Research Journal

Jerome Wynne

University of Bristol

19th June 2017 - Present

Contents

Thursday, 1st June 2017

1 Summary

- Set up this labbook.
- Wrote down what I'd like to achieve during my placement.
- Listed a bunch of classic *Computer Vision* papers.

2 Placement objectives

During my 8-week placement I would like to:

- Apply and understand the most useful *Computer Vision* algorithms.
- Write either a statistics, machine learning, or *Computer Vision* research paper worthy of publication.
- Develop a systematic workflow to problems involving data, and demonstrate this workflow in publicly available scripts or notebooks.

3 Reading a paper critically

- Motivations for the problem posed
- The choices made in finding a solution
- The assumptions behind the solution
- The validity of those assumptions
- Whether assumptions can be removed without invalidating the approach
- What was accomplished
- Future research directions

4 Structuring your time

I think it's realistic to aim for 12 hours of productive work each day. This includes reading, studying, and running my own experiments. To begin with, I think I should spend at least three-quarters of this time learning. Maybe by week four I will shift the ratio.

I will see how it goes, but I suspect I will prefer to work from the University rather than DNV-GL's offices.

To make this placement a success, I will need to be disciplined about how I use my time. I know that if I get up early and immediately go to work, I can easily crack out four hours without breaking a sweat. After this time I can take at least a couple of hours - go to the gym, read, walk, or - of course - eat. After that I'll get back on that horse for another four or five hours, before taking another short break, then do a few more hours.

Having lots of sleep is important for my mental health and emotional wellbeing, so I should aim to get at least eight hours. If I get up at six, this means that I should go to bed at half-nine. I can work from my flat, University buildings, or the DNV-GL offices. It would be a good idea to mix the places I study up - I know this has helped me keep focused in the past. I need to be careful to provide some time for myself too, so that I can recuperate: I think two hours at the end of the day, eight until ten, will be enough.

5 Figuring out what to write about

1. Narrow down your field of study

2. Define what to investigate

3. Establish a thesis or an argument

I am studying *Computer Vision* and statistics. This is because I want to build robust algorithms to understand visual information, which in turns makes it easier to automate difficult or tedious tasks, such as watching CCTV cameras or spotting damage to structures in video footage. I am doing this so that we will know more about how patterns in visual information can be found, and so that we can exploit these patterns to automate economically and socially beneficial tasks. I am also interested in how we can represent visual information to make it easier to work with and to understand.

6 Structuring your work

1. Find data that poses an unsolved problem, or find a problem that needs data to be solved.
2. Find the data, or pose the problem.
3. Review the relevant literature.
4. Propose a solution, then run experiments and conduct analysis to test that solution.
5. Write up the results.

7 How to use this book

I should use this book to record what I'm doing, ideas and conversations I have, experiments I run, papers and books to read, things I understand and don't understand - everything related to my research. It only takes a few minutes to put a screenshot in this document - remember this!

As a habit, at the beginning of each day I will write down what I plan to do. At the end of the day I'll review what I've done. I should also review the book each week, to get an idea what I've been up to and where I'm headed.

Thursday, 1st June 2017

Papers

Title	**Authors**	**Year**	**Topic**	**Read**
Theory of Edge Detection	D. Marr, E. Hildreth	1980	Computer vision	No
A Computational Approach to Edge Detection	J. Canny	1986	*Computer Vision*	No
Determining optical flow	B.K.P. Horn, B. Schunk	1981	Computer vision	No
An iterative image registration technique with an application to stereo vision	B. Lucas, T. Kanade	1981	Computer vision	No
Snakes: Active contour models	M. Kass, A. Witkin, D. Terzpoulos	1988	*Computer Vision*	No
Eigenfaces for recognition	M. Turk, A. Pentland	1991	*Computer Vision*	No
Shape and motion from image streams under orthography: a factorization method	C. Tomasi, T. Kanade	1992	*Computer Vision*	No
Texture features for browsing and retrieval of image data	B. Manjunath, W. Ma	1996	*Computer Vision*	No
Conditional density propagation for visual tracking	M. Isard, A. Blake	1998	*Computer Vision*	No
Normalized cuts	J. Shi, J. Malik	2000	*Computer Vision*	No
Non-parametric model for background subtraction	A. Elgammal, D. Harwood, L. Davis	2000	*Computer Vision*	No
Distinctive image features from scale-invariant keypoints	D. Lowe	2004	*Computer Vision*	No

Monday, 19th June 2017

1 Summary

- Arrived at DNV offices @ 10:00.
- H & S induction. Remember to look at the green lights above the doors!
- Met Elizabeth Traiger:
  - Solar panel cracks toy problem.
  - DNV are keen to use Python.
  - Approx. 1500 minutes of turbine data available.
  - We need a method for labeling the data.
- IT was not set up - she will email me tomorrow when it is ready.
- DNV use Microsoft Azure services internally.
- The company is open to using TensorFlow.
- We agreed that weekly meetings would be appropriate. She is away this Friday (in London, speaking about using satellite imagery of windfarms to monitor fish populations (?), in association with the European Space Agency) and next week (site visit?).
- Various ideas came up in our conversation: super-resolution methods, using simulation data, creating synthetic data, thinking about how to segment the images.
- Footage from two turbines is available this week. More footage will be available next week.
- I can VPN into DNV's network using the laptop that they provided me.
- I left the DNV offices after lunch (14:00), then worked on the Design Project briefs until 16:30.
- Liz planned to book a meeting with the inspections team to discuss how they look for damage. Possible questions:
  - What does damage look like?
  - Over what time periods does it develop?

  - What causes the damage?
  - What mistakes do you make when looking for damage?
  - What type of damage is easiest/most difficult to spot?
  - What do you do when you find damage?
  - What problems do you think there are with using drones to identify damage?

In the afternoon I read the introduction to Canny's paper on edge detection. I learnt that there is a tension between edge detection and localization (the distance between the detected edge and the true edge). Apparently the paper determines the optimal operator for various common edges.

I also read the introduction and first chapter of S.J. Prince's book on computer vision. The introduction outlined the book's structure and the first chapter was a quick review of basic probability theory.

- Read Canny paper.
- Read the Chapter 1, 3, and 6 of S.J. Computer Vision. Make notes and complete the associated exercises.
- Apply what was learnt in this chapter using Python.
- Outline the structure of compute vision / machine learning.

2 Prince: Computer Vision (Ch. 1, Introduction)

The book's structure was outlined. It consists of six sections:

1. Probability.
2. Machine learning for machine vision.
3. Graphical models (principled ways to simplify the relationship between image data and estimated properties) for machine vision.
4. Image preprocessing.
5. Geometric machine vision - generating 3D models based on image data and localizing a camera based on its view.
6. Families of machine vision models for common problems.

It took me about ten minutes to read through this first chapter of five pages. Relevant papers are listed at the end of each of the book's sections.

3 Prince: Computer Vision (Ch. 2, Introduction to Probability)

The second chapter took about an hour to read and make notes for. Almost all of it was review material: random variables, distributions, joint distributions, Bayes' rule, conditionality, and expectation. I did learn, however, two useful things:

- A Hinton diagram can be used to visualize probability mass functions. A square is associated with each point; the square's area relative to the total area of all squares corresponds to that value's relative probability. Figure 1 shows a Hinton diagram in two variables.

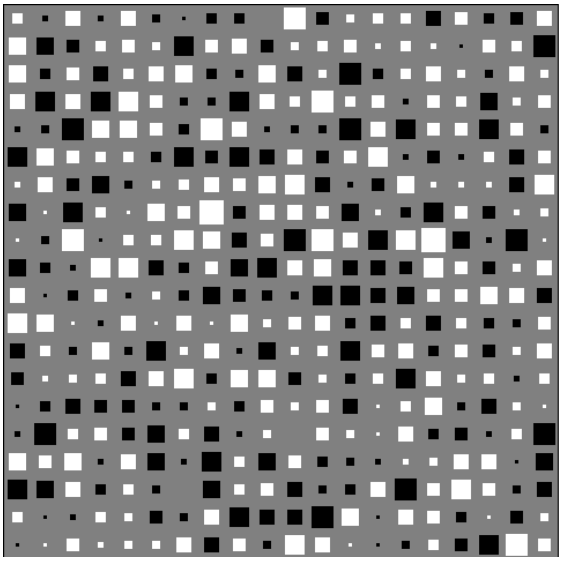


Figure 1: A Hinton diagram can be used to visualize probability mass functions in one or two variables.

- A useful identity: if two random variables are independent, then the product of their expectation is equivalent to the expectation of their product. More generally:

$$E[f(x) \cdot g(y)] = E[f(x)] \cdot E[g(y)] \tag{0.1}$$

Tuesday, 20th June 2017

1 Summary

I woke at six (I'm getting the hang of it). For the first hour of the day I reviewed the project brief MZ had commented on and sent it off to PH. PH mentioned I could use his office for the next two weeks while he was away - hopefully I can pick up the key from him this afternoon.

Today I would like to complete Chapters 3 and 4 of Prince's Computer Vision, do some modeling in Python, and begin the CVX course. I should also get an overview of the main *Computer Vision* tools for Python.

I made notes on Chapter 3 and completed five of its exercises, which related to conjugacy, the exponential family, and the modes of the distributions studied. PH said to pick up the key to his office from the porter. ET emailed me to say that IT have set up my account - I agreed to go in tomorrow morning to meet her and pick up my laptop. I should also ask her when we plan to meet the wind turbine inspections team.

I intend to complete Chapter 4 this afternoon. I am now going to read Canny's paper. I tried to read Canny's paper but kept zoning out. I did understand, however, that he formulated an optimization problem by expressing detection performance as signal-to-noise ratio and localization performance as a similar quantity but involving gradients.

I spent fifty minutes reading through Chapter 4. I made notes on maximum likelihood, maximum a posteriori, and bayesian methods for optimizing parameter estimates. I read through an example wherein the mean and variance of a univariate normal distribution were estimated using a normal-scaled inverse gamma prior; as it turns out, the benefits of using the conjugate here are twofold - not only does the posterior have a closed-form solution, but so also does the posterior predictive!

2 Prince: Computer Vision (Ch. 3, Common Probability Distributions)

I started on Chapter 3 of Prince's book yesterday. It covers eight probability distributions that will be useful for machine vision purposes. Four of these are for modelling image data or world properties; the other four are for modelling the parameters of these four.

I made notes, but have yet to complete the exercises. The distribution pairs covered were:

- Bernoulli - Beta
- Categorical - Dirichlet
- Univariate normal - Normal-scaled inverse gamma
- Multivariate normal - Normal inverse Wishart

The four distributions on the left were familiar to me, as was the beta distribution (however it was useful to note that the expectation of Beta$[\alpha, \beta]$ is defined by $\frac{\alpha}{\alpha+\beta}$). The Dirichlet, normal-inverse scaled gamma, and normal inverse wishart distributions were new to me.

The Dirichlet distribution is effectively a generalization of the beta distribution to cover more than one variable. It's used to model the K parameters of a categorical distribution. As such, $\sum_k \lambda_k = 1$ at all points in the Dirichlet distribution. Its hyperparameters $\alpha_1, ..., \alpha_K$ describe the categorical distribution's expecations over each of its variables $E[\lambda_1], ..., E[\lambda_K]$; the magnitude of the α values determines the Dirichlet's dispersion.

The Normal-scaled inverse gamma distribution is used to model uncertainty in the mean and variance of a univariate normal distribution. It has four parameters. In a similar vein, the normal inverse Wishart distribution describes uncertainty in the mean vector and covariance matrix of a multivariate normal distribution.

I'm now going to complete three questions from the exercises, then go to the barbers. When I get back I will complete three more exercises, before continuing to read the Canny paper.

I completed three questions. The first asked for the mode of the beta distribution (found by maximizing the pdf w.r.t. the variable). The second pointed out that all of the distributions in the chapter were members of the exponential family; it then requested the beta pdf in exponential family form. The third related a normal prior and restricted normal likelihood to a normal posterior (i.e. it was a conjugacy problem). These questions together took me about forty-five minutes to complete.

This is not the first time the exponential distribution has been mentioned - why is understanding multiple distributions as variants on one distribution useful?

I completed two more questions: showing that the normal-scaled inverse gamma is the conjugate prior to the univariate normal distribution; showing that the Dirichlet distribution is the conjugate prior to the categorical distribution. They were mostly exercises in algebra, but it did drive home why the Dirichlet and normal-scaled inverse gamma distributions are relevant, and what they look like (particularly w.r.t. the Dirichlet-beta similarities).

Tuesday, 20th June 2017

3 Prince: Computer Vision (Ch. 4, Fitting Probability Models)

The chapter began with maximum likelihood, maximum a posteriori, and bayesian methods for optimizing parameter estimates. I read through an example in which the mean and variance of a univariate normal distribution were estimated using a normal-scaled inverse gamma prior; as it turns out, the benefits of using the conjugate here are twofold - not only does the posterior have a closed-form solution, but so also does the posterior predictive! I think I would benefit from writing out the Bayesian solution, and possibly running a simulation in Python. Maybe I'll go through the other example before building the simulation.

I made notes on both worked examples. The second example involved inferring posterior and posterior predictive distributions from a categorical likelihood and its conjugate (a Dirichlet prior). As in the case of the univariate normal - normal-scaled inverse gamma problem, bott distributions had neat solutions. The posterior was another Dirichlet distribution (with correspondingly updated parameters $\tilde{\alpha}_j = \alpha_j + N_j$) and the posterior predictive turned out to be a function of beta functions.

I'm going to write two R simulations, one for each example.

I built an R simulation for the first example.

I attempted Q4.6, but fell short (the algebra became untangled). I will try again tomorrow morning; perhaps doing Q4.5 first will make it slightly easier. Once I've done those two questions, I should like to do Q4.7-4.10. Worked 4 - 7:30 on Com. Vis. Chapter 4.

This evening I watched the first lecture from the CVX course. It explained what convex optimisation is, common variants of it (least squares and linear programming), discussed application areas, defined a convex functoin, provided an overview of the course, and gave some history on the topic.

