ECE368: Probabilistic Reasoning

Aman Bhargava

January-April 2021

Contents

0.1 Introduction and Course Information

Course Information

- Professors: Prof. Saeideh Parsaei Fard and Prof. Foad Sohrabi
- Course: Engineering Science, Machine Intelligence Option

- Term: 2021 Winter

Main Course Topics

- Vector, temporal, and spatial models.
- Classification and regression model training.
- Bayesian statistics, frequentist statistics.

Chapter 1

Review Topics

See ECE286 notes for further reference

1.1 Review of Probability Functions

Probability Mass Function: For *discrete random variables*, $P_X(x)$denotes the probability that random variable X takes on value x.

Probability Density Function: For *continuous random variables*, the probability $\Pr\{X \in [x_1, x_2]\}$ is given by $\int_{x_1}^{x_2} f_X(x)dx$.

Joint PMF's and PDF's are similarly defined.

Marginal Probability Distributions: Given joing PMF $P_{X,Y}(x, y)$ or PDF $f_{X,Y}(x, y)$, we can **marginalize** them as follows:

$$P_X(x) = \sum_{y \in Y} P_{X,Y}(x, y) \tag{1.1}$$

$$f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x, y)dy \tag{1.2}$$

Conditional Probability Functions:

$$P_{Y|X}(y, x) = \frac{P_{X,Y}(x, y)}{P_X(x)} \tag{1.3}$$

Prior Probability: Probability **before** an additional observation is made (hence *prior*). Example: $P_X(x)$.

Posterior Probability: Probability **after** an observation is made (hence *poste*rior). Example: $P_{X|Y}(x, y)$.

Bayes Rule:

$$P(B|A) = P(A|B)\frac{P(B)}{P(A)} \tag{1.4}$$

1.2 Expectation, Correlation, and Independence

Expectation Value: $\mathbb{E}[x] = \sum_{x \in X} P_X(x) = \int_{-\infty}^{\infty} x f_X(x) dx$

Law or Large Numbers: $\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} x_i = \mathbb{E}[X]$

Variance:

$$\begin{aligned} \mathrm{Var}(X) &= \mathbb{E}[(X - \mathbb{E}[x])^2] \\ &= \mathbb{E}[X^2] - (\mathbb{E}[X])^2 \end{aligned} \tag{1.5}$$

Covariance:

$$\begin{aligned} \mathrm{Cov}(X, Y) &= \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] \\ &= \mathbb{E}_{XY}[XY] - \mathbb{E}[X]\mathbb{E}[Y] \end{aligned} \tag{1.6}$$

Correlation Coefficient:

$$\rho_{XY} = \frac{\mathrm{Cov}(X, Y)}{\sqrt{\mathrm{Var}(X)}\sqrt{\mathrm{Var}(Y)}} \tag{1.7}$$

- $\rho_{XY} \in [-1, 1]$
- $\rho > 0$ indicates positive correlation (line of best fit has positive slope).
- $\rho < 0$ indicates negative correlation.
- $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$ **iff** X, Y **are uncorrelated**.

Independence

Theorem 1 *Independence Random variables X, Y are independent* ***iff***

$$P_{XY}(x, y) = P_X(x) \cdot P_Y(y) \tag{1.8}$$

This also means that $\rho_{XY} = 0$, $P(X|Y) = P(X)$, etc.

1.3 Laws of Large Numbers

Weak Law: Sample mean converges to the mean.

Strong Law: If $\{x_i\}$ are **independent, identically distributed** (i.i.d.) random variables with mean μ, then the **probability of** the sample mean $= \mu$ is 1 as $n \to \infty$.

Chapter 2

Parameter Estimation

2.1 Estimation Terminology

- $\hat{\theta}_n$ is an **estimator** of some unknown parameter θ.
- **Estimation Error:** $\hat{\theta}_n - \theta$
- **Bias** of estimator: $\mathbb{E}[\hat{\theta}_n] - \theta$
  - **Unbiased** estimator: Bias$= 0 = \mathbb{E}[\hat{\theta}_n] - \theta$.
  - **Asymptotically Unbiased:** $\lim_{n\to\infty} \mathbb{E}[\hat{\theta}_n] = \theta$ for all θ.
- **Consistency:** Estimator is consistent if $\lim_{n\to\infty} \hat{\theta}_n = \theta$.

2.2 Maximum Likelihood Estimation

Framing: Let random variable $\vec{X} = [X_1, X_2, ..., X_n]$ be defined by either

1. Joint PMF $P_{\vec{X}}(\vec{x}; \theta)$
2. Joint PDF $f_{\vec{X}}(\vec{x}; \theta)$

$\vec{x}$ is a series of measurements.

Maximum Likelihood Estimation: The ML estimate of model parameter θ is

$$\hat{\theta}_n = \text{argmax}_\theta P_{\vec{X}}(\vec{x}; \theta) \tag{2.1}$$

Independent, identically distributed case: If each $x_i \in \vec{x}$ are independent and identically distributed, then

$$P_{\vec{X}}(\vec{x};\theta) = \prod_{i=1}^{n} P_X(x_i;\theta) \tag{2.2}$$

Which we can convert to a summation by taking the **log-likelihood** (recall that logarithm is monotonically increasing, so maximizing log-likelihood is equialent to maximizing likelihood).

$$\hat{\theta}_n = \arg\max_{\theta}(\sum_{i=1}^{n} \log P_X(x_i;\theta)) \tag{2.3}$$

2.3 Frequentist vs. Bayesian Statistics

Frequentist: In **classical statistics**, probability is taken to be approximately equal to the **frequency of events**. Model parameters are assumed to have some deterministic, fixed value (even though they might be unknown).

Bayesian Statistics: Model parameters are treated as **random variables** with their own distributions.

- Generally the more modern approach.
- We are most interested in the **joint probability distribution** of model parameters and model arguments (e.g., $f_x(x,\theta)$).
- **Main criticism:** probabilities are assigned to unrepeatable events (arguably violates the definition of probability as the limit of event frequency).

2.4 Maximum a Posteri Estimation (MAP)

$$\begin{aligned}\hat{\theta}_{map} &= \arg\max_{\theta} f_{\theta|x}(\theta|x) \\ &= \arg\max_{\theta} f_{X|\theta}(x|\theta)\frac{f_\theta(\theta)}{f_X(x)}\end{aligned} \tag{2.4}$$

Where $f_\theta(\theta)$ is the **prior distribution** of model parameter.

- If $f_\theta(\theta)$ is uniform, we will still get the same answer as a **maximum likelihood** estimation.

2.4.1 Picking a Prior Distribution

Best Practice: Pick a distribution of the same form as $f_{X|\theta}(x|\theta)$ (called "conjugate pair").

Beta Distribution: Used for **binomial distribution**.

- Binomial distribution:

$$P_{X=k|\theta} = \binom{n}{k}\theta^k(1-\theta)^{n-k} \tag{2.5}$$

  where

  - θ: Probability of success on each Bernoulli trial.
  - n: Total number of trials.
  - k: Total number of successful trials.

