).

- Section 19.1.2 Rephrased as ambiguous: A *Markov reward process* extends the Markov process to include a distribution $Pr(r_{t+1}|s_t)$ over the possible rewards r_{t+1} received at the next time step, given that we are in state s_t.

- Section 20.5.1 In general, the smaller the model, the larger the proportion of weights that can... Note that this statement is only true for pure pruning methods, and not for lottery tickets where the pruned network is retrained from scratch, and so it has been removed.

- Section C.2.1 Rule 2:

$$\begin{aligned}\mathbb{E}\big[k \cdot \mathrm{f}[x]\big] &= \int k \cdot \mathrm{f}[x] Pr(x) dx \\ &= k \cdot \int \mathrm{f}[x] Pr(x) dx \\ &= k \cdot \mathbb{E}\big[\mathrm{f}[x]\big].\end{aligned}$$

- Section C.5.4 Reformulated to reflect the fact that the Fréchet/ and 2-Wasserstein distances are the same:

  The Fréchet/2-Wasserstein distance is given by:

$$D^2_{Fr/W_2}\Big[\mathrm{Norm}[\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1]\Big|\Big|\mathrm{Norm}[\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2]\Big] = |\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2|^2 + \mathrm{tr}\left[\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2 - 2\left(\boldsymbol{\Sigma}_1\boldsymbol{\Sigma}_2\right)^{1/2}\right]. \tag{1.27}$$