Wednesday, 21st June 2017

1 Summary

Today I plan to:

- Complete the exercises from Ch. 4 of Computer Vision
- Walk through one of the examples available at PyImageSearch
- Read Ch. 5 of *Computer Vision* and complete the associated exercises
- Pick up Paul Harper's office key
- Meet ET at 10:00 to have IT induction
- Explore solar panel dataset
- Investigate Python tools for tagging images
- Read Chapter 1 of Boyd's Convex Optimisation

(07:00 - 09:50) I spent the first hour and forty-five minutes of the day completing Exercises 4.5 - 4.10 of Computer Vision. I then made notes on Chapter 5, which is focused on the normal distribution. It began by showing the difference between spherical, diagonal, and full covariance matrices - spherical matrices have circular isodensity contours, diagonals have ellipsoidal ones whose axes are aligned with coordinate axes, and full covariance matrices have ellipsoidal contours in any orientation.

As it happens, it's possible to rotate a distribution's coordinate system to permit a full covariance matrix to be reexpressed as a diagonal one. The rotation matrix is deduced using the singular value decomposition (something I should probably know, but don't). I'm about to head off to DNV, but I intend to watch Gilbert Strang's lecture on the singular value decomposition when I get back. I think I could also do with inspecting matrix inverses more closely.

I spent the afternoon (10:30 - 16:00) in the DNV offices getting set-up with IT and so on. I also went through the two datasets made available to me so far. In the evening I wrote a Python script to resize and subset the PV panel images. I also skimmed Ch. 6 of Computer Vision.

Tomorrow morning I should label a few hundred of these images and try out the algorithms described in Ch. 6 of Com. Vis. For the first three hours of the day I should

like to study from the book. The middle hours can be spent labelling and coding, before returning again to studying. I will read through Ch. 1 of S. Boyd's book on convex optimization also.

I did find a tool for annotating images - it's called Fiji (https://fiji.sc/). I should really start time-stamping my entries on this...

I also read a paper [*Detection and Classification of Surface Defects on Gun Barrels Using Computer Vision and Machine Learning*, Sanmuhamani et al.] on gun barrel damage classification. They had success using an extended maxima transform was used to segment their images. They used sequential forward-feature selection. The six features that presented themselves as most useful (in a texture-based classification problem, fed into a radial-basis SVM) were:

- Mean
- Variance
- Smoothness
- Skewness
- Uniformity
- Entropy

I think it would be worthwhile getting the definitions for these should any texture-based classification work come up.

I need to start logging to-dos using the Office 365 Planner app.

2 PV Panels

The first dataset is a 50 scans of PV panels. Possible objectives with these panels are:

- Detecting cracks.
- Detecting pit marks.
- Detecting blown cells.

An example image is shown in Figure 1. I made a repo for the PV project.

3 PV Panels: Crack detection

The objective of this task is to label images as according to whether or not they contain a crack - I don't think bounding boxes are appropriate. I'm going to begin by:

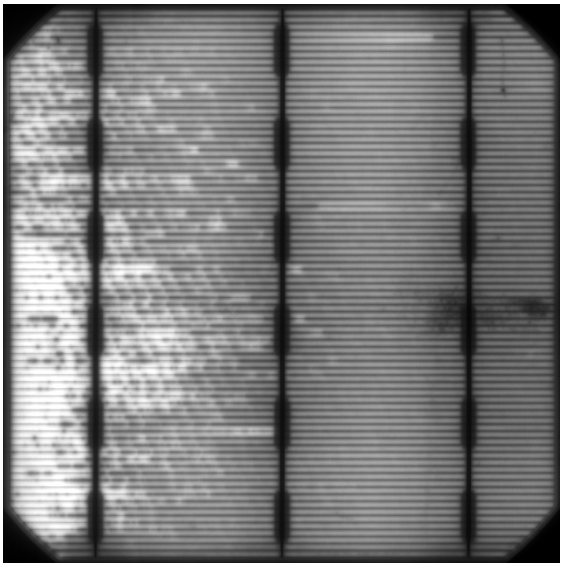


Figure 1: A sample image from the PV panels dataset.

- Rescaling the images to a constant size.
- Labeling the panels according to the damage they've incurred.
- Sorting the panels according to their type.

I created `panel-resizer.ipynb` to rescale the panel images to a fixed size[1]. This took me about an hour. While doing this, I realized that I could cluster the panels according to their metropolis distance from one another, generate an 'undamaged' panel for each class, then use subtraction to identify damaged panels. I think the biggest weakness with such an approach, however, is the need for an 'undamaged' panel image. How is easy would this be to generate synthetically?

1. Label images (crack/no crack).
2. 70/30 split of training/test data.
3. Benchmark 'dumb' classifiers.
4. Select the best dumb classifier and optimize it.

[1] I deleted one duplicated image (52) and removed three smaller images

Thursday, 22nd June 2017

1 Summary

Today I planned to:

- Review what I covered yesterday.
- Read the blog post "Classifying plankton with deep neural networks".
- Label at least 50% of the PV panel data.
- Set up a LaTeX report for the PV panel data.
- Update my current strategy for the PV data on Office 365.
- Complete problem exercises from Ch. 6 of *Computer Vision*.
- Implement the algorithms presented in Ch. 6 of *Computer Vision* on a dummy dataset.
- Benchmark 3 dumb classifiers against the labeled PV panel data.
- Read Ch. 1 of Boyd's *Convex Optimisation*.

If there is time I will also begin reading Ch. 7 of *Computer Vision*.

(07:30 - 08:00) I planned my approach to the PV panel crack detection task on Office 365.

(08:00 - 11:00)

2 PV Panels: Crack Detection

To-dos:

- Label subset images
- Build augmented images dataset
- Set up report
- Benchmark simple classifiers against augmented images

- Optimize best simple classifier
- Record simple classifier performances in report
- Set up Azure instance and train convnet against augmented images
- Upload augmented images dataset
- Investigate clustering of panels

3 Prince: Computer Vision (Ch. 5/6)

Yesterday my work from this book constituted reviewing the many properties of the multivariate normal distribution, among which were:

- The marginal distribution of a subset of a vector of normal r.v.s is also normal.
- The conditional distribution of a subset of vector of normal r.v.s, given the other r.v.'s values, is also a normal distribution.
- Its covariance matrix an be diagonalized using the singular value decomposition, which forms a rotation matrix that twists the reference frame to align with the axes of the distribution's ellipsoidal isodensity contours.
- It can have one of three types of covariance matrix - spherical, diagonal, or full - which determines how elliptical its isodensity contours are and what its orientation relative to the coordinate axes is.

In the evening I began work on Ch. 6, *Learning and Inference in Vision*. It defined the basic task of machine vision as inferring world properties based on image data. It also outlined a framework for solving this problem consisting of three components:

- A model relating image data to world properties.
- A learning algorithm for adjusting that model's parameters based on observations.
- An inference algorithm for returning a predicted world state based on input image data.

The model can be either discriminative or generative. A discriminative model describes $\Pr(\mathbf{w}|\mathbf{x})$, the world state based on the image data. By contrast, a generative model instead represents the likelihood $\Pr(\mathbf{x}|\mathbf{w})$ - what the image data should be for a given world state.

(08:20 - 9:20) Made notes on modeling a univariate continuous response with univariate continuous inputs using discriminative and generative methods. Also described how to model a binary response in terms of continuous image data, again via both generative and discriminative methods.

- Univariate continuous data x, univariate continuous world state w

  - Discriminative approach: Model $\Pr(w|x)$ using $\text{Norm}_w[\phi_0 + \phi_1 x, \sigma^2]$. Fit $(\phi_0, \phi_1, \sigma^2)$ based on training set $\{x_i, w_i\}_{i=1}^I$.
  - Generative approach: Model $\Pr(x|w)$ using $\text{Norm}_x[\phi_0 + \phi_1 w, \sigma^2]$, and $\Pr(w)$ using $\text{Norm}_w[\mu_p, \sigma_p^2]$. Fit (μ_p, σ_p^2) using the available world states $\{w_i\}_{i=1}^I$. Fit $(\phi_0, \phi_1, \sigma^2)$ using the training data $\{x_i, w_i\}_{i=1}^I$. Compute the posterior using Bayes' rule.

- Univariate continuous data x, binary world state w.
  - Discriminative approach: Model $\Pr(w|x)$ using a Bernoulli distribution parameterized by $\lambda = \text{sig}[\phi_0 + \phi_1 x] = \frac{1}{1+\exp(-\phi_0 - \phi_1 x)}$. Fit (ϕ_0, ϕ_1) using the training data.
  - Generative approach: Set up a normal distribution over the data $\Pr(x|w) = \text{Norm}_x[\phi_0 + \phi_1 w, \sigma^2]$ such that its parameters switch according to the world state $w \in \{0, 1\}$. Use a Bernoulli distribution for the world state prior $\Pr(w) = \text{Bern}_w[\lambda_p]$ - fit λ using the observed world states. For inference compute the posterior at the chosen set of inputs.

(09:30 - 10:10) Read through skin detection and background subtraction examples and the chapter summary. The skin detection algorithm set up a generative model describing pixel intensity based on skin/non-skin status via a multivariate normal distribution. The prior was a Bernoulli distribution on the world states. By contrast, the background subtraction example set up a likelihood function wherein the pixel intensities of foreground objects were described using a uniform distribution (i.e. all combinations of (x_R, x_G, x_B) were deemed equally likely) and the intensity at each background pixel in each scene was expressed in terms of a multivariate normal distribution.

(10:20 - 11:45) Completed Ch.6 exercises 1, 3, 5, 7. Related modelling a continuous response against a binary world state to quadratic discriminant analysis and linear discriminant analysis (I ended up seeing this when comparing the for the decision boundary formed by two normal distributions with equiprobable associated classes with the decision boundary derived from a logistic regression classifier).

(12:50 - 13:45) Labeled PV panels dataset.

(14:00 - 16:05) Read (and made notes on) Ch. 7 of *Computer Vision*.

(16:05 - 16:45) Typed up lab notes from this afternoon's work on hidden variables and the EM algorithm.

(19:15 - 22:15) Augmenting PV panel images. Image augmentations:

- vertical flip
- rescaling
- adding noise

4 Prince, Computer Vision (Ch. 7, Modeling complex data densities)

This chapter frames the methods introduced by considering the shortcomings of attempting to build a generative model for a face classifier. Given a worldstate corresponding to face/no face, $w \in \{0, 1\}$, what is the probability density function for each pixel's intensity in an image of a fixed size? We might choose to model this likelihood using a normal distribution whose parameters are fit using training data:

$$\Pr(\mathbf{x}|w) = \text{Norm}_{\mathbf{x}}[\mu, \mathbf{\Sigma}]$$

Notice that given $w = 1$, the parameters will be $\{\mu_1, \mathbf{\Sigma}_1\}$, and given $w = 0$ the parameters will be $\{\mu_0, \mathbf{\Sigma}_1\}$. By defining a prior over the world state (a Bernoulli distribution whose parameter λ is fit using training data), we can get a hold of a posterior describing the probable world state, given a particular image's set of pixel intensities, $\Pr(w|\mathbf{x})$.

This modeling strategy has several shortcomings:

- The normal distribution is sensitive to outliers - one extreme observation will set estimates of its mean and covariance way out of kilter.
- A unimodal distribution doesn't describe what pixel intensities are probable if someone's face is in an image - people have different skin, eye, and hair colours.
- An outrageous number of parameters needs to be estimated. Given a 30×30 RGB image, 2700 mean intensity values and $\frac{2700 \times 2701}{2}$ covariance values need to be estimated. To make this estimate unique then, about three-and-a-half million training examples are needed.

To get around these problems we can - respectively - use t-distributions, mixture models, and subspace models. All of these tools exploit hidden variables. A joint distribution over the variable being modeled $\mathbf{x}$ and a hidden variable $\mathbf{h}$ is defined such that when $\Pr(\mathbf{x}, \mathbf{h})$ is marginalized over $\mathbf{h}$, a complex density function of $\mathbf{x}$ emerges. As with any distribution, this joint distribution is parameterized by some set of values θ which are chosen to maximize the likelihood function:

$$\hat{\theta} = \arg\max_{\theta} \left[\sum_{i=1}^{I} \log \left[\int \Pr(\mathbf{x}_i, \mathbf{h}_i|\theta) d\mathbf{h}_i \right] \right]$$

The expectation maximization algorithm is used to maximize the likelihood. This algorithm proceeds by iteratively raising a lower bound on the likelihood function. The lower bound is a function of the parameters θ and density distributions over the hidden variables. On the t^{th} iteration of the algorithm:

1. The boundary is raised as far as possible by setting the density functions $q_i(\mathbf{h}_i)$ to be $\Pr(\mathbf{h}_i|\mathbf{x}_i, \theta^{[}t-1])$.

2. $\theta^{[t]}$ is chosen to maximize the log-likelihood for the given density functions.

The first step is called the E-step (for expectation) and the second step is called the M-step (for maximization). By executing this algorithm repeatedly, it is possible to converge upon a local maximum in the likelihood function (which may also be the global maximum).

Use various scales of panel images to generate more data.

Friday, 23rd June 2017

(08:00 - 13:30) Read and made notes on Ch.7 of *Computer Vision*. Covered hidden variables, expectation maximization, mixture of gaussians, the t-distribution, and factor analysis. Spent three hours building a mixture of gaussians model using World Happiness Report data. Figure 1 shows a fit being made with four component distributions. It may not look like much, but I learnt a lot from doing it! (16:00 - 17:45) Made notes on

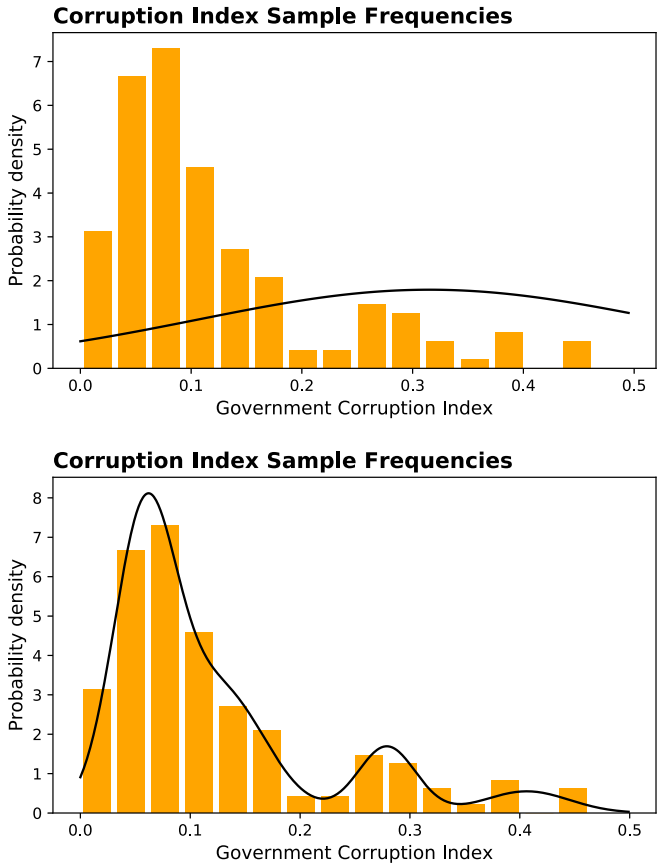


Figure 1: Fitting a four-component mixture of gaussians model using expectation maximization.

factor analyzers.