- **Beta Distribution:**

$$f_\theta(\theta;\alpha,\beta) = \begin{cases} c\theta^{\alpha-1}(1-\theta)^{\beta-1} & \text{for } \theta \in [0,1] \\ 0 & \text{else} \end{cases} \tag{2.6}$$

  Where

  - α, β are customizable parameters.
  - $c = [\Gamma(\alpha+\beta)]/[\Gamma(\alpha)\Gamma(\beta)]$
  - $\Gamma(x) \equiv \int_0^\infty u^{x-1}e^{-u}du$
  - $\Gamma(x+1) = x\Gamma(x)$ for all $x \in \mathbb{R}$.
  - $\Gamma(n+1) = n!$ for integer n.
  - $\therefore c = \frac{(\alpha+\beta-1)!}{(\alpha-1)!(\beta-1)!}$ for integer α, β.
  - $\mu_f = \mathbb{E}[f_\theta(\theta)] = \frac{\alpha}{\alpha+\beta}$
  - Maximum likelihood $\arg\max_\theta f_\theta(\theta) = \frac{\alpha-1}{\alpha+\beta-2}$

2.5 Conditional Expectation Estimator

Key Idea: Find the **expected value** for the estimator given your observations.

$$\hat{\theta}_{conditionalexpectation} = \mathbb{E}[\theta|\vec{X} = \vec{x}] = \int_{-\infty}^{\infty} \theta f_{\theta|\vec{x}}(\theta|\vec{x}) \tag{2.7}$$

2.6 Bayesian Least Mean Square Estimator (LMS)

Key Idea: To estimate random variable model parameter θ, we find

$$\hat{\theta}_{LMS} = \arg\min_{\hat{\theta}} \mathbb{E}[(\theta - \hat{\theta})^2] \tag{2.8}$$

- $\hat{\theta}_{LMS} = \mathbb{E}[\theta]$ achieves the goal.
- **Equivalently:** We can also find

$$\hat{\theta}_{LMS} = \arg\min_{\hat{\theta}} (\mathbb{E}[\theta - \hat{\theta}])^2 \tag{2.9}$$

Chapter 3

Hypothesis Testing

Goal: Given two hypotheses H_0, H_1 and observation $\boldsymbol{x} = (x_0, \ldots, x_n)$, we wish to decide which hypothesis is **better**.

Error Types: These are generally with respect to H_0, or the "null hypothesis".

- **Type I:** False **rejection**. We reject hypothesis H_i despite it being the correct one.
- **Type II:** False **acceptance**. We accept hypothesis H_i despite it being false.

3.1 Likelihood Ratio Test

$$\mathbb{L}(\boldsymbol{x}) = \frac{P_x(\boldsymbol{x}; H_1)}{P_x(\boldsymbol{x}; H_0)} \lessgtr z \tag{3.1}$$

If $\mathbb{L}(\boldsymbol{x}) > z$, we **accept** H_1. Else, we accept H_0. $z = 1$ corresponds to the maximum likelihood decision rule.

Neyman-Pearson Lemma: We let α represent the probability of **false rejection** and β represent the probability of **false acceptance** (i.e., type I and type II error probability respectively).

$$P(\mathbb{L}(\boldsymbol{x}) > z; H_0) = \alpha \tag{3.2}$$

$$P(\mathbb{L}(\boldsymbol{x}) \leq z; H_1) = \beta \tag{3.3}$$

There is a direct tradeoff between α and β – they are inversely proportional (i.e., we cannot get better overall confidence "for free" with the same data).

There is equivalence between selecting some likelihood cutoff z and some γ for $\boldsymbol{x} \lessgtr \gamma$.

3.2 Bayesian Hypothesis Testing

The likelihood ratio test is essentially a **maximum likelihood** method. Bayesian hypothesis testing is equivalent to **maximum a posteri** methods.

- **Goal:** Choose the most probable hypothesis *given the data*.
- **Given:** Hypotheses $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_M\}$, data $\boldsymbol{x} = (x_1, \ldots, x_n)$.
- **Method:** Select the optimal hypothesis based on

$$\begin{aligned} \hat{\theta} &= \arg\max_{\theta \in \boldsymbol{\theta}} P(\theta|\boldsymbol{x}) \\ &= \arg\max_{\theta \in \boldsymbol{\theta}} P(\boldsymbol{x}|\theta) P(\theta) \end{aligned} \tag{3.4}$$

By maintaining $\boldsymbol{x}$ as a free variable in these computations, the rejector regions $\mathcal{R}$ for $\boldsymbol{x}$ can be found relatively easily.

3.3 Gaussian Vector Distribution

Scalar Gaussian Normal Distribution:

$$\begin{aligned} f_x(x) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp[\frac{-(x-\mu)^2}{2\sigma^2}] \\ &\Leftrightarrow x \sim \mathcal{N}(\mu, \sigma^2) \end{aligned} \tag{3.5}$$

Gaussian Vector

$$f_{\vec{x}}(\vec{x}; \vec{\mu}, \Sigma) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{\frac{1}{2}}} \exp[\frac{-1}{2}(\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu})] \tag{3.6}$$

Where

- D is the dimension of the vectors x.
- $\vec{\mu} \in \mathbb{R}^D$ is the mean vector.

- $\vec{\mu} = \mathbb{E}[\vec{x}]$

- $\Sigma \in \mathbb{R}^{D \times D}$ is the covariance matrix. It is **positive semidefinite**.
  - $\Sigma = \mathbb{E}[(\vec{x} - \vec{\mu})(\vec{x} - \vec{\mu})^T]$

3.3.1 Eigen Analysis of Gaussian Vectors

From ECE367 : All PSD matrices A have **orthogonal** eigenvectors. We can also arbitrarily scale them to be ortho**normal**.

$$A = Q\Lambda Q^T \tag{3.7}$$

Where each column of Q is an eigenvector and $\Lambda = \text{diag}(\text{eigen values})$. $QQ^T = I$ and $Q^T = Q^{-1}$ by orthonormalcy.

Applying to Gaussian Vectors: We note the term $\frac{-1}{2}(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})$. Since Σ is PSD we can decompose it into

$$\begin{aligned} \Sigma &= Q\Lambda Q^T \\ \Sigma^{-1} &= Q\Lambda^{-1}Q^T \end{aligned} \tag{3.8}$$

We define a helper variable $\vec{y}$ as follows:

$$\begin{aligned} &\vec{y} \equiv Q^T(\vec{x} - \vec{\mu}) \\ &\Rightarrow \vec{y}^T = (\vec{x} - \vec{\mu})^T Q \\ &\Rightarrow (\vec{x} - \vec{\mu})^T\Sigma^{-1}(\vec{x} - \vec{\mu}) = \vec{y}^T Q^T\Sigma^{-1}Q\vec{y} \\ &= \vec{y}^T\Lambda^{-1}\vec{y}^T \end{aligned} \tag{3.9}$$

The big payoff is that $\vec{y}$ is a random variable with a **diagonal** covariance matrix (i.e., independent components). $\vec{y}$ also has zero mean!

$$\begin{aligned} f_y(\vec{y}) &= \frac{1}{(2\pi)^{D/2}|\Lambda|^{1/2}} \exp(\frac{-1}{2}\vec{y}^T\Lambda^T\vec{y}) \\ &= \prod_{i=1}^{D} \frac{1}{\sqrt{2\pi\lambda_i}} \exp[\frac{-y_i^2}{2\lambda_i}] \end{aligned} \tag{3.10}$$

3.4 Gaussian Estimation

Given: Observations $x_i \in \mathbb{R}$ where we know that were generated by $x_i = \theta + w_i$ and $w_i \sim \mathcal{N}(0, \sigma_i^2)$. In other words, each x_i is a "measure" of the same mean value θ, but there is some zero-mean noise w_i with variance σ_i^2. Note that each data point has its own variance.

