Compiled Notes

Joseph Marino

August 12, 2017

Contents

Chapter 1

Intelligence

1.1 Intelligence

Intelligence is a vaguely defined concept. In many situations, we often use the term to refer to the ability to perceive and understand certain concepts, such as math, art, politics, business, etc. But this narrow definition is human specific and fails to capture the broad spectrum that intelligence occupies. Instead, I use this working definition:

Intelligence is the ability of a system to perform meaningful actions within a particular environment.

We could argue whether a system that can perceive aspects of its environment but is unable to act is intelligent, but this is meaningless, as this system has no practical purpose. This definition has a few components: A **system** is some collection of matter: a molecular structure, single-celled organism, animal, machine, computer, human, society, etc. **Meaningful actions** are more difficult to define. In general terms, these are some non-random interactions with the environment, i.e. the distribution of actions given the environmental state has a relatively low entropy. Often we also associate these actions with achieving some goal or reward, although this is not strictly necessary. Finally, **within a particular environment** emphasizes the idea that intelligence is always specific to an environment (the data). Intelligence is not a characteristic of a system alone, it's always conditioned on the environment.

1.2 Creating Intelligence

There are two ways in which to create an intelligent system: through **design** and through **learning**. In design, one intelligent system constructs another intelligent system. In learning, a system gains intelligence through interaction with the environment.

Chapter 2

Probability & Statistics

2.1 Probability Basics

2.1.1 Definitions

2.1.2 Distributions

Chapter 3

Information Theory

3.1 Basic Concepts

Entropy is a measure of the uncertainty of a random variable. It tends to agree with the notion of information. Let X be a random variable, with alphabet $\mathcal{X}$ and probability mass function $p(x) = \Pr(X = x)$, with $x \in \mathcal{X}$. The entropy of X is defined as

$$H(X) = -\sum_{x\in\mathcal{X}} p(x) \log p(x). \tag{3.1}$$

The units of entropy depend on the base of the logarithm. If the base is 2, then the units are in bits. If the base is e, then the units are in nats. To convert between different units of entropy, use $H_b(X) = (\log_b a) H_a(X)$. Note that entropy can be interpreted as the expectation of $-\log p(x)$. Also, entropy is non-negative: $H(X) \geq 0$.

The **joint entropy** of multiple random variables is defined similarly:

$$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y) \log p(x,y) \tag{3.2}$$

Likewise, the **conditional entropy** is defined as

$$\begin{aligned} H(Y|X) &= \sum_{x\in\mathcal{X}} p(x) H(Y|X=x) \\ &= -\sum_{x\in\mathcal{X}} p(x) \sum_{y\in\mathcal{Y}} p(y|x) \log p(y|x) \\ &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y) \log p(y|x). \end{aligned}$$

Note that

$$H(X,Y) = H(X) + H(Y|X). \tag{3.3}$$

In words, joint entropy = individual entropy + conditional entropy. This also implies that

$$H(X,Y|Z) = H(X|Z) + H(Y|X,Z). \tag{3.4}$$

Because conditional entropy involves the conditional probability, we see that $H(Y|X) \neq H(X|Y)$. However,

$$H(X) - H(X|Y) = H(Y) - H(Y|X). \tag{3.5}$$

Relative entropy is a measure of the distance between two distributions. Put differently, it is the inefficiency of assuming the distribution of the data is $q(x)$ when the true distribution is $p(x)$. This is also referred to as the **Kullback-Leibler distance** or **divergence**.

$$D_{KL}(p(x)||q(x)) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} = \mathbb{E}_{x \sim p(x)} \Big[\log \frac{p(x)}{q(x)} \Big] \tag{3.6}$$

Note that if there is a value $x \in \mathcal{X}$ such that $p(x) > 0$ and $q(x) = 0$, then $D_{KL} = \infty$. KL divergence is non-negative and is only zero when $p(x) = q(x)$. It is not symmetric, so it is not a proper distance measure.