Friday, 23rd June 2017

(17:45 - 20:00) Ironed out some bugs in the PV panel organizer. Ran a random forest and an rbf SVM against augmented traiing data (they did all right, 80% accuracy). I'll type out what I did in more detail later.

1 Prince, Computer Vision (Ch.7 Modeling Complex Data Densities)

So today I covered three things, all of in this chapter, all bearing a single commonality. They were introduced as tools with which to address weaknesses of using a normal model model for the likely intensities of a pixel (or some other continuous feature) for a given world state. In particular, a single peak probably doesn't represent reality very well, the parameter fits are very sensitive to outliers, and a vast number of parameters need to be determined as a result of the covariance matrix. The solutions to these problems, respectively, were:

- Using a mixture of gaussians instead of one.
- Using a t-distribution rather than a normal distribution.
- Reducing the number of parameters to be determined by only modeling the covariation in directions where it is strongest.

Neatly, it turns out that all of these model variations can be represented in terms of hidden variables (which I described yesterday). A mixture of gaussians uses a categorically distributed hidden variable to weight the various normal distributions, each of which have their own covariance and mean. A variable with a t-distribution can actually be thought of as a normally distributed variable whose variance is parameterised by a scalar gamma-distributed variable. That is,

$$\begin{aligned}\Pr(\mathbf{x}|\mathbf{h}) &= \text{Norm}_{\mathbf{x}_i}[\mu, \mathbf{\Sigma}/h] \\ \Pr(\mathbf{h}) &= \text{Gamma}_h[\nu/2, \nu/2]\end{aligned}$$

While I had previously appreciated that the t-distribution's greater spread is a consequence of uncertainty w.r.t. its variance, this formulation made it especially clear what the role played by the degrees of freedom is.

The third solution - factor analyzers - is something I'm still wrapping my head mathematically but I think I understand in principle. Sample a vector $\mathbf{h}$ from a multivariate standard normal distribution then use it to adjust the mean of a normal distribution to somewhere in the subspace described by its factors. Sample $\mathbf{x}$ from this shifted normal distribution. The hidden variable here is the sampled vector: the distribution of samples from the shifted normal distribution turns out to be normal also, but with a covariance such that:

$$\Pr(\mathbf{x}|\mathbf{h}) = \text{Norm}_{\mathbf{x}_i}[\mu, \phi\phi^T + \mathbf{\Sigma}]$$

Where ϕ is a $D \times K$ matrix whose columns are the basis vectors (factors) of the subspace and $\mathbf{\Sigma}$ is a $D \times D$ diagonal (or spherical) covariance matrix. The factors are chosen to describe as much of the covariation in the full columnspace as possible, with the diagonal covariance matrix capturing the meat of the variance: a factor analyzer attempts to capture as much of the variation in the data's full D-dimensional columnspace in terms of an $\mathbb{R}^K$-dimensional subspace. One of the things I'm still trying to understand is why $\phi\phi^T$ represents covariation in the subspace - I plan to look at this today (I wrote the follow-up to the 23^{rd}'s work on the 24^{th}). As an aside, the author mentioned that when $\mathbf{\Sigma}$ is a spherical covariance matrix (that is, a scalar multiple of the identity matrix), this method is called principal component analysis. The benefit of using a spherical rather than a diagonal matrix is that it reduces further still the number of parameters to be fit (hence its widespread use when wrangling wide data).

The reason the author framed the t-distribution, mixture of gaussian, and factor analysis model in terms of hidden variables is because density functions that can be expressed as a marginalization over a hidden variable (Equation 1) lend themselves to optimization via expectation maximization.

$$\Pr(\mathbf{x}|\theta) = \int \Pr(\mathbf{x}, \mathbf{h}|\theta) \cdot d\mathbf{h}$$

The model parameters θ can be fit by iterative maximization of a bound on the l.h.s's likelihood function. This bound is a function of the r.h.s.; EM raises this boundary by iteratively:

- updating density functions over the hidden variables,
- maximizing the boundary's value by adjusting the values of the parameters in θ.

Eventually, the boundary converges on the log-likelihood function such that a set of locally optimal parameter values are found.

As I said, I think there are still things about factor analyzers I'd like to understand; I would quite like to build a dummy model tomorrow to get more of a feel for how they work. I also need to look at the proof the boundary that the EM algorithm exploits and understand what its limitations are.

2 PV Panels: Crack detection

As I briefly mentioned in today's summary, I fixed some bugs in the `panel-organizer` set of scripts and ran some simple classifiers against them, namely a random forest and an rbf SVM. Like I said, they both achieved accuracies of $\approx 80\%$. Given that these models don't exploit the spatial information available in the images (I downsampled them from 159×159 to 60×60, then flattened them), I'm optimistic about what we can achieve. I am aware that there might still be a bug w.r.t. class balances (I think I may have generated too many augmented crack images) and that only augmenting cracked panels

is methodologically flawed (augmentation introduces distinctive features to the images), problems I intend to resolve tomorrow. I would also like to experiment with capturing spatial data (e.g. HOGs or moments, I feel that reading would also help here). It may also be the case that segmenting the images might help, as might clustering models of panel. I also need to set up a report for the panel work and conceive of a system to log classifier/preprocessing performance (or borrow a method from someone smarter than me!).

Saturday, 24th June 2017

I woke at 06:45 this morning to swim for an hour and a quarter (I did 64 lengths of a 30m pool - 1920m). I had never swum more than 2 or 3 lengths in a pool before: my form was terrible and for the first five lengths I had difficulty breathing. The breathing trouble made me slightly panicked, which made it even harder to breather (and so on). I pushed through however, and by the end I was breast-stroking like an enthusiastic amateur. I also felt pretty great. I swam the lengths as part of a relay effort parallel to a friend's Channel attempt (1000 lengths, 30km) and felt that doing the lengths helped me to better understand their achievement.

I spent a couple of hours with MZ discussing various things, most notably our 4^{th} year project, a book on scarcity he is reading, and the VEST and Fulbright scholarships. I intend to apply for both of these scholarships, so I should probably capture the information here.

(12:00 - 14:40) I spent the first hour-and-a-half reviewing what I worked on yesterday and basically attempting to orient myself for the forthcoming week. I then an hour understanding the Fulbright and Vest scholarships. With regards to these, at the beginning of August I need to begin drafting my university and funding applications. I should also make a schedule of application deadlines and requirements. Register for ACT/SAT tests. This evening and tomorrow morning I should make time to plan my research placement more thoroughly.

(14:40 - 16:40) Detailed notes on the EM algorithm (yay understanding). Prove Jensen's inequality.

(16:40 - 19:10) Wrote source code for identifying faces using a factor analyzer. Found dataset (`http://cswww.essex.ac.uk/mv/allfaces/faces94.html`), imported and labeled the data.

Today I would like to neaten up my understanding of the EM algorithm, build a factor analyzer, and read the first chapter of Boyd's *Convex Optimization*.

1 Fulbright Scholarship

Fulbright scholarships are awarded by the US Bureau of Educational and Cultural affairs under the Department of State. Approximately 800 scholarships are awarded each year to international students (Visiting Scholars). Emphazises leadership, commitment to

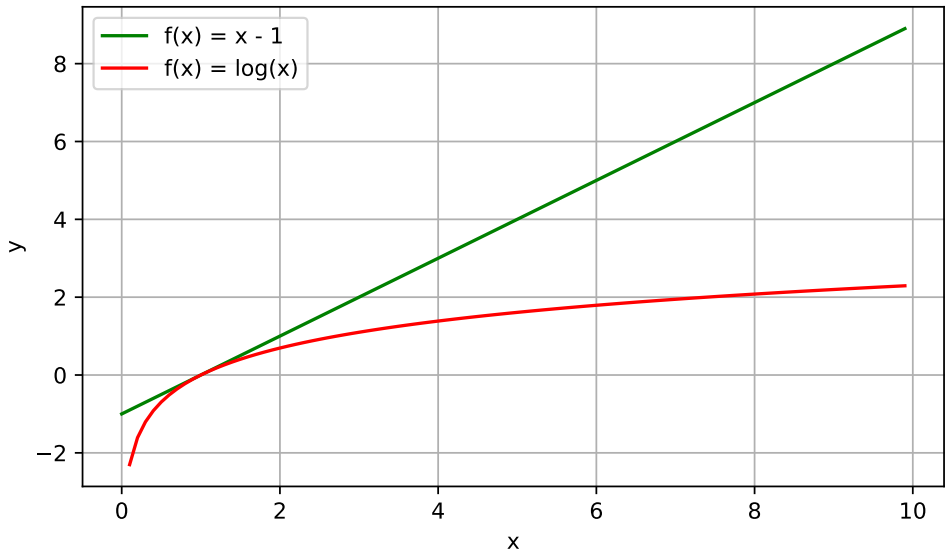


Figure 1:

international engagement, and cultural interaction. Fulbrighters seek to create a more peaceful and prosperous world. Funding is available for research and/or lecturing.

Benefits of studying in the USA:

- Quality of research
- Flexible and interdisciplinary
- International experience
- Funding

Fulbright postgraduate scholarships (http://www.fulbright.org.uk/going-to-the-usa/postgraduate): any master's/doctoral degress at any graduate programme at a US university. The scholarship offers:

- Financial contributions towards the first year of study
- Sickness and accident coverage
- J-1 visa sponsorship and pre-departure orientation
- Support whilst in the USA
- Access to the Fulbright alumni network

The programme aims to foster cultural understanding between the US and UK - little previous experience with the USA is desirable. Awards available:

- Brown University: PhD in any discipline.
- The New School: Master's degree in any discipline.
- Lloyd's of London: Master's degree in any discipline related to risk.
- University of South Florida: PhD in any discipline.
- Fulbright: Master's/PhD/research at any university.
- Elsevier Data Analytics: Postgraduate study requiring data analysis or research at any university.

"We are looking for applicants who can engage with the USA in an atmosphere of openness, academic integrity and intellectual freedom, thereby promoting mutual understanding." Desirable applicants would be able to demonstrate:

- Strong academic proposals and/or research projects
- Ambassadorial skills
- Intercultural sensitivity
- Genuine desire to learn aspects of American culture
- Extracurricular activities
- Community involvement
- Leadership potential
- Plans to further the Fulbright mission and give back to the UK upon returning

Applications process:

1. Read the UK-specific instructions.
2. Applications open: link on the bottom of this page (http://www.fulbright.org.uk/going-to-the-usa/postgraduate/fulbright-postgraduate-scholarships/how-to-apply). This requires bio information, academic/professional accomplishments, Fulbright project plans, personal statement, details of the US institutions that you intend to apply to, and three reference nominations.
3. Upload the US-UK Fulbright Commission Form to this applications.
4. Upload other information (passport, CV, transcripts, US university admission letter, portfolio)

Applications to US universities must be made separately and in good time. If shortlisted, the interview will be in January or February. Status is notified in March. Applications open in August. The British Fulbright Commission provide lots of information regarding

American University's application procedures (see here: http://www.fulbright.org.uk/going-to-the-usa/postgraduate/educationusa-advice).

2 VEST Scholarship

The Charles M. Vest NAE Grand Challenges for Engineering International Scholarship Program provides an opportunity for graduate students at international universities to pursue reserach addressing a global Grand Challenge at leading United States university. The goals of the Vest Scholarships are to provide a platform to exchange ideas, share problem-solving skills and strengthen international relationships in order to advance progress in some of the most critical global challenges in the twenty-first century. The scholarship offers;

- The opportunity to spend a year at a leading US engineering school pursuing research.
- Living and travel expenses and tuition covered by the host institution for the 12-month duration of the scholarship.
- An opportuntiy to perform research in the laboratory of a leading faculty scholar.
- Access to engineering classes and academic credit towards graduate degrees.
- 10 scholarships available - one at each partner institution.

Applicants must be enrolled in a graduate-level engineering program at the time of application and during the scholarship year. Applicants will be asked to submit a personal statement proposing a one-year research and study plan that addresses one of the 14 Grand Challenges for Engineering. Winning statements will be those that presen the most compelling and potentially impactful proposal, and that map best onto Grand Challenge activities at the participating institutions. Academic aptitude and potential for success will be an important part of the selection process. Applications open August 1 and close November 1. The winnders are contacted on February 15, 2018.

The Grand Challenges relevant to me are:

- Engineer the tools of scientific discovery (Caltech, Duke, Washington)
- Advance health informatics (MIT, North Carolina, University of Minnesota)
- Advance personalized learning (MIT, Olin)

It is only possible to apply under one Challenge. Read more about each challenge and institution at `https://vestscholars.org/research-challenges`.

Sunday, 25th June 2017

(08:45 - 09:45) I overslept this morning (it's a Sunday), but I can pick up the hours this afternoon/evening. Yesterday I built part of a factor analyzer, produced detailed notes on the EM algorithm - namely a proof for the bound used and a proof for the choice of density function in the E-step - and read about the Fulbright and Vest scholarships.