Goal: Estimate θ.

3.4.1 Maximum Likelihood

$$\begin{aligned}\hat{\theta}_{ML} &= \arg\max_{\theta} f_x(\vec{x};\theta) \\ &= \arg\max_{\theta} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_i} \exp[\frac{-1(x-\theta)^2}{2\sigma_i^2}] \\ \hat{\theta}_{ML} &= \frac{\sum_{i=1}^{n} \frac{x_i}{\sigma_i^2}}{\sum_{i=1}^{n} \frac{1}{\sigma_i^2}}\end{aligned} \tag{3.11}$$

Intuitively, this corresponds to a weighted sum of each x_i. Weight is proportional to $\frac{1}{\sigma_i^2}$, which corresponds to "certainty" in the validity of the data point in representing θ.

3.4.2 MAP Estimation

We assume a **conjugate prior** distribution for $\theta \sim \mathcal{N}(x_0, \sigma_0^2)$. Conveniently enough, these x_0, σ_0 are functionally identical to just having another data point!

$$\begin{aligned}\hat{\theta}_{MAP} &= \arg\max_{\theta} f_\theta(\theta) f(x|theta) \\ &= \dots \\ \hat{\theta}_{MAP} &= \frac{\sum_{i=0}^{n} \frac{x_i}{\sigma_i^2}}{\sum_{i=0}^{n} \frac{1}{\sigma_i^2}}\end{aligned} \tag{3.12}$$

Note that the sums now start at zero to incorporate the prior distribution's information!

Chapter 4

Statistical Machine Learning

4.1 Naive Bayesian Classifier

Technically this was in the first half, but it fits well to motivate LDA and QDA.

Goal: Let $\theta \in \{1, 2\}$ be the class of data points $\vec{x} \in \mathbb{R}^n$. We wish to find $P(\theta|\vec{x})$.

Naive Bayesian Assumption: Each component of $\vec{x}$ is **independent** with respect to class θ. By the probability axioms relating to independence,

$$P_{\vec{x}|\theta}(\vec{x}|\theta) = \prod_{i=1}^{n} P_{x_i|\theta}(x_i|\theta) \tag{4.1}$$

Classification Equations:

$$P(\theta|\vec{x}) = \frac{P(\theta)\prod_{i=1}^{n} P_{x_i|\theta}(x_i|\theta)}{P(\vec{x})} \tag{4.2}$$

Where $P(\vec{x}) = \sum_\theta [P(\theta)\prod_{i=1}^{n} P_{x_i|\theta}(x_i|\theta)]$

$$P(\theta = 1)\prod_{i=1}^{n} P_{x_i|\theta}(x_i|\theta = 1) \lessgtr P(\theta = 2)\prod_{i=1}^{n} P_{x_i|\theta}(x|\theta = 2) \tag{4.3}$$

Bag of words: For classifying text, we choose a set of W of the most common words. In our vectors $\vec{x}$, each index x_i corresponds to whether word

w_i appears in the given text.

$$P_{x_i|\theta}(x_i = w_d|\theta = j) = \frac{\text{occurrences of } w_i \text{ in set } j}{\text{total words in set } j} \tag{4.4}$$

Laplace Smoothing: Even if we perform the computation in the log domain, we might run into a $P_{x_i|theta}(x_i|\theta) = 0$ if we lack a datapoint (word occurrence) for a given class. Laplace smoothing simply appends the vocabulary set W to each class (i.e., all words in the set occur at least once a priori).

4.2 Linear Discriminant Analysis (LDA)

Goal: Create algorithm LDA : $\vec{x}_i \to c_i$ where $\vec{x}_i$ is a data point and $c_i \in C$ is the class it belongs to.

General Methodology: We use **Bayesian hypothesis testing** with variable classes we model with normal distributions. An important simplifying assumption is that each class c has different $\vec{\mu}_c$ **BUT** they all have the same Σ. This is what makes our decision boundaries **linear**!

Classification Equations:

1. $\hat{y}(\vec{x}) = \arg\max_{c \in C} \beta_c^T \vec{x} + \gamma_c$ where
   - $\beta_c = \Sigma^{-1}\mu_c$
   - $\pi_c = P(c)$ (prior probability of class c)
   - $\gamma_c = \log(\pi_c) - \frac{1}{2}\mu_c^T \Sigma^{-1} \mu_c$
2. Posterior probability of class c:

$$P(c|\vec{x}) = \frac{\exp(\beta_c^T x + \gamma_c)}{\sum_{c' \in C} \exp(\beta_{c'}^T x + \gamma_{c'})} \tag{4.5}$$

4.3 Quadratic Discriminant Analysis (QDA)

Goal: Same as LDA.

Assumptions: We are given μ_c, Σ_c, π_c for each class, and each $\vec{x}_i \sim \mathcal{N}(\mu_c, \Sigma_c)$ for some $c \in C$.

Classification Equations:

$$\hat{y}(\vec{x}) = \arg\max_{c \in C} [\log(\tilde{\pi}_c) - \frac{1}{2}(x - \mu_c)^T \Sigma_c^{-1}(x - \mu_c)] \tag{4.6}$$

Where $\tilde{\pi}_c = \frac{1}{(2\pi)^{D/2}|\Sigma_c|^{1/2}}\pi_i$

$$P(c|x) = \pi_c \frac{1}{(2\pi)^{D/2}|\Sigma_c|^{1/2}} \exp(\frac{-1}{2}(x - \mu_c)^T \Sigma^{-1}(x - \mu_c)) \frac{1}{P(x)} \tag{4.7}$$

Where $P(x)$ is given by

$$\begin{aligned} P(x) &= \sum_{c' \in C} P(x|c')P(c') \\ &= \sum_{c' \in C} \mathcal{N}(x; \mu_c, \Sigma_c)P(c') \end{aligned} \tag{4.8}$$

Calculating μ_c, Σ_c The maximum likelihood estimation of these are as follows:

$$\hat{\mu}_c = \frac{1}{n}\sum_{i=1}^{n} x_i \ \forall x_i \in c \tag{4.9}$$

$$\hat{\Sigma}_c^{ML} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)^T \ \forall x_i \in c \tag{4.10}$$

PROBLEM: $\hat{\Sigma}_c^{ML}$ is a biased estimator. For the $D = 1$ case, $\mathbb{E}[\hat{\sigma}_c^{ML}] = \frac{n-1}{n}\sigma^2$. To correct, we use the following:

$$\hat{\Sigma}_c^{corrected} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)^T \ \forall x_i \in c \tag{4.11}$$

The primary problem with these sorts of classifiers is that Σ has high dimensionality. If we use the same Σ for all classes, we get LDA. If we force Σ_c to be diagonal, we get a naive Bayesian classifier.