Mutual Information is the amount of information that one random variable contains about another random variable. In other words, it is the reduction in the uncertainty of one random variable due to knowledge of the other. It is expressed as the relative entropy between the joint distribution $p(x, y)$ and the product of the marginals $p(x)p(y)$.

$$\begin{aligned} I(X;Y) &= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \\ &= D_{KL}(p(x, y)||p(x)p(y)) \\ &= \mathbb{E}_{x,y \sim p(x,y)} \Big[\log \frac{p(x, y}{p(x)p(y)} \Big]. \end{aligned}$$

This can also be written as

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X). \tag{3.7}$$

Thus, X *says as much about* Y *as* Y *says about* X. In words, mutual information is the amount of information in X minus the amount of information in X after observing Y (and vice versa). This corresponds to the amount of information in X that is explained by Y (and, again, vice versa). If Y perfectly explains X, then $H(X|Y) = 0$, so $I(X;Y) = H(X)$. At the other extreme, if Y says nothing about X, then $H(X|Y) = H(X)$, so $I(X;Y) = 0$. Also,

$$I(X;Y) = H(X) + H(Y) - H(X, Y), \tag{3.8}$$

and

$$I(X;X) = H(X) - H(X|X) = H(X). \tag{3.9}$$

Thus, entropy can be considered as the amount of self-information.

Chapter 4

Neuroscience

4.1 Neurons

4.2 Neural Circuits

4.3 Brain Structures

Chapter 5

Neural Networks

5.1 Basics

5.1.1 Linear Threshold Units

5.1.2 Multi-Layer Perceptrons

5.1.3 Backpropagation

Chapter 6

Graphical Models

Chapter 7

Representations & Representation Learning

7.1 Motivation & Definitions

Overview of types of representations. How do we learning representations? What training criteria/conditions result in different representations? Are certain representations better than others? In which cases and why? The quality of a representation can only fundamentally be judged by its usefulness in helping to perform some task. That is, representations must be viewed in the context of a task.

7.2 Disentangled Representations

Define disentanglement. Why are disentangled representations helpful? Trade off between disentanglement and faithfully representing the data. How do we learn disentangled representations, while at the same time, respecting the structure of the latent space? Importance of priors.

Two random variables are **independent** if their joint probability can be expressed as the product of their marginals:

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y}), \tag{7.1}$$

and they are **conditionally independent** if their conditional joint probability can be expressed as

$$p(\mathbf{x}, \mathbf{y}|\mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{y}|\mathbf{z}). \tag{7.2}$$

Independence is related to covariance, but is a stronger property. Two variables that are independent have zero covariance and two variables that have non-zero covariance are dependent. Zero covariance implies that the variables have no *linear* dependence. Independence implies that the variables also have no *non-linear* dependence. That is, it is possible for two dependent variables to have zero covariance.

Notes:

[Cheung et al., 2014] propose using a cross-covariance penalty term in the setting of semi-supervised learning of the form:

$$C(\hat{\mathbf{y}}^{1\ldots N}, \mathbf{z}^{1\ldots N}) = \frac{1}{2}\sum_{ij}\left[\frac{1}{N}\sum_{n}(\hat{y}_i^n - \bar{\hat{y}}_i)(z_j^n - \bar{z}_j)\right]^2, \tag{7.3}$$

where $\hat{\mathbf{y}}$ is a vector of (one-hot) inferred labels and $\mathbf{z}$ is a latent representation. N is the batch size.

[Higgins et al., 2016] propose to use a regularization weight in a VAE setting to "up-weight" the amount of regularization as a means of promoting **redundancy reduction and independence** among the latent representation. They also argue for the importance of **dense sampling of the (continuous) data manifold** for disentanglement. Sparse sampling of the data manifold results in ambiguity is the manifold interpretation, requiring additional supervision for disentanglement.

[Siddharth et al., 2016] use supervision to impose structure on part of the latent representation in a VAE.

Dropout [Srivastava et al., 2014] is a technique for preventing "fragile coadaptation" between units within a representation, effectively enforcing that they represent different (independent) quantities.

Chapter 8

Predictive Coding

8.1 Introduction

The area of predictive coding, in its current form, was introduced by [Rao and Ballard, 1999]. This theoretical neuroscience model posited that sensory processing is primarily a filtering operation, in which latent quantities underlying the environment are estimated according to their agreement with the observed input as well as prior beliefs about these quantities. This offered an explanation for "extra-classical" receptive field effects in visual processing, in which stimuli outside the receptive field of a particular cortical neuron are able to affect that neuron's activity. This was explained as the result top-down prior beliefs.

TODO: mention other early work before Rao and Ballard with similar ideas

Predictive coding claims that sensory processing, i.e. perception, is fundamentally about constructing a generative model of the input sensory signal. To perform *inference* in this model, that is, to perceive, the model uses its current estimate of the latent variables underlying the environment to generate reconstructions or predictions of the input. Using the residual (error) from this reconstruction or prediction, along with residuals from prior beliefs, the model updates its estimate of these latent variables. *Learning* then corresponds to updating the parameters of the generative model to improve these residuals.