Today I would like to:

- Finish the factor analyzer
- Email MM and ET with a review of what I've been up to this week:
  - Reading: generative models of data using hidden variables and the EM algorithm (esp. mixture of gaussians and factor analyzers). Built a univariate MoG model and a factor analyzer for face identification. Read Canny *A Computational Approach to Edge Detection.* Read paper *Detection and classification of surface defects of gun barrels using computer vision and machine learning.* I started a MOOC focused on convex optimization.
  - DNV: IT induction. Gained access to wind turbine and PV panel datasets. Began work on PV panel problem to detect cracked panels. I built several simple classifiers (a random forest, a boosted forest, and a radial basis SVM) against a subset of the data.
  - NW Reading: Bayesian linear, nonlinear, and sparse linear regression models (2 days). Bayesian logistic, nonlinear logistic, and boosting models (2 days). Graphical models (2 days).
  - NW DNV: If Azure access is available, train a convnet against the solar panel dataset. Set up a dummy generative adversarial model for synthetic data generation.
- Read the GAN paper.
- Fix the size of the augmented dataset. Change the programme such that it augments both cracked and non-cracked panels (add Bernoulli variable to speckling).
- Build a HOG feature generator.
- What's SIFT?
- Use `skimage`'s tools to segment the panels.

- Email family describing what you've been up to this week.

There are 7 weeks left as of right now:

- Week 2:
  - Read five papers. (9 hours)
  - Complete two sets of lectures from *Convex Optimization* (12.5 hours).
  - Complete Chapters 8, 9, 10 of *Computer Vision* (25 hours).
  - Complete as much of the TensorFlow Udacity course as you can. (25 hours).
  - Complete and report the PV panel crack detection problem (12.5 hours).
  -
- Week 3:
  - Read five papers. (9 hours)
  - Complete two sets of lectures from *Convex Optimization* (12.5 hours).
  - Complete Chapters 11, 12, 13 of *Computer Vision* (25 hours).
  - Finish the TensorFlow Udacity course. (25 hours).
  - Choose a research problem to solve.
  - Complete and report the PV panel crack detection problem (12.5 hours).
- Week 4:
  - Read five papers. (9 hours)
  - Complete two sets of lectures from *Convex Optimization* (12.5 hours).
  - Complete Chapters 14, 15, 16 of *Computer Vision* (25 hours).
  - Work on the research problem (37.5 hours).
  - Set a reminder to register for NIPS.
  - Set a reminder to start investigating postgraduate degrees at universities (and schedule submission dates, exams and so on).
- Week 5:
  - TBD

(12:30 -14:00) Reviewed yesterday's work, made notes on the applications of the models discussed in Ch.7 of *Computer Vision*. These included regression onto missing data (described in the context of reorienting faces), using t-distributions to model patches of images for object classification, identifying faces using factor analysis, and understanding transformations as hidden variables.

(14:00 - 16:25) Building factor analyzer for face recognition (likelihood fitter)

(18:00 - 21:00) Fixed the fitting part of the factor analyzer (yay)

Hours worked this week:

- Monday: Untracked.
- Tuesday: Untracked.
- Wednesday: 8.5hrs
- Thursday: 11hrs
- Friday: 7.75hrs
- Saturday: 7hrs
- Sunday: 8hrs

1 Prince, Computer Vision (Ch. 7, Modeling Complex Data Densities

Yesterday I finished this chapter by re-covering factor analyzers - generative models that use a subspace to quantify covariance rather than the full feature space - and tying up the loose ends in my understanding of the EM algorithm. The particular loose ends covered were:

- Proof that the bondary function is less than the log-likelihood. That is, we proved that:

$$\sum_{i=1}^{I} \int q_i(h_i) \log \left[\frac{\Pr(h_i, x_i|\theta)}{q_i(h_i)} \right] dh_i \;\leq\; \sum_{i=1}^{I} \log \left[\int q_i(h_i) \frac{\Pr(h_i, x_i|\theta)}{q_i(h_i)} dh_i \right]$$

  Or, in a form that gives a better insight into what's actually going on:

$$\sum_{i=1}^{I} E\left[\log \left[\frac{\Pr(x_i, h_i|\theta)}{q_i(h_i)} \right] \right] \leq \sum_{i=1}^{I} \log \left[E\left[\frac{\Pr(x_i, h_i|\theta}{q_i(h_i)} \right] \right]$$

  The proof for this inequality relies on Jensen's inequality, which states that:

$$E[\log y] \leq \log[Ey]$$

  I haven't proven this, but logarithms tend to squish big numbers, hence shrinking their expectations. I think it's the case that this form of boundary is chosen because it makes it clearer what the the appropriate density functions $q_i(h_i)$ to maximize the boundary are.

- In particular, we proved that the choice of $q_i(h_i)$ in the E-step should be the hidden variable's posterior given the data and parameters at that step:

  $$\hat{q}_i(h_i) = \Pr(h_i|x_i, \theta^{[t]})$$

  We did this by rearranging the optimization problem of the E-step to bring out the Kullback-Leibler divergence:

  $$q_i(h_i) \cdot \log \Big[\frac{\Pr(h_i|x_i, \theta)}{q_i(h_i)}\Big]$$

  which can be shown to always be negative, further meaning that the boundary-maximizing choice of $q_i(h_i)$ is indeed the aforementioned posterior.

- On a high level, I understood why the boundary function is always less than or equal to the log-likelihood of the joint distribution of $\Pr(x_i, h_i|\theta)$, and what choice of $q_i(h_i)$ satisfies the equality of the two for a given choice of model parameters θ. The EM algorithm is useful because it allows nonlinear functions to be optimized, provided that they can be expressed in terms of a hidden variable.

After making notes on this, I started building a factor analyzer to identify faces from images. I wrote out the source code; today I intend to create a functional programme. I think I'll also go through the applications, since these are what have really been allowing me to convert theoretical knowledge to practical utility.

Idea: regressing cracks onto the rest of their panels, then conditioning over an incomplete uncracked panel to generate a cracked panel image (See pp. 104 of *Computer Vision*).

I think I would benefit most from Chapter 7's problems if I were to do them in the middle of next week (say Thursday) rather than right now.

2 Prince, Computer Vision (Ch. 8, Regression models

Figure 1: The mean (left) and single factor (right) of a grayscale image from the factor analyzer.

Monday, 26th June 2017

1 Summary

(08:00 - 11:00) Prince's *Computer Vision*, Ch. 8. Linear regression, Bayesian linear regression, nonlinear regression (inc. kernels, radial basis functions, and gaussian process regression), sparse linear regression.

(11:15 - 13:00) Tensorflow lectures - received first assignment.

(15:15 - 16:45) Sent a weekly update to ET and MM. It was as follows:

> Hello Professor Mirmehdi,
>
> I hope you're well. Paul Harper introduced us several months ago: I'm an undergraduate researcher in machine vision working at the University/DNV-GL. Pleased to meet you (again).
>
> My reason for emailing is that since you are my point of contact with the University and are an expert in machine vision, I thought it would be a good idea if gave you a weekly update of what I've been up to. This is so that:
>
> (a) you know I'm spending my placement time productively
>
> (b) I can benefit from your guidance as to whether I'm following an unpromising path / could make use of some theory/tools in particular.
>
> I appreciate that you are busy and will try to keep these emails succinct - if you'd rather not hear from me, please let me know. Anyway, last week I:
>
> - Spent a lot of time reading about how to use the expectation maximisation algorithm. I built a factor analyser to identify faces in this dataset (http://cswww.essex.ac.uk/mv/allfaces/index.html) and a Gaussian mixture model against another dataset. I know that these are quite old-school methods, but I think understanding them will aid my general grasp of machine learning and vision.
> - Began working on classifying grayscale images of cracked PV panels for DNV. I've cleaned, subset, augmented and split a dataset that was originally of about 50 unique panels. I also ran a random forest and a radial basis function support vector machine against the training set, without any preprocessing (apart from standardisation). I was planning to use a histogram of oriented gradients as the classifier's input (possibly in addition to the pixel intensities?). This is because the cracks seem to be distinguished by their orientation. I may try a convnet, but think it will probably overfit.

- Started a course on numerical methods for convex optimisation. This is in an effort to understand how to phrase optimisation problems in a sensible way. I think the first few weeks focus on theory, after which there's a strong emphasis on applications.
- Became more interested in methods for generating synthetic data. I was thinking of building a generative adversarial network (GAN, a pair of competing neural networks, one of which attempts to generate fake images, the other which tries to spot images that are fake) and looking at using generative models of cracked PV panels to transplant fake cracks onto images of uncracked panels

This week I was going to finish the PV panel classifier and summarise its performance in a report. I have also allocated a big chunk of time to reading, some of which will go towards reading about and building GANs.

Do you have time to meet this week? I was had a few questions about choice of model and preprocessing procedure for the PV panel classification task. Also, your suggestions of what I should read about/work on to make the placement worthwhile would be really useful to me.

Thank you and have a great week,

Jerome

(16:45 - 17:45) Tensorflow assignment.

(17:55 - 18:40) Read Ch. 1 of *Convex optimization*. The course begins by focusing on the maths of convex sets, functions, and optimization, before moving onto applications and algorithms. A convex function is a function that is less than or equal to a line between two points in its

(18:45 - 19:45) Finished the Tensorflow assignment. It focused on data cleaning and fitting a logistic regression classifier. It didn't actually involve using TF! I used the package `tqmd`, which made it easy to associate loading bars with `for` loops. My memory of pickling was also refreshed - it is a pretty convenient way to store large Python structures on disk for access at short notice.

2 Prince, Computer Vision (Ch. 8, Regression models)

After the previous chapter's focus on various generative models, which modeled the data's distribution for given world states, this chapter focuses on discriminative methods for regression. It began by introducing a simple linear model for a univariate continuous world state w and continuous multivariate vector $\mathbf{x}$:

$$\Pr(\mathbf{w}|\mathbf{X}, \phi) = \text{Norm}_w[\mathbf{X}^T\phi, \sigma^2]$$

Where $\mathbf{X}$ is a $D \times I$ matrix of I examples, each of D dimensions. In other words, we assume that the mean response is a linear function of the inputs $\mathbf{x}$, with constant

variance. The model's parameters can be fit using maximum likelihood, maximum a posteriori, or bayesian analysis. The ML estimate of the parameters is:

$$\phi = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}^T\mathbf{w}$$

$$\sigma^2 = \frac{(\mathbf{w} - \mathbf{X}^T\phi)^T(\mathbf{w} - \mathbf{X}^T\phi)}{I}$$

Notice that here the likelihood being maximized is $\Pr(w|x, \theta)$, whereas when fitting the generative methods we maximized the likelihood $\Pr(x|w, \theta)$.

Several flaws in the simple linear model were pointed out:

- There is no embedded representation of the uncertainty associated with extrapolation - the model is as confident where there have been many observations as it is where there are none.
- A linear function may not be representative of the true relationship between the inputs and mean response.
- Many of the input features may be redundant or of little predictive power, but will be assigned tiny coefficients upon fitting the model - this makes the model needlessly complex.

To represent the uncertainty in the model's coefficients more accurately, we can model the parameters as normally distributed variables:

$$\Pr(\phi) = \text{Norm}_\phi[\mathbf{0}, \sigma_p^2]$$

Where σ_p^2 is a hyperparameter (we'd usually fix it to a large value to represent our initial uncertainty). Then, when fitting the model, we can use Bayesian analysis rather than maximum likelihood:

$$\Pr(\phi|\mathbf{X}, \mathbf{w}) = \frac{\Pr(\mathbf{w}|\mathbf{X}, \phi) \cdot \Pr(\phi)}{\Pr(\mathbf{w}|\mathbf{X})}$$

This yields a normal posterior distribution over the parameter values with a covariance that increases with $\mathbf{x}$. To make predictions using this distribution, compute the posterior predictive and evaluate its value at the query world state and data.

Several matrix identities were introduced in this section with the intention of making matrix inverse calculations less expensive. One of these was the Sherman-Morrison-Woodbury identity, which can be used to speed up posterior and posterior predictive inference in the aforementioned Bayesian model.

To work around the issue that the expected response and the inputs are unlikely to have a linear relationship, the model was generalized to nonlinear functions in neat way: simply replace the raw input vector $\mathbf{x}$ with its value under nonlinear transformation:

$$\mathbf{z} = \mathbf{f}[\mathbf{x}]$$

Then model the relationship between the world state and input as a linear one:

$$\Pr(\mathbf{w}|\mathbf{Z}, \phi) = \text{Norm}_{\mathbf{w}}(\mathbf{Z}^T\phi, \sigma^2)$$

Apply the equivalent versions of the ML estimates or the Bayesian analysis methods mentioned earlier to determine the parameters, then evaluate this epression at a new query for inference. The parameters of the nonlinear functions can be defined beforehand, or they can be fit. To fit them, we need to marginalize the above equation over ϕ:

$$\Pr(\mathbf{w}|\mathbf{Z}) = \int \Pr(\mathbf{w}|\mathbf{Z}, \phi) \cdot \Pr(\phi) \cdot d\phi$$

We can then maximize this - the evidence - again the nonlinear function's parameters λ:

$$\hat{\lambda} = \arg\max_{\lambda}[\Pr(\mathbf{w}|\mathbf{Z})]$$

Use a nonlinear optimization procedure (i.e. local search with multiple instantiations, global search) to solve this.

If $\mathbf{z}$ is very high-dimensional, we can replace the dot products $\mathbf{z}_i^T\mathbf{z}_j$ in the ML and Bayesian models with the kernel function $k[\mathbf{x}_i, \mathbf{x}_j]$.

The final area that I began to work on was sparse linear regression, which pushes coefficient values towards zero. By using prior distributions that favour zero-valued coefficients, it's possible to do exactly this.

Tuesday, 27th June 2017

Today I would like to:

- Summarize yesterday's work
- Finish the Ch. 8 of *Computer Vision* (2 hours)
- Build at least one of the regression models described in this chapter (2 hours)
- Spend two hours on the Tensorflow lectures (2 hours)
- Spend four hours on the PV panel classifier: (4 hours)
  - Fix the augmenter balance.
  - Augment undamaged panels.
  - Build a gradient descriptor.
  - Document progress so far.
- Print and read GAN papers

1 Summary

(07:00 - 08:00) Summarized yesterday's work. (08:00 - 11:00) Sparse regression, dual linear regression.