4.4 General Bayesian Inference on Gaussian Vectors

Given: We know that $\vec{z} \sim \mathcal{N}(\vec{\mu}, \Sigma)$. We know the value of only part of the $\vec{z}$ vector, and want to determine the probability distribution and ML/MLE/MMSE estimation for the remaining indices we don't know.

$$\vec{z} = \begin{bmatrix} \vec{x} \\ \vec{y} \end{bmatrix} \tag{4.12}$$

We are given the value of $\vec{y}$ and want to know $\vec{x}$.

Key Tools:

$$\Sigma = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} \tag{4.13}$$

Where $\Sigma_{xy} = \mathbb{E}[(x - \mu_x)(x - \mu_x)^T]$.

- *The maximum of a Gaussian is at the mean!*
- $f_{x|y} \sim \mathcal{N}$
- $\hat{x}_{MAP} = \hat{x}_{MMSE,LMS} = \mathbb{E}[\vec{x}|\vec{y}]$

Results:

$$f_{x|y}(x|y) \sim \mathcal{N}(\mu_{x|y}, \Sigma_{x|y}) \tag{4.14}$$

Where

$$\mu_{x|y} = \mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y - \mu_y) \tag{4.15}$$

$$\Sigma_{x|y} = \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx} \tag{4.16}$$

Which is pretty cool, especially considering that $\hat{x}_{MAP,MMSE} = \mu_{x|y}$ is now linear with $\vec{y}$. We can think of $\Sigma_{x|y}$ as the "remaining uncertainty" in $\vec{x}$ after we receive information $\vec{y}$.

4.5 Linear Gaussian Systems

Given: $P(x) \sim \mathcal{N}(\mu_x, \Sigma_x)$. $y = Ax + b + z$ where

- A is a known matrix.
- b is a known vector.
- $z \sim \mathcal{N}(0, \Sigma_z)$
- x, z are independent.

Goal: We observe y. What are our estimates $\hat{x}_{MAP}$, $\hat{x}_{MMSE}$ that correspond to this particular y?

Solution: Unsurprisingly, x, y have Gaussian distributions.

$$\hat{x}_{MAP,MMSE} = \mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y - \mu_y) \tag{4.17}$$

Where

$$\mu_y = A\mu_x + b \tag{4.18}$$

$$\Sigma_{xy} = \Sigma_x A^T \tag{4.19}$$

$$\Sigma_{yy} = A\Sigma_x A^T + \Sigma_z \tag{4.20}$$

$$\Sigma_{x|y} = (\Sigma_x^{-1} + A^T\Sigma_z^{-1}A)^{-1} \tag{4.21}$$

Chapter 5

Linear Regression

Linear Regression Model: Our aim is to predict y as a linear combination of indices of vector $\vec{x}$. A key difference now is that we are looking at **probabilistic** least squares, so we want a measure of certainty about our predictions. This is why we introduce a σ^2 term:

$$p(y, \vec{x}, \theta) = \mathcal{N}(y|\vec{w}^T\vec{x}, \sigma^2) \tag{5.1}$$

And of course $\mathcal{E}[y|\vec{x}] = \vec{w}^T\vec{x}$. You can also apply polynomial, gaussian, and sigmoid basis functions to transform $\vec{x}$ to get a more complex decision boundary.

Some Notation:

- $\mathcal{D} = (\vec{y}, X)$.
- $\theta = (\vec{w}, \sigma^2)$

Note that from here on out, we use **augmented vectors** in X. That is, we have a 1 at the 1st index of each X to account for the affine offset for each. This way, we don't need to include messy "bias" terms.

$$\vec{x}_{augmented} = \begin{bmatrix} 1 \\ \vec{x} \end{bmatrix}$$

5.1 Ordinary Least Sqauares (MLE) Linear Regression

Generic linear regression (no regularization, no prior) tries to find $\hat{\theta}_{MLE}$ as follows:

$$\begin{aligned}\hat{\theta}_{MLE} &= \arg\max_{\theta} \log[p(\mathcal{D}|\theta)] \\ &= \arg\max_{\theta} \frac{-1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \vec{w}^T\vec{x}_i)^2 - \frac{N}{2}\log(2\pi\sigma^2)\end{aligned} \tag{5.2}$$

Solutions for Ordinary Least Squares

$$\vec{w}_{OLS} = (X^TX)^{-1}X^Ty \tag{5.3}$$

$$\sigma^2_{ML} = \frac{1}{N}\sum_{n=1}^{N}[y_n - \vec{w}^T\vec{x}_n]^2 \tag{5.4}$$

Interestingly, least squares is identical to projecting $\vec{y}$ onto the columns of dataset X. $\vec{w}$ holds the projection coefficients for this.

5.2 Regularized Least Squares

Key idea: We introduce a penalization term in our error related to parameter $\vec{w}$. Error $= E_{\mathcal{D}}(\vec{w}) + \lambda E_{\vec{w}}(\vec{w})$. Known as "weight decay", "parameter shrinkage".

Ridge Regression: Literally just ℓ_2 regularized least squares.

$$\begin{aligned}\vec{w}_{ridge} &= (X^TX + \lambda I)^{-1}X^Ty \\ &= (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T\tilde{y}\end{aligned} \tag{5.5}$$

Where

$$\tilde{X} = \begin{bmatrix} X \\ -I\lambda \end{bmatrix}$$

$$\tilde{y} = \begin{bmatrix} y \\ 0 \\ \dots \\ 0 \end{bmatrix}$$

Interestingly, this is identical to the MAP estimate of $\vec{w}$ with prior distribution $p(\vec{w}) = \prod_{j=1}^{d} \mathcal{N}(w_j|0, \tau^2)$ where $\lambda^2 = \sigma^2/\tau^2$:

$$\begin{aligned} \hat{w}_{MAP} &= \arg\max_{\vec{w}} \log P(\mathcal{D}|\vec{w}, \sigma) P(\vec{w}) \\ &= \arg\min_{\vec{w}} \sum_{i=1}^{n} (y_i - \vec{w}^T \vec{x}_i)^2 + \lambda^2 ||\vec{w}||_2^2 \end{aligned} \tag{5.6}$$

You can also apply previously discussed linear Gaussian vector formulations by introducing noise variable $\vec{z} \sim \mathcal{N}(\vec{0}, \sigma^2 I)$ and using $\vec{w}$ as the variable we are trying to estimate with prior $\vec{w} \sim \mathcal{N}(\vec{0}, \tau^2 I)$.