8.2 The Static Setting

8.2.1 MAP Estimation

Following [Bogacz, 2017]

Consider a problem in which the value of a single latent variable z must be inferred from a single observed variable x. Let g denote a non-linear function defining how z generates x. Assume that the generation output (prediction) takes the form of a normal distribution, with mean $g(z)$ and variance σ_x^2:

$$p(x|z) = \mathcal{N}(x; g(z), \sigma_x^2). \tag{8.1}$$

Recall that in one dimension, a normal distribution takes the form

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{8.2}$$

We also have a prior on z, which we also assume is a normal distribution with mean μ_p and variance σ_p^2:

$$p(z) = \mathcal{N}(z; \mu_p, \sigma_p^2). \tag{8.3}$$

In general, computing the exact posterior distribution is computationally intractable. Instead, we'll resort to variational inference to find the maximum of the posterior (MAP). Define our approximate distribution (a point mass estimate at ϕ) as $q(z|x) = \delta(z = \phi)$, with the MAP estimate being $\hat{\phi}$. In fact, we are not really using variational inference, since the maximum of $p(z|x)$ must also be the maximum of $p(x, z)$ through the definition of conditional probability (Bayes' Rule):

$$p(z|x) = \frac{p(x, z)}{p(x)}, \tag{8.4}$$

since $p(x)$ does not depend on the value of z.

We want to maximize the evidence lower bound (ELBO), $\mathcal{L}$:

$$\mathcal{L} = \mathbb{E}_{z\sim q(z|x)} \left[\log p(x, z) - \log q(z|x)\right] = \mathbb{E}_{z\sim q(z|x)} \left[\log p(x|z) + \log p(z) - \log q(z|x)\right] \tag{8.5}$$

$$\mathcal{L} = \log \left(\frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-\frac{(x-g(\phi))^2}{2\sigma_x^2}} \right) + \log \left(\frac{1}{\sqrt{2\pi\sigma_p^2}} e^{-\frac{(\phi-\mu_p)^2}{2\sigma_p^2}} \right) \tag{8.6}$$

$$\mathcal{L} = -\frac{1}{2}\log(2\pi\sigma_x^2) - \frac{(x - g(\phi))^2}{2\sigma_x^2} - \frac{1}{2}\log(2\pi\sigma_p^2) - \frac{(\phi - \mu_p)^2}{2\sigma_p^2} \tag{8.7}$$

$$\mathcal{L} = \frac{1}{2}\left(-\log(\sigma_x^2) - \frac{(x - g(\phi))^2}{\sigma_x^2} - \log(\sigma_p^2) - \frac{(\phi - \mu_p)^2}{\sigma_p^2} \right) + \text{const.} \tag{8.8}$$

To find the MAP estimate, we must solve

$$\hat{\phi} = \text{argmax}_{\phi}\mathcal{L} = \text{argmax}_{\phi}\frac{1}{2}\left(-\log(\sigma_x^2) - \frac{(x - g(\phi))^2}{\sigma_x^2} - \log(\sigma_p^2) - \frac{(\phi - \mu_p)^2}{\sigma_p^2} \right) \tag{8.9}$$

In order to do so, we can find the gradient of $\mathcal{L}$ w.r.t. ϕ:

$$\frac{\partial \mathcal{L}}{\partial \phi} = \frac{x - g(\phi)}{\sigma_x^2} g'(\phi) + \frac{\phi - \mu_p}{\sigma_p^2} \tag{8.10}$$

We see that we have two terms: the first term moves the estimate toward agreement with the observation, and the second term moves the estimate toward agreement with the prior. Each of these terms are weighted by their corresponding variances. By repeatedly moving ϕ according to this gradient, we can hopefully arrive at the MAP estimate $\hat{\phi}$ (if the optimization surface is not highly non-convex). That is, we can define the following dynamics for ϕ:

$$\dot{\phi} = \frac{\partial \mathcal{L}}{\partial \phi} \tag{8.11}$$

8.3 Neural Implementation

This

Chapter 9

Experiment Best Practices

This chapter contains some helpful practices for carrying out (machine learning) research projects.