Dual linear regression expresses the gradient vector in terms of the training data points. It's appropriate for situations in which the data has many more dimensions than examples.

Sparse linear regression places a t-distributed prior on each of the gradient values to represent the fact that many of the gradient elements may not be useful in predicting the response. To fit the mean and variance of the parameter vector of the sparse regression model, it is necessary to maximize an approximation of the likelihood function. To make this approximation, a function must be maximized with respect to the hidden variables of the t-distributions. In reality, the likelihood function and the parameters that maximize it are optimized simultaneously.

Relevance vector regression combines the principles of sparse and dual linear regression to produce an efficient regressor: a prior is placed on each element of the dual vector (which correspond to weightings of each observation) that pushes it towards zero, then the mean and variance of this dual vector's posterior is computed while simultaneously approximating the likelihood function by maximizing a function w.r.t. the t-distribution's hidden variables. Relevance vector machines describe predictive models to be described in terms of just a few datapoints, as opposed to attempting to weight high-dimensional input vectors or use all of the datapoints available (which makes optimization expensive).

(11:00 - 13:00) Swept the steps. Took Lunch.

(13:00 -14:00) Bio for The Bakery.

(14:00 - 15:00) More on sparse linear regression. I've found it quite difficult to grasp, and still feel as though I'm missing something. Say we have a linear model:

$$\Pr(w|X, \sigma^2, \phi) = \text{Normal}_w[X^T\phi, \sigma^2 I]$$

X is a $D \times I$ matrix of inputs, σ^2 is common variance, w is a vector of I responses. A non-sparse Bayesian linear model would assume a normal prior distribution $\text{Norm}_\phi[\mathbf{0}, \sigma_p^2]$ over the weights, where σ_p^2 is large. In combination with the normal likelihood above, this would yield a normal posterior distribution:

$$\Pr(\phi|w, X, \sigma^2) = \frac{\Pr(w|X, \phi, \sigma^2) \cdot \Pr(\phi)}{\Pr(w|X, \sigma^2)}$$

Since the posterior and the likelihood were normal, the posterior predictive had a closed-form solution (yet another normal distribution). It was also possible to compute the posterior directly.

In a sparse linear model, the prior term is adjusted to weight zero-valued elements of ϕ more heavily. The purpose of this is to identify input that have little predictive value. This is achieved using a multivariate t-distribution:

$$\Pr(\phi) = \prod_{d=1}^{D} \text{Stud}_{\phi_d}[0, 1, \nu]$$

Problematically, this means that there is no analytical expression for the posterior distribution over the weights ϕ. We still need an expression for this posterior, however, to be able to express the posterior predictive distribution. The book asserts that we can estimate the parameters μ, Σ of a normal distribution that approximates the weight's true posterior by maximizing an approximation to the model's evidence, $\Pr(w|X, \sigma^2)$, with respect to the t-distribution's hidden variables and the model's common variance σ^2. I am puzzled about:

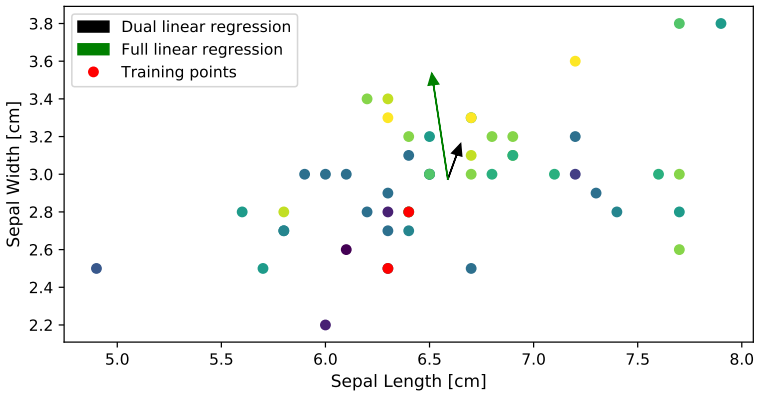
(a) Where the normal posterior came from to approximate the product of a normal and t-distribution.

(b) How the values for the hidden variables influence the choice of weights.

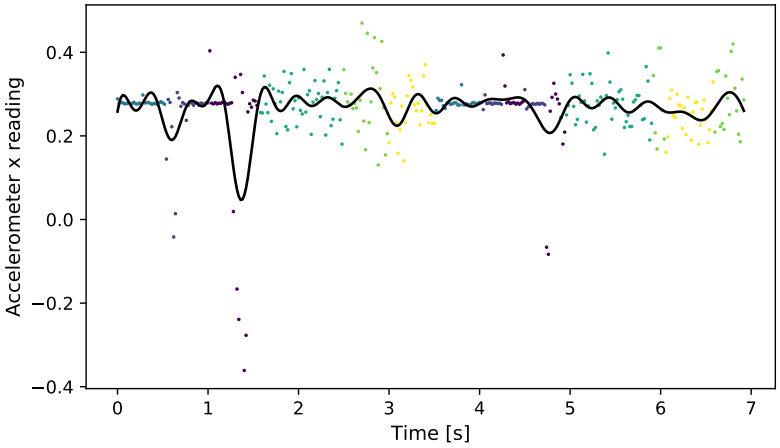
The former point I think I can appreciate in an unrigorous sense, but the latter needs more thought. The hidden variables of the t-distribution affects the distribution's variance - smaller values of h_d suggest a larger variance, further implying a more probable non-zero value for ϕ_d. So the idea with solving for values of the hidden variables is to stretch the prior in the directions that are likely to have non-zero values. By contrast, σ^2 has to be updated alongside the hidden variables because when the uncertainty in the values of the hidden variables changes, so too will the uncertainty in the response. I think I'm getting closer with this - maybe a few more readings...

(15:00 - 16:15) Wrote the above on sparse regression. Replied to John Ryan.

(16:15 - 17:50) Built a dual regression model on Iris dataset.



(17:50 - 19:45) Built a Gaussian nonlinear regression model.



labdayWednesday, 28th June 2017 Yesterday I built a nonlinear regression model and a dual linear regression model.

The nonlinear model has a simple premise - perform a nonlinear transformation of the inputs, then perform a linear regression of the transformed inputs onto the response. Kernels can be used to make this process much more efficient/feasible.

The dual model assumes that the gradient vector is in the subspace described by the observations. It's useful when the dimensionality of the data exceeds the number of observations. In such a situation, linear regression is not possible since its model would be under-constrained. Why is it valid to assume that the gradient vector lies in the subspace described by the observations? In short, the assumption is justified by the observation that if input variation is not observed in a given direction, then there is no information available that can be used to compute a gradient in that direction.

(09:30 - 12:30) *Convex Optimization* Lecture 2 - convex sets.

(13:45 - 17:30) Tensorflow work. Moved from the logistic regression model to neural networks by introducing nonlinear transfer functions (specifically, RELUs). Built a two-layer network for 10-way classification. Understood the TensorFlow workflow: set up a graph, pass the graph to a session. Reviewed stochastic and minibatch gradient descent, momentum (taking a running average of the loss gradient and using it to choose the direction of optimization for the current iteration), l2 regularization (penalizing large weights - why?), dropout (to encourage redundancy in the network), learning rate decay (to permit better overall convergence), and early termination (to avoid overfitting).

(19:00 - 21:15) Prince, *Computer Vision* (Ch. 9, Classification models). Logistic regression, Newton's method for making an ML fit of its parameters. Laplace approximation for a distribution. Integrating a multivariate normal distribution over one dimension instead of many via a linear projection. Highlighted the weaknesses of logistic regression:

- Using maximum likelihood to fit the parameters whitewashes the actual uncertainty in a class prediction (resolved via Bayesian logistic regression).
- Linear decision boundaries are sometimes inadequate.
- The model is expensive to fit in many dimensions, and is prone to overfitting in wide situations.

2 Boyd, Convex Optimisation (Lecture 2)

The lecture covered various basic types of convex sets, and operations that preserve convexity. Simple convex sets that he brought up included:

- The basic convex set: $S : \{x_1, x_2 | \theta x_1 + (1 - \theta)x_2 \in S, \theta \in [0, 1]\}$

- Affine sets: $\{x|\theta x_1 + (1-\theta)x_2, \theta \in \mathbb{R}\}$
- Hyperplanes: constraints on single elements from an affine set; $\{x|a^t x = b, a \in \mathbb{R}^n, b \in \mathbb{R}\}$
- Convex hulls: the set of all convex combinations of points in a set: $x = \theta_1 x_1 + ... + \theta_k x_k$ with $\sum \theta = 1, \theta_i \geq 0 \ \forall \ i$.
- Halfspaces: like a hyperplane, but with an inequality. $\{x|a^T x \leq b\}$.
- Convex cones: any linear combination of two points, provided that the coefficients are positive. $C : \{x|x = \theta_1 x_1 + \theta_2 x_2, \quad \theta_1, \theta_2 \geq 0\}$
- Euclidean balls and ellipsoids (linear transformations of a unit ball). $\{x|(x - x_c)P^{-1}(x - x_c) \leq 1\} = \{x_c + `0Au|\|u\|_2 \leq 1\}$. A euclidean ball is the set of points that satisfy $\{x|\|x - x_c\|_2 \leq r\}$. We also defined a norm as a function that satisfies various conditions.
- Polyhedra: A set of points satisfying finitely many inequalities (halfspaces) and equalities (hyperplanes). $\{x|Ax \preceq b, Cx = d\}$.

He emphasized that these basic convex sets in combination with convexity-preserving operations are useful for demonstrating a set's convexity. Alternatively, we can use the basic definition of convexity to show that a set is convex.

The second half of the lecture focused on these convexity-preserving operations, which were the:

- Intersection
- Affine function
- Perspective function - the division of the first n elements in a vector by the vector's nth and final element.

$$P(x,t) = \frac{x}{t}, \quad \text{dom}P = \{(x,t)|t > 0\}$$

- Linear-fractional function - the perspective function after an affine transformation.

$$f(x) = \frac{Ax + b}{c^T x + d}, \quad \text{dom}f = \{x|c^T x + d > 0\}$$

Thursday, 29th June 2016

(07:10 - 12:50) PV panel classifier work

(14:00 - 15:20)(16:45 - 18:45) Computer Vision exercises (Ch. 5,). Distribution of a multivariate normal r.v. under an affine transformation. Marginalizing a normal distribution using an affine transform. Standardizing a multivariate normal distribution. Proving the Schur complement identity. (Up to 5.5) (1, 2, 3, 4, 7, 8).

(18:45 - 20:45) Computer Vision exercises (Ch. 7) Describing generative models for a continuous input vector and a binary response. Described how the t-distribution can be used to make this type of model robust to outliers (e.g. mislabelled samples). Derived the component weight updates in the M-step of the EM algorithm when fitting a Gaussian mixture model. Learnt how to use Lagrange multipliers to solve constrained optimization problems (tangency between the constraint and objective function gradients!).

Friday, 30th June 2016

(09:00 - 11:00) Solving problems from Ch. 8 of *Computer Vision*. Deriving the t-distribution from the gamma and normal distributions. Deriving the M-step values for the mean and covariance values in a Gaussian mixture model. (11:00 - 13:00) Nonlinear logistic regression - as with linear regression, we simply apply the nonlinear transformation to the inputs. Fitting the model requires a nonlinear optimization method - fitting these models is usually done using maximum likelihood, since prior modeling over both the gradient vector ϕ and the parameters of the nonlinear function requires a lot of effort.

I also made notes on the dual logistic regression classifier, and how it can be fit using either maximum likelihood orBayesian inference. The latter requires the Laplace approximation - approximating a distribution using a normal distribution with equal mean and second derivative at the mean to the actual distribution being modeled. Fitting logistic regression or dual logistic regression models in a Bayesian manner requires that the true parameter (ϕ or φ) posterior distribution's parameters be approximated by their equivalents in the maximum a posteriori fit.

This afternoon I intend to:

- Finish Ch. 9 of *Computer Vision*
- Build the Generative Adversarial Network described by Analytics Vidyha
- Build a kernelized dual logistic regression classifier.
- Finish the *Convex Optimization* lecture.
- Find papers relating to crack detection.
- Complete IET end-of-year assessment.

(14:40 - 18:40) Chapter 8 of *Computer Vision*. A lot of material was covered:

- Nonlinear logistic regression
- Dual logistic regression
- Kernelized Bayesian nonlinear dual logistic regression (a.k.a. Gaussian process classification)
- Relevance vector classifiers (sparse Gaussian process classifiers)
- Forward stepwise selection

- Boosting
- Classification trees/gating functions
- Multiclass logistic regression

I should definitely type up a review of these areas - if not this evening, then tomorrow.

(18:40 - 20:05) Labelled images (annotations, not y/n).

(20:05 - 20:45) Resized, masked images.

1 Meeting with MM

Topics of conversation:

- PV panel classification problem : identifying cracks in scans of PV panels.
- Questions:
  - Very small dataset problem: 50:50 train/test, split the training data 70:30 into training/validation sets (i.e. I have very poor resolution on what the generalization error of the models I'm fitting is likely to be).
  - What preprocessing can I use to enhance the cracks?
    - The images have been equalized, resized for intensity classification.
    - Augmented the training images to expand the dataset size.
    - Did not augment testing images.
    - I've used HOGs, which resulted in improved performance (with more data, I think this would be a viable solution).
    - Other edge/feature detectors?
    - Possible idea: Cluster panels according to model, subtract an undamaged panel from the input?
  - Are there any papers that he would recommend?
  - Other methods for expanding the amount of data available?
  - Wind turbines:
    - Meeting the inspections team next week
    - ET wants me to focus on using visible spectrum data
    - Use stills, as opposed to sequenced videos

    - Need to label the data - recommendations on how to do this?
    - I will email him images of the turbines
- General placement enquiries:
    - What can I do that would be useful/what problems would be tractable for the placement, but would still be interesting/novel?
    - Does he think that doing an (admittedly small) piece of original research is possible in this timeframe?
    - I'm interested in synthetic data generation - does this sound like a suitable area to focus my efforts?

Monday, 3rd July 2017

(06:30 - 11:10) PV Panel subsetter -...