5.3 Logistic Regression

Here we apply a sigmoid function to the $\vec{w}^T \vec{x}$ value and use the resulting value as the probability of "positive" y via a bernoulli distribution:

$$P(y|\vec{w}, \vec{x}) = \text{Bernoulli}(\sigma(\vec{w}^T \vec{x})) \tag{5.7}$$

$$\hat{w}_{ML} = \max_{\vec{w}} [-\sum_{i=1}^{n} y_i \log(\sigma(\vec{w}^T \vec{x})) + (1 - y_i) \log(1 - \sigma(\vec{w}^T \vec{x}))] \tag{5.8}$$

Interestingly this is pretty much just cross entropy loss. There's no analytical solution, so we need to use numerical techniques. It's also pretty easy to add regularization to this "loss" function.

5.4 Bayesian Linear Regression

In case you wanted to be really extra and compute the full probability distributions of $\vec{w}$, σ^2 for a linear regression model, you can use Bayesian linear regression.

Goal: We wish to compute **full posteriors** for $\vec{w}, \sigma^2$ given the dataset $\mathcal{D}$. We assume a Gaussian prior on $\vec{w}$ so that the end distribution is also nice and Gaussian:

Tools to Get There: I won't build the full proof, but here are the building blocks:

$$p(\vec{w}) = \mathcal{N}(\vec{w}|\vec{w}_0, \Sigma_0) \tag{5.9}$$

$$p(\vec{y}|X, \vec{w}, \sigma^2) = \mathcal{N}(\vec{y}|X\vec{w}, \sigma^2 I_{N\times N}) \tag{5.10}$$

Solution for Known σ: Using "Bayes theorem for Gaussians",

$$\begin{aligned} p(\vec{w}|X, \vec{y}, \sigma^2) &\propto \mathcal{N}(\vec{w}|\vec{w}_0, \Sigma_0)\mathcal{N}(\vec{y}|X\vec{w}, \sigma^2 I_{N\times N}) \\ &= \mathcal{N}(\vec{w}|\vec{w}_N, \Sigma_N) \end{aligned} \tag{5.11}$$

Where

$$\begin{aligned} \vec{w}_N &= \Sigma_N \Sigma_0^{-1}\vec{w}_0 + \frac{1}{\sigma^2}\Sigma_N X^T \vec{y} \\ \Sigma_N^{-1} &= \Sigma_0^{-1} + \frac{1}{\sigma^2}X^T X \end{aligned} \tag{5.12}$$

Some notes on intuition:

- Since Σ_N^{-1} scales with $X^T X$, it will grow very large as the number of data points increases. This means that Σ_N will be pretty small with many samples, which intuiviely makes sense because additional data (should) result in more "certainty".
-

Posterior Distribution of y: Let's say we are given some dataset $\mathcal{D}$ to train on. Now we want to perform "inference" on some new data point $\vec{x}$ to get a distribution on the corresponding value of y:

$$p(y|\vec{x}, \mathcal{D}, \sigma^2) = \mathcal{N}(y|\vec{w}_N^T\vec{x}, \sigma_N^2(x)) \tag{5.13}$$

Where $\sigma_N^2(x) = \sigma^2 + \vec{x}^T\Sigma_N\vec{x}$.

Bayesian Regression with Unknown σ^2: Generally, we are not going to know what σ^2 is in advance for y. See Murphy p. 234 for how to do this (this topic was not covered in this course).

Chapter 6

Markov Chains

Often we are interested in understanding the evolution of some random variable over time (e.g., a robot's position). Here, we examing **Markov decision processes**, a framework for understanding the evolution of some random variable over discrete time steps and discrete states. **Markov chains** are of particular interest.

6.1 Preliminary Definitions

Markov Chain: Consists of

1. A finite set of discrete **states** $S = \{1, \ldots, m\}$.
   - At each time step n, our random variable is in one of those state $X_n = s_i \in S$.
2. **Transition probabilities:** $\Pr(i \to j) \forall i, j \in S$. Matrix $A \in \mathbb{R}^{m \times m}$ as
$$A_{i,j} = \Pr(X_{n+1} = j | X_n = i) \tag{6.1}$$
3. **Memoryless assumption:** The probability distribution of the next state depends *only* on the current state. Also known as the "Markov property".
4. There can be no other states than those in S, so $\sum_{j=1}^{m} A_{i,j} = 1$ (rows of P sum to 1).

Transition probability matrix: Matrix A from above always has an eigen value of 1, and the rows all sum to 1. It's called a *stochastic matrix*, and we learned about these in ECE367! It can also be represented as a transition probability graph with edge weights representing the probability of traversal.

Path probability: $P(< X_0 \to X_1 \to \cdots \to X_n >) = \prod_{i=1}^{n-1} A_{X_i \to X_{i+1}}$.

Initial conditions: We can have a *probabilistic initial condition* with vector $\vec{\pi}^0$:

$$\Pr(X_0 = j) = \vec{\pi}_j^(0) \tag{6.2}$$

We can find the probability vector for the next time step as:

$$\vec{\pi}^{(i+1)} = \vec{\pi}^{(i)T} A \tag{6.3}$$

6.2 Markov Chain Steady State

Definition: If $\lim_{n\to\infty} \vec{\pi}^{(n)} = \tilde{\pi}$ regardless of the starting state $\vec{\pi}_j^{(0)}$, then $\tilde{\pi}$ is the **steady state distribution** satisfying

$$\tilde{\pi} = \tilde{\pi}^T A \tag{6.4}$$

This has deep connections to flow optimization (ECE358: Algorithms and Datastructure). When viewed on the graphical model of the chain, the steady state distribution corresponds to the "flow" into each node being equivalent to the "flow" out of each node.

Conditions for Steady State:

1. A **periodic** Markov chain has no steady state. The steady state behaviour of these chains oscillate between two or more nodes with $\Pr = 1$ (think of a 2-node cycle with transition probabilities of 1).

2. **Multiple recurrent graph** Markov chains have no steady state. This occurs when some node has an edge configuration such that, after traversing the node, part of the graph can never be reached. An example of this is if the starting node has ONLY two outgoing nodes leading to disconnected sub-graphs.

3. **Absorbing states** yield Markov chains with no steady state. If a state has transition probability 1 for returning to itself, then it "absorbs" all paths into itself forever.

4. **Singly connected/"Recurrent States"** are a REQUIREMENT for a steady state to exist. There must exist a path from $i \to j$ and $j \to i$ for all $i, j \in S$.

Theorem 2 *Steady State Convergence Theorem: For all Markov chains with*

1. *Finite states*
2. *Single recurrent classes*
3. *Aperiodicity*

There exists a ***steady state distribution*** $\tilde{\pi}$ *such that:*

1. $\lim_{n\to\infty} \pi_j^{(n)} = \tilde{\pi}_j$ *for all initial states.*
2. $\tilde{\pi}$ *is a unique solution to*

$$\begin{aligned} \tilde{\pi} &= \tilde{\pi}^T P \\ \sum_{j=1}^{m} \tilde{\pi}_j &= 1 \end{aligned} \tag{6.5}$$

3. *For all "transition states" j,* $\tilde{\pi}_j = 0$.
4. *For all "recurrent states" i,* $\tilde{\pi}_i > 0$.