9.1 GitHub

GitHub is an online platform for project source control. It is essential when collaborating on projects, as it allows you to meticulously track changes and manage conflicts between multiple versions of the code. But GitHub is also helpful when working alone on a project across multiple machines or if you just want to open source your code. The following are the basic commands for using GitHub:

Clone a repository:

```
git clone <repository address>
```

Add a file to the repository:

```
git add <filename>
```

Or to add everything to the repository:

```
git add .
```

or

```
git add -A
```

Commit changes to the repository:

```
git commit -m "<message>"
```

or

```
git commit
```

Note that if you run the second command, you will enter a vi interface where you can enter a multi-line message. To exit, type esc :wq

Push all committed changes to the repository:

```
git push
```

Pull the repository:

```
git pull
```

To create a new branch:

```
git branch <branch name>
```

Switch current branch:

```
git checkout <branch name>
```

Merge branch changes back into master (while currently checking out other branch):

```
git merge master
```

You can send pull requests on GitHub.com. After the branch is merged, you can delete the branch on the website.

To conclude, a good workflow practice is to (1) pull the repository at the beginning of each work session, (2) push any incremental changes at regular intervals, and (3) create new branches for any major changes to be implemented.

9.2 Experiment Logging

Maintaining proper records of experiments is an essential part of conducting good research. Scientific notebooks (preferably in a digital format) are often helpful for keeping high-level notes about experiments and methodologies, but one must keep experimental logs to track the different experimental set-ups that have been tested and the results. These are vital for surveying your project (what set-ups work? what set-ups should we consider trying?) and eventually communicating your results. Implementing and keeping track of these logs can seem like a hassle, but they are just as important as the experimental code itself.

A basic experiment logging pipeline looks something like this:

1. At the beginning of the experiment, **create a log directory dedicated to this experiment**. This log directory should have a unique name. An easy choice is the date and time of the start of the experiment, which allows one to sort all of the log directories in chronological order.

2. **Copy the source files into a subdirectory in the log directory**. This is absolutely vital, as it allows one to maintain the code that led to the result of that experiment, ensuring repeatability. If you are modifying the source code while running experiments, reproducing earlier experimental conditions can be a nightmare.

   The following code creates a new log folder for the experiment and saves the files from the current directory into a subdirectory called "source."

   ```
   import os
   from time import strftime

   def init_log(log_root):
       log_dir = strftime("%b_%d_%Y_%H_%M_%S") + '/'
       os.makedirs(log_path)
       os.system("rsync -au --include '*/' --include '*.py' --
           exclude '*' . " + log_path + "source")
   ```

3. In addition to maintaining source code of the experiment, it's important to **log the experimental results**. At the bare minimum, one should log train and validation performance metrics. Unless these metrics are logged, you have no means of comparing different experiments after they have run. There is an unending list of other results that can be logged, such as: outputs of the model, activations within the model, distributions of different metrics and parameters as they evolve during training, etc. The downside of logging this information is that it can slow down training and take up disc space. Often, it's best to set certain logging or visualization flags within your code. This allows you to log and understand what is happening within your model during the early experimental phase/debugging, then turn off this functionality during extended training sessions.

4. **Save the model weights**. This one seems obvious, but it is easy to be lazy and write code in which you cannot save and load previous models. If you want to keep training your model, perform further evaluation, or just run and demonstrate it's capabilities, it is essential that you build this functionality into your code. Many basic libraries will make this easy, but it's up to you to add this to your experimental code to make this seamless.

There are some upfront costs associated with setting up these practices within your code, but much of this code base can also be reused for each project. You have no excuse for not keeping proper records of your experiments!

9.3 Plotting

Bibliography

[Bogacz, 2017] Bogacz, R. (2017). A tutorial on the free-energy framework for modelling perception and learning. *Journal of Mathematical Psychology*, 76:198–211.

[Cheung et al., 2014] Cheung, B., Livezey, J. A., Bansal, A. K., and Olshausen, B. A. (2014). Discovering hidden factors of variation in deep networks. *arXiv preprint arXiv:1412.6583.*

[Higgins et al., 2016] Higgins, I., Matthey, L., Glorot, X., Pal, A., Uria, B., Blundell, C., Mohamed, S., and Lerchner, A. (2016). Early visual concept learning with unsupervised deep learning. *arXiv preprint arXiv:1606.05579.*

[Rao and Ballard, 1999] Rao, R. P. and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87.

[Siddharth et al., 2016] Siddharth, N., Paige, B., Desmaison, A., de Meent, V., Wood, F., Goodman, N. D., Kohli, P., Torr, P. H., et al. (2016). Inducing interpretable representations with variational autoencoders. *arXiv preprint arXiv:1611.07492.*

[Srivastava et al., 2014] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Appendix A

Appendix 1