(11:30 - 12:10) Meeting with MM. Notes:

- Local binary patterns
- Gabor filters
-

(12:45 - 13:15) Lunch with Paul Harper.

(13:45 - 18:00) DNV.

(21:00 - 22:00) TF Reg. assignment.

Tuesday, 4th July 2017

(07:30 - 11:00) Finished Tensorflow assigment. Emailed Mikkel Skou requesting Azure access. (11:30 - 12:00) Skype call with ET and RD w.r.t. Azure access. Learnt how to save/restore sessions in TensorFlow. (12:15 - 13:15) Conceived of probabilistic relaxation method. Figuring out details. (13:45 - 15:00) Meeting with James and ET at DNV w.r.t. image processing. They recommend vastly expanding the feature space, possibly focusing on cracks located at the edges. (15:00 - 22:00) Meeting family. No work.

Wednesday, 5th July 2017

Today I plan to:

- Build the probabilistic relaxation model.
- Generate at least ten sets of candidate extracted features.
- Build and report on the progress with the PV panel classification problem so far.
- Read at least two papers.
- Finish *Convex Optimisation* lecture 2.
- Understand the singular value decomposition.
- Summarize last weeks work (or lack thereof!)

Stay hungry, stay foolish.

(06:20 - 19:15) Built a generative normal model of crack/uncracked pixels. Fit a random forest to the resulting probabilities. Built lots of filters. (19:15 - 20:30) Printed three papers at the University - Hough transform, probabilistic relaxation, and hidden markov models for crack detection. (20:30 - 22:30) Read the Hough transform paper. Will summarize tomorrow when I wake up.

Thursday, 6th July 2017

Good morning - it's 06:50. Yesterday I experimented with various convolutional filters, some of which were moderately successful. I also read a paper on the Hough ('how') transform, *Use of the Hough Transformation to Detect Lines and Curves in Pictures* by Ruchard Duda and Peter Hart. The Hough transform is used to detect colinear points in an image.

1 Use of the Hough Transform to Detect Lines and Curves in Pictures

The idea of the Hough transform is to map a point at (x, y) in the figure domain onto a plane parameterized by (ρ, θ). θ corresponds to the angle between the plane and the origin and ρ is the distance of the plane from said origin. The unit normal vector of the plane is $[\cos\theta, \sin\theta]$, such that the points (x, y) are projected onto a line at an angle θ to the origin by:

$$\rho = x\cos\theta + y\sin\theta$$

. If θ and are ρ are chosen correctly, then a colinear set of points in the figure will be projected onto a single point in the $\rho - \theta$ plane. In this way then, ρ encodes the detected line's distance from the origin and θ the angle it makes with the y-axis. For the implementation:

1. Binarize the image.
2. Create a matrix with rows indexed by quantized values of $\rho \in [0, R]$ and columns indexed by quantized values $\theta \in [0, \pi)$.
3. Project the points onto a vector that makes an angle $\theta \in [0, \pi)$ with the origin, then count the number of projected points that lie in each radial interval. Record this number in the matrix cells corresponding to the given angle and radial intervals.
4. Repeat (3) for all angles. The sum of each column will be equal to the total number of points.
5. Cells that have an unusually high number of points will tend to correspond to lines in the image.

The Hough transform comes with a few warnings attached - high-scoring cells don't necessarily correspond to contiguous lines, the chosen resolution of cells has big implications

for the method's efficacy, and careful thought needs to be given about the number of pixels in the lines of interest. The method can be generalized to other parameterizations beyond straight lines: All that's necessary is to parameterize a curve using the x, y values. In general, points in (x, y) on the geometry described by a function $f(\phi_1, ..., \phi_p)$ will map to single point in the space $\phi_1, ..., \phi_p$. By discretizing the values of ϕ and binning the positions of the points, it is possible to detect an arbitrary geometry. A parabola in (x, y), for example, would be parameterized by:

$$(x - a)^2 + y^2 = b$$

Worked on filters until 12:00.

2 Relaxation labelling algorithms - a review, Kittler and Illingworth

Relaxation labeling attempts to assign labels (discrete or probabilistic) to nodes a_i in a graph based on measurements of those nodes X_i and information about the surrounding nodes' labels. The set of possible labels each node are membes of the set $\Lambda = \{\lambda_1, ..., \lambda_m\}$. The contextual information conveyed by one node's label about another node's label is expressed using compatibility coefficients:

$$r(\theta_i = \lambda_l, \theta_j = \lambda_k)$$

These coefficients are weighted by the neighbouring node's label probabilites to form a scoring function for a particular label at a given node. Assuming initial probabilities that the nodes have a given label, $\Pr^s(\theta_j = \lambda_k)$:

$$q_j^s(\theta_i = \lambda_l) = \sum_{k=1}^{m} r(\theta_i = \lambda_l, \theta_j = \lambda_k) \cdot \Pr^s(\theta_j = \lambda_k)$$

This expression is known as the support provided by node a_j in favour of assigning the label λ_l to node a_i- it is the sum of the compatibilities of the possible labels for a_j with the label λ_l for a_i, with each term weighted by the corresponding probability for a_j's label.

The overall support in favour of assigning the label λ_l to node a_i on step s is the sum of its support over the entire network. Because some nodes - such as those closer to a_i - convey more information about what a_i's label is, the support provided by each network is weighted by coefficients C_{ij}:

$$Q^s(\theta_i = \lambda_l) = \sum_{j=1}^{N} C_{ij} \sum_{k=1}^{m} r(\theta_i = \lambda_l, \theta_j = \lambda_k) \cdot \Pr^s(\theta_j = \lambda_k)$$

The coefficients sum to 1.

To update the proability that a node has a particular label, we consider what fraction of the overall support is attributable to the label:

$$\Pr^{s+1}(\theta_i = \lambda_l) = \Pr^{s}(\theta_i = \lambda_l) \cdot \frac{Q^s(\theta_i = \lambda_l)}{\sum_{k=1}^{m} \Pr^{s}(\theta_i = \lambda_k) \cdot Q^s(\theta_i = \lambda_k)}$$

The initialization probabilities for this framework can be estimated according to a classifier of the form:

$$\Pr^{0}(\theta_i = \lambda_l) = \Pr(\theta_i = \lambda_l | X_i)$$

It seems that most controversy lies in the choice of compatibility function $r(\theta_i = \lambda_l, \theta_j = \lambda_k)$ and the support function $q_j(\theta_i = \lambda_l)$. Supposedly its preferable to use the geometric mean as opposed to the expectation of the compatiblities. I.e.:

$$q_j(\theta_i = \lambda_l) =$$

Friday, 7th July 2017

I spent the morning (07:00 - 09:00) setting up AWS instances and becoming familiar with its console. From 10:10 I was in Queens working on the PV Panel assignment. Specifically, I was writing up its report. Doing so spurred me to relabel the training data. From 14:10 onwards I read about various ways of representing images. I read a paper on the Gabor filter and a paper on detecting cracks in eggs. The cracked-egg detector was quite simple, but (apparently) effective - it consisted of a subtraction, segmentation, then contour detection. This evening I plan to build the ConvNet for the TensorFlow assignment. I did indeed, and will write about it tomorrow morning.

Saturday, 8th July 2017

Registered for British Machine Vision conference 2017. The workshop is being held from the 4th until the 7th of September in London. The workshops are on:

- Activity monitoring by Multiple Distributed Sensing (Pier Mazzeo, Paolo Spagnolo)
- Deep Learning on Irregular Domains (Xianghua Xie, Michael Edward)
- Lip-reading using Deep Learning Methods (Themos Stafylakis)
- Automatic Face Analytics for Human Behaviour Understanding (Xiaohua Huang)

The keynote speakers are Richard Szeliski (Facebook) and Pietro Perona (Caltech). Tutorial speakers are Michael Bronstein, Lourdes Agapito, Shimon Whiteson, Andreas Geiger. Workshop keynot is by Jamie Shotton from the Hololens team at Microsoft.

(08:00 - 10:00) Working on TensorFlow convnet. Trialled a P2 AWS instance.

(10:30 - 12:10) CVX lecture 2 notes. Proper cones, generalized inequalities, separating and supporting hyperplanes, minimum and minimal points in a set, dual cones.

Configuring an instance with Jupyter notebooks.

- Create a config profile `jupter notebook --generate-config`
- Create a security key in a sensible location (i.e. a directory) using

  ```
  sudo openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout
  mycert.pem -out mycert.pem
  ```

- Go to `/.jupyter` and edit the config file.

  ```
  c = get_config() # Kernel configuration object

  c.IPKernelApp.pylab = 'inline'

  c.NotebookApp.certfile = u'/home/ubuntu/certs/mycert.pem'

  c.NotebookApp.ip = '*'

  c.NotebookApp.open_browser = False

  c.NotebookApp.port = 8888 # fixed, known port
  ```

- Make sure the instance's security group allows TCP access to port 8888.

- Connect to the instance at the the server's DNS path using HTTPS, with the port specified at the URL's end after a colon.
- Get the token from the instance via `jupyter notebook list`.
- Use the shell's `screen` command to create multiple shell screens before launching the notebook.

(12:30 - 17:20) Building PV panel convnet and adding it to the report.

Instance setup:

1. Launch Ubuntu P2 instance (i.e. an instance with a Nvidia card).
2. Install Anaconda 3+
3. Install CUDA Toolkit and drivers:
    - Install gcc (GNU compiler collection, the compiler that CUDA uses)
    - Install kernel headers and development packages for the running version of the kernel (these are required by the CUDA Driver).
    - The CUDA Toolkit can be installed using either a [Linux] distribution-specific package, or a [Linux] distribution-independent package. Download and install the Nvidia CUDA Toolkit - this contains the CUDA driver and tools needed to create, build and run a CUDA application. USE THE .DEB INSTALL - THE OTHER ONE GAVE US TROUBLE. This is installed via the package manager, not the runfile.
    - Post-installation mandatory actions: add the CUDA `bin` directory to the `PATH` variable.
    - To verify the installation, install the samples with write permissions using the convenience script.
    - `nvcc` command compiles CUDA programmes. It calls `gcc` in the process. `nvcc -V` checks the CUDA Toolkit version.
    - Calling `make` in the same directory as the CUDA samples compiles them using nvcc. The resulting executables are in the `bin` file. To run these, use `./<exe>`. Use `./deviceQuery` after compilation to check that the Nvidia GPU is recognised by CUDA.
4. Install the NVIDIA cuDNN library. Download the zip file locally, then upload it to the instance using SCP. Extract and copy its contents into the CUDA install directory as per these instructions (https://askubuntu.com/questions/767269/how-can-i-install-cudnn-on-ubuntu-16-04).
5. Install libcupti-dev (NVIDIA CUDA Profile Tools Interface).
6. Create a conda environment for tensorflow (`conda create -n tensorflow`).

7. Activate the environment `source activate tensorflow`.
8. Install the appropriate version of TensorFlow using `pip install --ignore-installed --upgrade tfBinaryURL` where tfBinaryURL is the link for the suitable TF version for the Python version and GPU/CPU mode.
9. Validate the TensorFlow installation.
10. Check where CUDA installed using `which nvcc`
11. Add PATH variables to `/.bashrc` as per `https://stackoverflow.com/questions/41991101/importerro`
12. Run `python` in the shell and `import tensorflow as tf`.

Installing software on Ubuntu - use `sudo dpkg -i /path/to/deb/`.

I also learnt how to attach an EBS volume to an AWS instance. `lsblk` lists the available disks and where they are mounted. Format the filesystem using `sudo file -s /dev/xvdb`. To mount the volume to a specific directory, use `sudo mount /dev/xvdb /<dir>`. `df` shows the storage available. Note that on reboot EBS volumes are unmounted.

(19:00 - 21:00) ResNet paper. Essential idea: preventing degradation in really deep nets by fitting stacks of layers to a residual of the target mapping then summing with the stack input. The idea is that

- If the previous layers have already approximated the overall target mapping then the surplus layers can easily push their outputs to zero, leaving only the identity to be passed forward.
- Layers will tend to generate filters that are closer to an identity mapping than a zero mapping. Using residual layers initializes them closer to their targets.

i.e. rather than modeling a target mapping $\mathcal{H}(\mathbf{x})$, we model the mapping $\mathcal{F}(\mathbf{x}) = \mathcal{H}(\mathbf{x}) - \mathbf{x}$ (i.e. the residual mapping). The identity mapping is $I : I(\mathbf{x}) = \mathbf{x}$, where $\mathbf{x}$ is the input to the stack of layers. The paper was by a team from Microsoft's research division.

Spaced repetition came up this afternoon during random reading - 3, 10, 30, 60 days.

Tomorrow I will:

- Build the PV panel convnet I planned to build today.
- Review the chapters of Prince, *Computer Vision* that I have been through so far.
- Complete the homework from *Convex Optimization*.

Sunday, 9th July 2017

. I finished reading the ResNet paper, *Deep Residual Learning for Image Recognition*, over breakfast. They trialled a 1000+ layer network against the CIFAR-10 dataset. Their ultimate best performance was achieved using an ensemble of six networks, each of different depths. To analyze the network, the team charted the standard deviation of each layer's responses (assumedly across all images in the training set). They claimed that the smaller responses supported their hypothesis that the residual layers would tend to be closer to zero than non-residual responses.

There were several papers referenced that caugt my eye:

- *Understanding the difficulty of training deep feedforward neural networks*, Bengio et. al (Vanishing gradients and normalized initialization).
- *Maxout networks*, Bengio, Goodfellow et. al (Maxout).
- *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.*, He and Zhang (Weight initialization).
- *Batch normalization: Accelerating deep network training by reducing internal covariate shift.*, Ioffe and Szegedy (Batch normalization).
- *Product quantization for nearest-neighbor search*, Schmid, Douze, Jegou. (Vector quantization)
- *Imagenet classifiation with deep conolutional neural networks*, Krizhevsky, Sutskever, Hinton. (AlexNet)
- *Very deep convolutional networks for large-scale image recognition*, Simonyan and Zisserman (VGG)
- *Visualizing and understanding convolutional neural networks*, Zeiler and Fergus.

(08:45 - 14:00) Built a data wrangler for the convnet.