6.2.1 Eigen Analysis of Steady State

Recall the steady state equation:

$$\tilde{\pi}^T A = \tilde{\pi} \tag{6.6}$$

Let $M = A^T$ and $\vec{x} = \tilde{\pi}^T$. We can frame this as an eigen value problem as follows with $\lambda = 1$:

$$M\vec{x} = \lambda\vec{x} \tag{6.7}$$

Theorem 3 *For stochastic matrix M, there exists eigen value* $\lambda = 1$, *and the rest of the eigen values satisfy* $|\lambda| < 1$.

Since $\det(A) = \det(A^T)$, *M and A have the same eigen values. We can also show that iterated multiplications by A result in all eigenvectors except the one corresponding to* $\lambda = 1$ *go to zero. Therefore, the steady state value will be the eigenvector associated with* $\lambda = 1$.

Power iteration algorithm: To find steady state (i.e., $\vec{v}^{(1)}$):

1. Initialize $v^{(0)}$.
2. For $\ell = 1, \ldots$:
$$v^{(\ell)} = \frac{Mv^{(\ell-1)}}{||Av^{(\ell-1)}||} \tag{6.8}$$
3. End.

Applications include Pagerank (see ECE367.pdf).

Chapter 7

Directed Graphical Models

Diagramatic representations of joint and conditional probability distributions can be very helpful for representing and understanding the relationships between random variables. For graphical models, we have:

- **Nodes:** Random variables.
- **Edges:** Joint/conditional distribution connections.

Directed and undirected graphs are the main classes for graphical models. Directed graphical models are known as "Bayes nets", "causal networks", or "belief networks". Undirected models are known as "random Markov fields".

7.1 Defining Graphical Models

Hierarchy: If there exists an edge connecting node $a \rightarrow b$, then a is a parent of b and b is a child of a. This also means that there exists some term in the joint probability distribution of all the nodes $p(b|a)$. That is, the value of a informs the value of b.

Graphical Model $\rightarrow$ Joint Probability Distribution: For a graph compoased of nodes $x_{1:N}$, the joint probability distribution is:

$$P(x_{1:N}) = \prod_{i=1}^{N} P(x_i|pa_i) \tag{7.1}$$

Where pa_i is the set of **parents** of node x_i.

Independence: Recall that independence occurs when knowing the value of B variable does not inform the probability distribution of A (and vice versa):

$$P(A|B) = P(A); \;\; P(B|A) = P(B) \tag{7.2}$$

$$P(A, B) = P(A)P(B) \tag{7.3}$$

Conditional Independence: Independence when the two variables are conditioned on some tertiary variable C. Surprisingly, conditional independence generally does not imply regular independence.

$$P(A, B|C) = P(A|C)P(B|C) \tag{7.4}$$

Which is equivalent to

$$A \perp\!\!\!\perp B \perp C \tag{7.5}$$

Generally, we arrive at conditional independence by comparing the values of $P(A|C)P(B|C)$ with $P(A, B|C) = P(A, B, C)/P(C)$.

7.2 Conditional Independence for Connection Types

Head to Head Connections: A head to head connection is when the "arrow heads" of our graphical model meet at the same node. For a graphical model of a, b, c, we would have edges $a \to c$ and $b \to c$ forming a *head to head* connection at c.

Head to Tail Connection: c has a head-to-tail connection if we have $a \to c$ and $c \to b$.

Tail to Tail Connections: c has a tail-to-tail connection if we have $c \to a$ and $c \to b$.

Result:

Theorem 4 *Connection Types and their Conditional Independences:*

- ***Head-to-Head:*** *a is independent of b. They are NOT conditionally independent given c, though!*

$$a \perp\!\!\!\perp b \; BUT \; a \not\perp\!\!\!\perp b \perp c \tag{7.6}$$

- ***Head-to-Tail:*** *a is independent of b given c. This corresponds to a Markov chain's next state depending only on the present state.*

$$a \perp\!\!\!\perp b \perp c \tag{7.7}$$

- ***Tail-to-Tail:*** *a and b are generally dependent on eachother. However, once you know the shared parent, they become independent.*

$$a \not\perp\!\!\!\perp b \; BUT \; a \perp\!\!\!\perp b \perp c \tag{7.8}$$

7.3 Direct Separation (D-Sep)

D-separation property for directed graphs tells us if we have conditional independence for subsets of a graph.

Theorem 5 *D-separation theorem: Let A, B, C be disjoint subsets of the nodes in a directed, acyclic graphical model. To ascertain whether $A \perp\!\!\!\perp B \perp C$, we do the following:*

1. *Consider all paths p from all nodes in A to all nodes in B (note that this path can traverse any edge regardless of the direction of the edge).*
2. *A path is **blocked** if it includes a node c where either*
   (a) *There is a **head-to-tail** or **tail-to-tail** meeting some node $c \in C$.*
   (b) *There is a **head-to-head** meeting at node $c \notin C$ AND none of the **descendants** of c are in C.*

   If all paths p are blocked, then A is d-separated from B by C. All joint distributions over all variables satisfy $A \perp\!\!\!\perp B \perp C$.

Utility of D-separation: We can determine how/if variables in a model are conditionally independent.

7.4 Markov Blanket/Boundary

Definition: A markov blanket for a given node x_i consists of the

- Parents
- Children

- Co-parents

of the node. The meaning is that $P(x_i|x_{\{j\neq i\}} = f(\text{Markov blanket nodes})$. That is, you only really "need" the nodes in the Markov blanket to maximally "inform" the distribution of variable x_i.

This can be shown by analysis of $p(x_i|x_{i\neq j})$:

$$\begin{aligned} p(x_i|x_{j\neq i}) &= \frac{p(x_{1:N})}{\int p(x_{1:N})dx_i} \\ &= \frac{\prod_k p(x_k|\text{pa}_k)}{\int \prod_k p(x_k|\text{pa}_k)dx_i} \end{aligned} \tag{7.9}$$

and factoring out terms in the integral in the denominator that are not functions of x_i (i.e., those not members of the Markov blanket). As one might expect, the node x_i is independent from the rest of the graph given the Markov blanket node values.

Chapter 8

Markov Random Fields (Undirected Graphical Models)

8.1 Conditional Independence Properties

The analogy for D-separation is simpler for undirected graphical models since we don't need to worry about the case of descendants informing us, and there is no way to distinguish head-to-head vs. head-to-tail connections.

Theorem 6 *Conditional Independence for Undirected Graphical Models: Let* A, B, C *be disjoint subsets of the nodes in the graph.* $A \perp\!\!\!\perp B \mid C$ ***iff*** *all paths from any node in* A *to any node in* B ***pass through*** C*.*

From here, we can derive that the **Markov blanket** for any node is simply **all directly neighbouring nodes**.

8.2 Factorizing a Markov Random Field

Maximal Clique: A subset of the nodes in G that is a clique BUT would cease to be a clique if any more nodes were added.

We factorize undirected graphical models in terms of the max cliques in the graph.

- Let x_c be set of variables in max clique c.
- Let $\phi_c()$ be a **potential** function on clique x_c where $\phi_c(x_c) \geq 0$ always.

Then we have

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_c \phi_c(x_c) \tag{8.1}$$

That is, we multiply the potential functions on all the max cliques in $\boldsymbol{x}$ and normalize with factor Z to get the joint probability distribution of $\boldsymbol{x} = \{x_1, \ldots, x_N\}$.

Energy Functions: To obtain potential functions $\phi_c \geq 0$, we can let $\phi_c(x_c) = \exp[-E(x_c)]$ where $E(x_c)$ is the energy function.

We note that potential functions ϕ and energy functions E have no true probabilistic meaning. They simply denote "preferable" configurations for the system.

Graphical Models as Filters:

- Let $UI = \{p(\boldsymbol{x}) | p(\boldsymbol{x})$ aligns with conditional distributions represented in the graph$\}$.
- Let $UF = \{p(\boldsymbol{x}) | p(\boldsymbol{x})$ that can be expressed as $\frac{1}{Z}\prod_c \phi_c(x_c)\}$.

Theorem: $UI = UF$. We can regard the graphical model as a "filter" that only "lets through" distributions that satisfy the aforementioned requirements on the conditionals.

8.3 Relating Undirected Graphical Models to Directed

The general method for conversion from directed to undirected graphical models is as follows:

1. For head-to-head connections $p(x_i | x_j, x_k, x_\ell)$, we **marry the parents** by adding additional undirected connections between them.
2. The rest of the directed links are converted to undirected edges.
3. We initialize $\phi_c(x_c) = 1$ for all max cliques c in the graph.
4. We multiply in the corresponding terms from the directed version to each ϕ_c such that $p(\boldsymbol{x})$ remains the same.

D-map: A graph id a d-map of $p(\boldsymbol{x})$ if all conditional independence statements on $p(\boldsymbol{x})$ are reflected in the graph.

I-map: A graph is an i-map of $p(\boldsymbol{x})$ if all conditional independence statements implied by G are atisfied by $p(\boldsymbol{x})$.

A perfect graph is a d-map and an i-map.

Chapter 9

Inference on Graphical Models

Graphical models offer a convenient way to understand the **reasoning** behind predictions. Here we examine some efficient methods that take advantage of graph structure. For marginalizing inference, we would generally need to do the following:

$$P(x_n) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_N} p(x_1, \ldots, x_N) \tag{9.1}$$

These nested sums make for pretty terrible run time ($O(N^k)$ where each x_i has domain of size k), and would need to be replaced by ugly integrations for continuous variable.

General strategy – "pushing in" sums: To make this process faster, we the following technique. For a factorized probability distribution $p(x_1, \ldots, x_N)$, we can "push in" summation operators as follows:

$$\sum_{i=1}^{5} \alpha f(i) = \alpha \sum_{i=1}^{5} f(i) \tag{9.2}$$

In other words, if factorized elements are not a function of the summation variable, they can be factored out to the left of the corresponding summation operator. Since graphical models have a strong ability to factorize probability distributions nicely, we can leverage this for faster inference algorithms.

9.1 Message Passing for First Order Markov Chain

For a first order Markov chain $x_1 \to x_2 \to \cdots \to x_N$, we can convert to an undirected model and find the following joint PDF on all nodes:

$$P(\boldsymbol{x}) = \frac{1}{Z}\phi_{1,2}(x_1, x_2) \cdot \phi_{2,3}(x_2, x_3) \cdot \cdots \cdot \phi_{N-1,N}(x_{N-1}, x_N) \tag{9.3}$$

The "message passing" algorithm will take advantage of this "pushing in" strategy to arrive at the following:

$$p(x_n) = \frac{1}{Z}[\sum_{x_{n-1}} \phi_{n-1,n}(x_{n-1}, x_n) \dots [\sum_{x_2} \phi_{2,3}(x_2, x_3)[\sum_{x_1} \phi_{1,2}(x_1, x_2)]]] \cdot [\sum_{x_{n+1}} \phi_{n,n+1}(x_n, x_{n+1}) \dots [\sum_{x_N} \tag{9.4}$$

$$p(x_n) = \frac{1}{Z}\mu_\alpha(x_n)\mu_\beta(x_n) \tag{9.5}$$

Note the recursive structure of μ_α, μ_β.

$$\begin{aligned} \mu_\alpha(x_n) &= \sum_{x_{n-1}} \phi_{n-1,n}(x_{n-1}, x_n) \cdot \mu_\alpha(x_{n-1}) \\ \mu_\beta(x_n) &= \sum_{x_{n+1}} \phi_{n+1,n}(x_{n+1}, x_n) \cdot \mu_\beta(x_{n+1}) \end{aligned} \tag{9.6}$$

With base cases

$$\begin{aligned} \mu_\alpha(x_2) &= \sum_{x_1} \phi_{1,2}(x_1, x_2) \\ \mu_\beta(x_{N-1}) &= \sum_{x_N} \phi_{N-1,N}(x_{N-1}, x_N) \end{aligned} \tag{9.7}$$

The time complexity is now only $O(NK^2)$. This is a very efficient method, in part because dynamic programming can also be used to calculate μ_α, μ_β. These μ values correspond to "forward" message and "backward" messages respectively in the chain.

9.2 Message Passing for Inference on Trees

A similar scheme to above can be employed for fast inference on tree graphical models.

Polytree: If the tree structure has nodes with more than 1 parent. However, there must be only one path between each pair of nodes (disregarding edge direction).

9.2.1 Factor Graphs

A factor graph is another method for visualizing graphical models that we will make use of for the message passing algorithm on trees.

- Two node types: Circles for random variable nodes, squares for **factor functions**.
- **Factor functions** represent elements of the factorization for the joint probability function. They connect to each random variable node that is an argument for the particular factor function.
- Note that an undirected graph can be easily converted by introducing one **factor node** per max clique c.
- To convert directed graphs into factor graphs, we first convert to undirected graphs.

Factor graphs are **bipartite** because members of the *factor function* node class only connect with members of the *random variable* node class.

9.2.2 Generalized Message Passing Algorithm

Input: Any undirected tree, directed tree, polytree.

Output: A marginalized joint PDF $P(x_n)$

$$\begin{aligned} p(x_n) &= \sum_{\forall x \in \boldsymbol{x} | x \neq x_n} p(\boldsymbol{x}) \\ &= \prod_{s \in ne(x)} [\sum_{X_s} F_s(x, X_s)] \\ &= \prod_{s \in ne(x)} [\sum_{X_s} \mu_{f_s \to x}] \end{aligned} \tag{9.8}$$

Where $ne(x)$ is the set of factor nodes neighbouring x, X_s is the set of variables in subtree at factor node f_s. $F_s(x, X_s)$ is the product of all the factor nodes in the subtree rooted at f_s (i.e., the probability factor corresponding to the entire subtree at f_s).