(15:50 - 17:50) PV panel base convnet.

VOICE TRANSCRIPT

Sunday 9th of July 2017

I woke up at 8 o'clock. Over breakfast, I read a paper on the inception module. The key idea of the inception module is to perform multiple convolutions on each layer with the

convolution windows being slightly different sizes. the idea of this method is to provide multiple levels of abstraction per layer.

After breakfast I worked on the PV panel classifier. The method I was using was a convolutional neural network. It was a simple architecture of just two layers. It performed very well, achieving 85

In the late afternoon I printed off the convex optimisation notes from Stephen Boyd course and solved two practice problems. The first problem involved proving that a linear combination of points in a set can be can be shown to be in the set purely by virtue of the sets convexity.

I received an email from Paul Harper in the afternoon inviting me to an energy symposium on Wednesday. He also mentioned an Startup launch by a previous engineering design student. the Startup name was anon.ai. The student's name was Harry Reed. Incidentally, he studied machine learning at Stanford.

This week's I intend to build more convolutional neural networks read more papers and implement the probabilistic smoothing model I've read about previously. I would also like to review the work I done on Princes book already and watch the entirety of lectures three of the convex optimisation course.

Tomorrow I need to:

- Email Liz Traiger updating her with what I've been up to.
- Email Grandma and Grandad letting me know what I've been up to.
- Complete the convex optimisation homework.
- Review the Prince work I have done so far: probability distributions, fitting models, generative models, discriminative regression models, discriminative classification models. This is in preparation to move onto graphical models.
- Build a more sophisticated convolutional neural network (i.e. implement residual layers and inception modules, possibly including more filters at the input).
- Test the network's performance on a full image.
- Update the PV panel report to reflect this additional work.
- Go swimming at 10:00 for one hour (count the lanes).
- Complete the next set of Tensorflow lectures.

Monday, 10th July 2017

(07:20 - 10:30)TensorFlow work (11:30 - 13:30) Reporting

(14:30 - 17:10) CVX homework questions 1 - 3. Deriving the distance between hyperplanes, the Voronoi description of a halfspace, that convexity of slabs, rectangles, wedges, and the set of points closer to a given point than a given set.

(17:10 -19:30) CVX lectures (week 3) - convex functions. First and second order conditions for convexity. Relating convex functions to convex sets via the epigraph. The α-sublevel set of a convex function is a conve set.Jensen's inequality $f(Ez) \leq Ef(z)$. Checking the convexity of a function by checking its convexity when restricted to a line. Cleaning up notation by using the extended-value extension of a function. Strict convexity.

Tuesday, 11th July 2017

(07:00 - 08:00) Reading about bootstrapping. Sketched out a layout for an image classification testing utility.

(08:00 - 12:45) Building `pymv` testing utility. (13:50 - 14:30) Convex Optimization homework. Evaluating the convexity of sets that intersect the probability simplex. (14:30 - 17:10) Convex optimization homework (finished proving the convexity of sets that intersect the probability simplex). Made notes on operations that preserve convexity when applied to convex functions.

Wednesday, 12th July 2017

(07:20 - 09:00) Spatial transform networks paper. Learning an image transformation to a canonical form that subsequently enhances classification accuracy. (10:00 - 13:30) Prince, *Computer Vision*. Reviewing the chapters covered so far. Bernoulli, beta, categorical, Dirichlet, normal, normal inverse Wishart distributions and their associated conjugacy. Generative models (modelling the contingency of the data on the world state) and discriminative models (modelling the contingency of the world state on the data). In other words, generative models demand parameterization of a function describing the data's likelihood, given the world state, and selection of an appropriate prior for the world state. They are generally more complex than discriminative models because it's expensive to compute the posterior and [therefore the] posterior predictive. Discriminative models, on the other hand, model the posterior - i.e. the world state, given the data - directly. They're much cheaper to learn and use, however they don't offer a simple means of incorporating prior information. Naive Bayes is a form of generative model that makes two distinguishing assumptions:

- The elements of the data vector are independent of one another, conditional on the world state.
- The elements of the data vector can be modelled using categorical distributions.

To provide an example, consider the possibility that the data vector is $[x_r, x_g, x_b]$, where x_i denotes the ith channel's pixel intensity value (an integer between 0 and 255). For each world state, we fit categorical distributions to each element to reflect the relative probabilities of particular pixel intensities. At inference, we use a prior that accounts for the base rate of each world state to infer the most probable world state. Notice that the categorical distributions over each pixel's intensities implicitly assume that the other channels do not provide useful information about that channel's intensity value.

(18:00 - 20:00) Prince, *Computer Vision*. Using hidden variables to design better generative models. Chapter 7 concerned modelling complex data densities. It begins by highlighting the weaknesses in modelling the distribution of multivariate data using a normal distribution:

- The distribution's parameters are very sensitive to outlier datapoints.
- Many parameters must be estimated on account of the covariance matrix.
- The distribution is unimodal - if observations for a given world state cluster around more than one value of a variable, the normal model will fare poorly.

It also introduced hidden variables. By having a variable h that acts to select one of many possible distributions to generate a given datapoint, we can combine distributions to form an overall distribution in x that is more expressive. In notation,

$$\Pr(x|\theta) = \int \Pr(x, h|\theta) dh$$

Imagine drawing h from its distribution, then drawing the distribution in x at that value of h. The additional source of variation is used to weight many possible distributions in x. When we marginalize over h - i.e. aggregate over it - these many possible distributions in x form a weighted combination.

But how to fit such a model? The distributions in x must be parameterized at each possible value of h, and the distributions at each value of h need to be weighted so as to line up with the data suggests. Weighting the distributions in x is a matter of apportioning probabilities to the values of h: more probable h's see their distributions in x realized more frequently. However, there is a problem - it isn't possible to find the optimal distribution over h without knowing what the distribution over x is at each value of h. Additionally, we do not know which distribution in x associated with each h value generated the observations x_i. To get circumvent this issue, we adopt a strategy of alternating between updating the marginal distribution over h and updating the model parameters at each value of h, θ. This algorithm is known as the expectation maximization (EM) algorithm.

The EM algorithm has two steps:

1. E-step: In the E-step we update the estimates for the values of h_i that were realized in drawing the observations x_i. Since each observation x_i is assumed to be generated from a distribution associated with one value of h, this means that we update I estimates of the values of h from which x_i was probably drawn. We denote these distributions over the plausible values of h as $q_i(h_i)$.

2. M-step: The M-step acts to adjust the parameters of the probability distributions associated with each value of h. We denote these parameters θ. These parameters are adjusted such that they maximize the bound on the log-likelihood.

The book specified three models that can be interpreted in terms of hidden variables. These were:

- Gaussian mixture models. Here a discrete hidden variable determines which of several Gaussians to draw an observation from. As such, weighting certain values of the hidden variable more heavily results in their associated Gaussians being responsible for more of the data generated.

- A t-distribution. A continuous hidden variable is drawn from a gamma distribution and is then used to scale the variance of a normal distribution. In other words, at each value of h, there is a normal distribution with a unique variance. The marginal distribution of these values of h is a gamma distribution. Marginalizing

over h results in the t-distribution. As it happens, the distribution over h at each value of x is also a gamma distribution, so when we update our estimates of the possible values for each h_i, we end up modifying the parameters for the gamma distributions at each x_i.

- A factor analyzer. Factor analyzers model covariance in a linear subspace of the data. They can be understood as being constructed by:
    1. Sampling a K-dimensional vector $\mathbf{h}$ from a multivariate standard normal with diagonal covariance.
    2. Sampling $\mathbf{x}$ from a D-dimensional multivariate normal distribution located at $\mu + \phi\mathbf{h}$, where the columns of ϕ are vectors that span the K directions in which the data varies most.

In other words, we assume that $\mathbf{x}$ has been sampled from a normal distribution with uncorrelated components, but that this normal distribution is jiggled around in the directions in which covariance is largest. At each value of h, there is a normal distribution. h is also normally distributed. Consequently, factor analyzers are effectively a continuous equivalent of a mixture of gaussians model.

(20:00 - 21:00) Coding testing utility.

Thursday, 13th July 2017

(07:45 - 08:15) Read the introduction to *Deep Learning*. (08:40 - 10:40) Continued a review of the chapters of Prince, *Computer Vision*, that I have read so far. (10:45 - 11:30) Researching research groups.

- Applications for Autonomous Systems open on the 1st of September. (Autonomous and Intelligent Machines and Systems)

(11:30 - 12:10) Replied to an email from dad. (12:10 - 20:30) Convolutional networks for cracked panel detection. Wrestling with TensorFlow.

Friday, 14th July 2017

(08:30 - 11:00, 12:00 - 15:00) Learned how to use TensorBoard (`tf.name_scope`, `tf.summary.scalar`, `tf.summary.FileWriter`, `tensorboard --logdir=<FileWriterOutputdir>`, right-aligning formatted text in Python, accessing ports on a private IP vs. accessing ports through a public IP, using `vim`). I think I'm getting it (slowwwwllllyyyyy). Inadvertently overwrote data in git (arghhh).

Saturday, 15th July 2017

(08:05 - 12:20) Prince, *Computer Vision*. Reviewed dual linear regression, attempted to unravel sparse linear regression (I can't wrap my head the approximation he uses for the integral of the product of a t-distribution and a normal distribution), tied them together (somewhat) in relevance vector regression. Began Chapter 10, which is on graphical models. Keywords that I should flesh out later:

- Directed graphical models
- Undirected graphical models
- Products of experts
- Cliques
- Cliquey undirected graphical models (i.e. Markov random fields)
- Hidden Markov models
- Markov trees
- Kalman filter
- Conditional independence
- Potential and cost functions (and their normalizing constant - the partition function)

(14:10 - 16:10) Learnt how to branch in git, then refactored TensorFlow training API. Need to add validation performance summary to graph, then add image augmenter. After that, start working on fancier models. Tomorrow I really need to update what I've been working on. I also need to bump accuracy scores to the graph rather than in the Tester. Additionally, I need to make it possible to pass in an image and have the program return a heatmap of crack/nocrack probabilities.

Branching in Git: `git checkout -b <branchname>`. This creates a new pointer which moves with the commits. When you switch back to another branch, Git will reset the working directory to look like it did last time you committed on that branch. To switch to another branch, use `git checkout <branchname>`. After making the edits on the new branch, switch back to the master branch and then merge the new branch using `git merge <branchname>`. If the branch isn't divergent (i.e. the master is reachable by following the branch's history), then all merging will do is move the master to the end of the branch - this is called fast-forwarding. After merging a branch, we can delete

it using `git branch -d <branchname>`. If you create a branch then move the `master` pointer, later merges will use a slightly different strategy. Instead of moving the `master` pointer along the branch, they instead find the `master` and branch's common ancestor, then merge automatically, creating a new commit in the process. If you've edited the same part of the same file in both the `master` and the branch, then there will be a merge conflict - but don't freak out! No commit has been created yet, and calling `git status` will show you which file is causing the conflict. Git adds conflict markers to the offending files, so it's easy to see what's causing problems. Remove above/below the `=======`, save the file, then run `git add <filename>` to tell Git that the conflict has been resolved. Commit the change and the branch will be merged! Huzzah!

(17:05 - 19:30) Migrating accuracy calculations to graph.

Sunday, 16th July 2017

(09:00 - 13:00) Prince, *Computer Vision*. Chapter 10: Graphical models. Various types of graphical models: the hidden Markov model, Markov tree, Markov random field, and Kalman filter. Using conditional independence to simplify models with many parameters. Maximum a posteriori for the world state - approximating this distribution through sampling. Sampling from directed graphical models (ancestral sampling). Sampling from undirected graphical models (Markov chain Monte Carlo methods, specifically Gibbs sampling) - relevant as it can be used to build up an approximation to a posterior distribution: $\Pr(w_{1\ldots N}|\mathbf{x}) \propto \Pr(\mathbf{x}|w_{1\ldots N}) \cdot \Pr(w_{1\ldots N})$. Fitting graphical models (i.e. joint distributions simplified via conditional independence) - analytical/conv. opt solutions for directed graphical models, using contrastive divergence (approximating the gradient) to fit undirected graphical models.

(14:00 - 16:00) Meeting with MZ regarding The Bakery work with Gilette (i.e. P & G).

(17:15 - 21:00) Coding PV panel classifier.

- Combine validation and training accuracy scores under one file writer.
- Classify query testing images.
- Add image augmentation step during training.
- Update report.

Why are the query images not coming out correctly?

Monday, 17th July 2017

(06:30 - 08:15) Fixed image subsetting problem in TensorFlow API. (08:15 - 13:00) Reporting (17:00 - 20:30) Prince, *Computer Vision.* Hidden Markov models and Markov trees.

1 Prince, Computer Vision (Ch. 10: Graphical Models)

This chapter introduced graphical models as a means of representing and simplifying joint distributions. In the models we've seen so far, it is possible to run up against vast compute requirements as the model expands and an increasing number of parameters need to be estimated. Graphical models circumvent this problem by factorizing the joint distribution into the product of many conditional distributions.

Directed Graphical Models

Directed graphical models perform this factorization directly. The variables are assumed to be independent given their conditional variables, allowing the joint distribution to be factorized into a product over these conditional distributions.

$$\Pr(x_{1\ldots N}) = \prod_{n=1}^{N} \Pr(\mathbf{x}_n | x_{\mathrm{pa}[n]})$$

The l.h.s. of this equation corresponds to the joint distribution over N random variables x. The r.h.s. is the product of conditional distributions over each variable. Each variable is conditional on only a subset of the other variables: otherwise it is independent of all other variables. Quick refresher: the definition of independence:

$$\Pr(x_i, x_j) = \Pr(x_i|x_j) \cdot \Pr(x_j) = \Pr(x_i) \cdot \Pr(x_j)$$

Directed graphical models are easy to read: We represent each random variable as a node, and indicate whether one variable influences another using arrows between nodes. For two nodes to be independent, they should share no common ancestors. A variable is conditionally indpendent of all other variables given its *Markov blanket* - its parents, children, and other parents of its children nodes.

Undirected Graphical Models

Undirected graphical models have a different form to directed graphical models. Rather than expressing a joint distribution as the product of conditional distributions, they instead use the normalized product of a series of functions (*potential* functions) accepting the random variables as arguments.

$$\Pr(x_{1...N}) = \frac{1}{Z}\prod_{c=1}^{C}\phi_c[x_i, ...]$$

Where the normalizing constant Z is known as the partition function:

$$Z = \int \prod_{c=1}^{C}\phi_c[x_i, ...]dx_{1...N}$$

The potential functions $\phi_c[\cdot]$ are constrained to return positive values. Sometimes undirected graphical models are represented using an alternative parameterization, the Gibbs distribution:

$$\Pr(x_{1...N}) = \frac{1}{Z}\exp\big[-\sum_{c=1}^{C}\varphi_c[x_i, ...]\big]$$

If the entire set of variables is passed to each of the potential functions, then this model is known as a product of experts. If subsets are used - *cliques* - then it's a *Markov random field*. The summation is referred to as the *energy* of a given combination of inputs. As for drawing the damned thing: Produce nodes for each variable, then draw connections between nodes in the same clique. To factorize a model from its graph, find the *maximal clique* - the largest subset containing the node in which every node is connected to every other node - for each variable and bundle these together under a potential function.

A benefit of using an undirected representation of a joint distribution is that independence of sets of nodes can be easily deduced: Two sets of nodes are conditionally independent of one another if a third set separates them.

There is another type of graph - a factor graph. I won't relay the details here however, since they will come up in my discussion of the sum-product algorithm.

Modes of Inference for Graphical Models

This chapter highlighted a few modes of inference for graphical models. In inference, we desire the distribution of world states $w_{1...N}$ for a given input $\mathbf{x}1...N$ (assume the inputs variables each correspond to a vector). Apparently, this would be expensive to compute (I haven't yet convinced myself of this). So instead of computing the distribution, we

simply obtain the world states that are maximally probable:

$$\begin{aligned}\hat{w}_{1\ldots N} &= \arg\max_{w_{1..N}} \Big[\Pr(w_{1\ldots N}|\mathbf{x}_{1\ldots N})\Big] \\ &= \arg\max_{w_{1\ldots N}} \Big[\Pr(\mathbf{x}_{1\ldots N}|w_{1\ldots N}) \cdot \Pr(w_{1\ldots N}))\Big]\end{aligned}$$

The likelihood function may contain lots of parameters: If this is the case, then we must resort to other modes of inference, such as computing marginal distributions over each of the world states. This has the benefit of not requiring the relationships between world states to be parameterized.

$$\Pr(w_n|\mathbf{x}_{1\ldots N}) = \int\int \Pr(w_{1\ldots N}|\mathbf{x}_{1\ldots N}) \cdot w_{1..n-1} \cdot w_{n+1\ldots N}$$

If we exploit conditional independence relations, then this is integral is tractable. If it isn't tractable analytically, then we can approximate it by sampling. We could conceivably parameterize a distribution using the samples we draw.

More generally, sampling can prove an effective strategy for understanding the shape of a joint distribution. As has been said, it's unlikely that we can compute and store the parameterization of a large joint distribution. Instead, we can approximate it by sampling its variables in accordance with its graph's strucure. In directed graphical models, this is relatively easy: we use techique called *ancestral sampling* that can be summarized as:

1. Sample a value from the distributions of variables that have no parents.

2. Move through the graph, sampling from nodes whose parents have already been sampled from, parameterizing the distributions of these nodes using the previously sampled values.

We cannot perform ancestral sampling in an undirected graphical model because there is no concept of parentage and we have potential/cost functions as opposed to conditional probability distributions. Because of this, Markov chain Monte Carlo sampling methods are used: drawing a series of samples, conditioning each draw on its predecessor. One particular form of Markov chain Monte Carlo is Gibbs sampling. A Gibbs sampler initializes a vector of values, one for each variable, then immediately sets about creating another vector of sampled variables by drawing from the conditional distributions. The conditional distributions are parameterized using the values sampled in the initial step. By repeating this process many times, the frequency distribution within the series of samples eventually approximates the variables' joint distribution.

Learning in Graphical Models

The parameters θ of the conditional distributions represented by a directed graphical model can be fit by maximizing the conditional likelihoods. Pleasingly convenient!

$$\hat{\theta} = \arg\max_{\theta} \Big[\sum_{i=1}^{I} \sum_{j}^{N} \log \big[\Pr(x_j | x_{\text{pa}[j]}) \big] \Big]$$

Where we have I instances of sets of N variables. If we're lucky we can solve this problem analytically, but if not it is possible to apply a convex optimization algorithm.

Learning in undirected graphical models is more difficult. We again seek to maximize the log-likelihood, however the partition function creates a computationally intractable sticking point. To get around this, we make an approximation. We can show that:

$$\frac{\partial \log[Z(\theta)]}{\partial \theta} = \sum_{x} \Pr(x) \frac{\partial \log \big[f[x, \theta] \big]}{\partial \theta}$$

Where $Z(\theta) = \sum_x f[x, \theta]$. The r.h.s. in the above expression is the expectation w.r.t. x of the derivate. We can approximate this expecation by sampling from the distribution of $\frac{\partial \log \left[f[x,\theta] \right]}{\partial \theta}$ and averaging. This permits us to approximate the gradient of the log-likelihood, further allowing a convex optimization algorithm to be used to maximize the likelihood w.r.t. the parameters.

07:00 - 10:30 : Prince, *Computer Vision* / Bengio paper on representation learning.
12:00 - 16:00 : Library. Took the evening off.

2 Prince, Computer Vision (Ch. 11: Models for Chains and Trees)

- Markov chain assumption: measurements are independent given their world state, world states are only dependent on their immediate predecessors.
- Hidden Markov models assume discrete world states.
- Markov trees consist of nodes that can have more than one inbound connection. Maximum a posteriori for them is also achieved using Viterbi, however compuing the minimum cost pathway at nodes with multiple parents will be more expensive.
- Maximum a posteriori inference for chains - using the Viterbi algorithm to deduce the set of world states that maximizes the posterior probability $\Pr(w_{1\ldots N} | x_{1\ldots N})$. Exploiting conditional independence relations to express this posterior in a form that lends itself to optimization using Viterbi. Viterbi:

1. Plot columns of nodes. Each column corresponds to a world state variable, and each row to a possible discrete world state value.
2. Begin at the first column. Store the cost of each node, $U[w_1 = k]$ within itself.
3. Move to the second column. At each node, compute and store the cheapest pathway to the node - this is a sum of the connected node's stored cost, the pathway cost, and the cost of the node itself.
4. Repeat the above process for all columns of nodes. Denoting the cost stored in the kth node of the lth world state as $S_{k,l}$, this results in:

$$S_{k,l} = \min_j \Big[S_{j,l-1} + P[w_l = k, w_{l-1} = j] + U[w_l = k] \Big]$$

   Where U is the cost of the node described that satisfies $w_l = k$ and P is the the cost of tranisitioning from state j to state k.
5. When the final column (i.e. world state) is reached, inspect the cost stored in each of its nodes and retrieve the pathway associated with the smallest value.

- Viterbi algorithm: Given a set of states, costs associated with each state, and costs to transition between states, what is the least costly series of states?
- Marginal posterior inference for chains: the forward-backward algorithm.
- The sum-product algorithm / factor graphs.

Wednesday, 19th July 2017

(08:15 - 11:35) Crack prediction models. TensorFlow API.

- Allow larger negative samples (i.e. upsample positive instances).
- Pass the dictionary to configure the graph and preprocessing settings.
- Pass a test identifier on each run.

(12:20 - 15:00) Prince, *Computer Vision*. Continuing to fiddle with Hidden Markov models... Still don't really get belief propagation.

(19:00 - 21:00) Read paper, *Efficient Representations of Words in Continuous Vector Spaces*. Outlined how to construct a continuous bag-of-words model. Understood semantic and syntactic similarity measures. Understood the skip-gram model (predicting context - adjacent words - from a given input word). Need to understand subsampling of negative instances...

Thursday, 20th July 2017

Refactored the testing utility (again). It's now notebook-friendly. The change was spurred by dissatisfactory results when running an experiment against a variety of image and patch sizes.

Friday, 21st July 2017

(08:10 - 12:15) I set up patch and image size tests under new infrastructure, then set them running. While the models were training I read the batch normalization paper *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. I'll write out a summary of it when I get back. (13:00 - 15:30) Spatial transformer networks paper. Complete test results for basic convnets against pixel intensities (high resolution encouraged better performance). (18:00 - 21:00) Passed stack of Gabor/Sobel filtered and intensity inputs to two-layer convnet. Generalized some of the testing code. Tomorrow experiment with batch normalization and inception modules.

Saturday, 22nd July 2017

(08:30 - 13:00) Wrestled with TensorFlow code for batch normalization, to no avail. (17:00 - 21:00) Revised modeling strategy - bumped models to an object-oriented format (much cleaner). Observed some unusual behaviour when training the batch-normalized networks: thin networks (i.e. ones with few feature maps per layer) saw little benefit from BN, whereas much broader ones did. Why is this? I suppose first I should point out why some networks may fail to converge, and how batch normalization enables networks to learn more rapidly. (21:00 - 22:30) Refreshed my memory of backpropagation - time well spent.

Backpropagation is an algorithm for computing gradients in neural networks (or, more generally, directed acyclic graphs). Consider a graph in which we have a series of nodes $u^{(1)}, ...u^{(n)}$, with each node $u^{(i)}$ corresponding to a scalar variable. Variables from which a node is computed via some function $f^{(i)}$ are known as node i's parents. Graphically, directed arcs between nodes indicate parentage: We denote the parents of a given node $u^{(i)}$ by $\mathcal{A}^{(\rangle)}$, and we'll use pa$[i]$ to refer to the indices of node i's parents.

Say that we wished to compute the rate of change of the output of this network, $u^{(n)}$, with respect to a variable $u^{(i)}$ in the graph, holding all other variables fixed. In notation, we seek:

$$\frac{\partial u^{(n)}}{\partial u^{(i)}}$$

We can evaluate this quantity by recursively applying the chain rule:

$$\frac{\partial u^{(n)}}{\partial u^{(i)}} = \sum_{j:i\in \text{pa}[j]} \frac{\partial u^{(n)}}{\partial u^{(j)}} \cdot \frac{\partial u^{(j)}}{\partial u^{(i)}} = \sum_{j:i\in \text{pa}[j]} \frac{\partial u^{(j)}}{\partial u^{(i)}} \cdot \sum_{k:j\in \text{pa}[k]} \frac{\partial u^{(j)}}{\partial u^{(k)}} \cdot \frac{\partial u^{(n)}}{\partial u^{(k)}}$$

We can keep expanding the derivatives until we reach the final layer in the network before $u^{(n)}$, where the expression for the gradient can be evaluated (if possible). Backpropagation allows us to efficiently evaluate expressions such as the above when computing derivatives with respect to many nodes in the network. Rather than re-evaluating a given gradient w.r.t. the output each time it is needed, we compute it once, then store it. When we move towards earlier layers in the network, we can look up this gradient rather than compute it - this reduces execution time.

1 Convergence in Neural Networks

As I understan

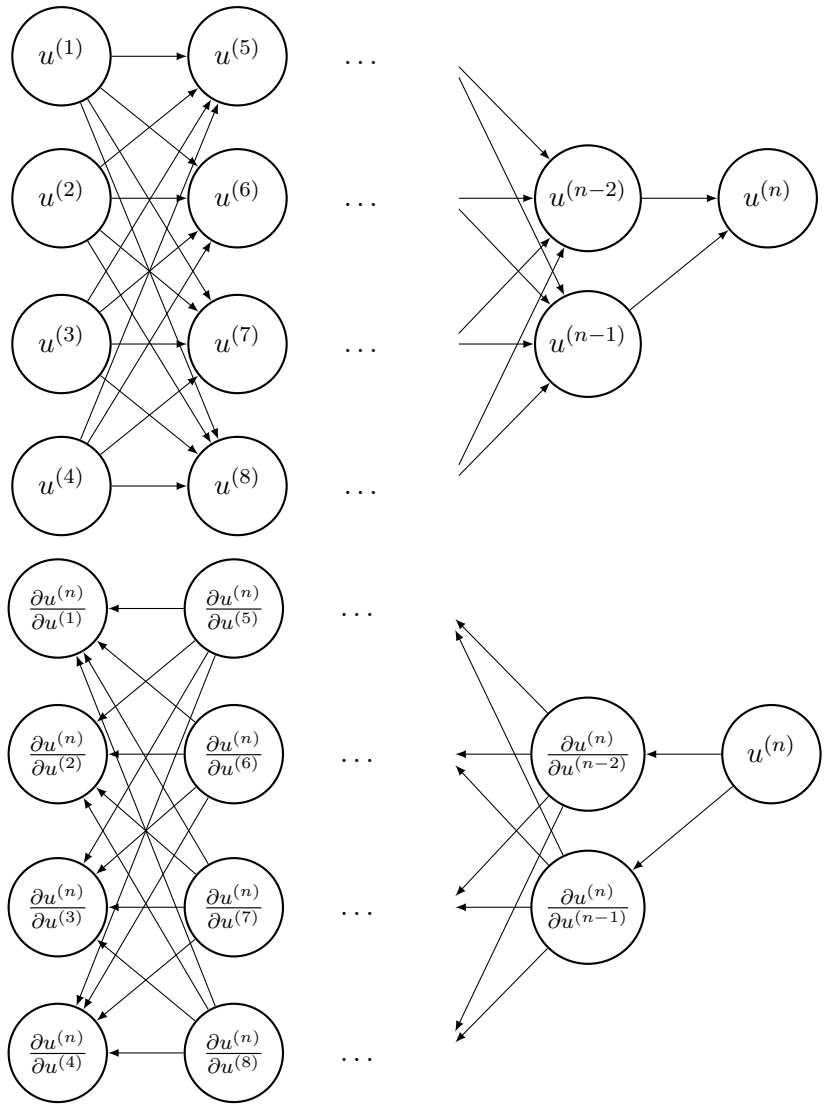


Figure 1: *(Top)* A multi-layer perceptron. *(Bottom)* Its associated backpropagation graph.

Sunday, 23rd July 2017