1. Convert to factor graph.
2. Initialize each $\mu_{x\to f} = 1$ where $x \in \boldsymbol{x}$ and f is a factor node connected to x.
3. Initialize each $\mu_{f\to x} = f(x)$ where $f(x)$ is the factor function.
4. Begin by computing variable $\to$ factor node messages:

$$\mu_{x\to f}(x) = \prod_{\ell \in Ne(x)|\ell\neq f} \mu_{f_l\to x}(x) \tag{9.9}$$

5. Next, calculate factor $\to$ variable node messages:

$$\mu_{f_s\to x_i}(x_i) = \sum_{\boldsymbol{x}_s|\neq x_i} \phi_{f_s}(\boldsymbol{x}) \cdot \mu_{x_1\to f_s} \cdot \mu_{x_2\to f_s} \cdot \cdots \cdot \mu_{x_N\to f_s} \tag{9.10}$$

   Where $x_{1:N}$ are the children of factor node f_s.

6. Return

$$f(x_n) = \frac{1}{Z} \prod_{s\in ne(x)} \mu_{f_s t\ x}(x) \tag{9.11}$$

This algorithm gives an exact solution for inference and can approximate solutions for graphs with cycles. The max-product algorithm uses similar techniques to solve for $\boldsymbol{x}^* = \arg\max P(\boldsymbol{x})$.

9.2.3 Max Sum Algorithm Sketch

1. Initialize

$$\begin{aligned} \mu_{x\to f}(x) &= 0 \\ \mu_{f\to x}(x) &= \ln f(x) \end{aligned} \tag{9.12}$$

2. Recursion:

$$\begin{aligned} \mu_{f\to x}(x) &= \max_{\boldsymbol{x}=ne(f)} [\ln(f(\boldsymbol{x})) + \sum_{m\in ne(f)|m\neq x} \mu_{x\to f}(x)] \\ \mu_{x\to f}(x) &= \sum_{\ell\in ne(x)|\ell\neq f} \mu_{f_\ell\to x}(x) \end{aligned} \tag{9.13}$$

Chapter 10

Hidden Markov Models

10.1 Introduction

A hidden Markov model is a Markov process with indirect observation of the state. Standard Markovian assumptions still hold, namely that the future depends only on the present state.

Notation:

- Hidden states are represented as z_n for $n \in [N]$. The full sequence is denoted $\boldsymbol{z}$.
- Observations or "emissions" are denoted x_n for $n \in [N]$. The full sequence is denoted $\boldsymbol{x}$.
- We are generally given $P(x|z)$, $P(z_{n+1}|z_n)$ distributions.

Our goal is usually about inferring hidden state from observations. The two sub-problems are (1) estimating a specific hidden state $p(z_n)$ and (2) finding the most likely sequence of states.

$$P(\boldsymbol{x}|\boldsymbol{z}) = \prod_{i=1}^{N} P(x_i|z_i) \tag{10.1}$$

$$P(\boldsymbol{x}, \boldsymbol{z}) = p(z_1)(\prod_{i=2}^{N} P(z_n|z_{n-1}))(\prod_{n=1}^{N} P(x_n|z_n)) \tag{10.2}$$

General Procedures for Estimating $\hat{z}_n$: Given $\boldsymbol{x}$, we can use MLE

$$\hat{z}_n = \arg\max_{z_n} P(z_n) \tag{10.3}$$

Where $P(\boldsymbol{z}, \boldsymbol{x})$ is given as above. We simply must marginalize **efficiently** using the message passing algorithm. When used with HMM's, this is known as the **forward backward algorithm**.

10.2 Forward-Backward Algorithm

Let us convert the directed Markov chain to $h - z_1 - f_2 - f_3 - \cdots - f_{n-1} - z_{n-1} - f_n - z_n - \cdots - z_n$ where

$$h(z_1) = P(z_1)P(x_1|z_1) \tag{10.4}$$

and each z_i representing a hidden node and each f_i representing a factor node. Forward messages are represented as α_n for $n \in [N]$ and backward messages are β_n.

$$\begin{aligned} \alpha(z_{n-1}) &= \mu_{z_{n-1}\to f_n}(z_{n-1}) = \mu_{f_{n-1}\to z_{n-1}}(z_{n-1}) \\ &= P(x_n|z_n)\sum_{z_{n-1}} P(z_n|z_{n-1})\alpha(z_{n-1}) \end{aligned} \tag{10.5}$$

$$\begin{aligned} \beta(z_n) &= \mu_{f_{n+1}\to z_n}(z_n) \\ &= \sum_{z_{n+1}} \beta(z_{n+1})P(x_{n+1}|z_{n+1})P(z_{n+1}|z_n) \end{aligned} \tag{10.6}$$

These are pretty much the same as our $\mu_{\alpha,\beta}$ values from the message passing before.

$$\begin{aligned} P(z_n, \boldsymbol{x}) &= \alpha(z_n)\beta(z_n) \\ P(z_n|\boldsymbol{x}) &= \frac{\alpha(z_n)\beta(z_n)}{P(\boldsymbol{x})} \\ &= \gamma(z_n) \end{aligned} \tag{10.7}$$

Intuition for α, β, γ: Bases can be inferred from these as well.

- $\alpha(z_n) = P(z_n, x_{1:n})$
- $\beta(z_n) = P(x_{(n+1):(N)}|z_n)$
- $\gamma(z_n) = P(z_n|x_{1:N})$

10.2.1 Implementing Forward-Backward Algorithm

α, β can become very small since they are conditioned on $x_{1:n}, x_{n+1:N}$ respectively. Therefore, we introduce the following:

$$\begin{aligned} \hat{\alpha}(z_n) &= \frac{\alpha(z_n)}{P(x_{1:n})} = P(z_n|x_{1:n}) \\ \hat{beta} &= \frac{\beta(z_n)}{P(x_{n+1:N})} \end{aligned} \tag{10.8}$$

Importantly, this corresponds to $\alpha(z_n), \beta(z_n)$ being normalized!

Recursion Formulae: To convert from $\alpha, \beta \to \hat{\alpha}, \hat{\beta}$, just normalize them w.r.t. possible states for z_n!

$$\begin{aligned} \alpha(z_n) &= P(x_n|z_n) \sum_{z_{n-1}} \hat{\alpha}(z_{n-1}) P(z_n|z_{n-1}) \\ \beta(z_n) &= \sum_{z_{n+1}} \hat{\beta}(z_{n+1}) P(x_{n+1}|z_{n+1}) P(z_{n-1}|z_n) \end{aligned} \tag{10.9}$$

Missing observations: Simply marginalize over all possible observations. This corresponds to setting terms $P(x_i|z_i) = 1$.

10.3 Viterbi Algorithm

This algorithm efficiently computes the most likely sequence that resulted in the observations at play.

$$\max_{\boldsymbol{x}} P(\boldsymbol{x}) = \frac{1}{Z} \max_{x_1} \dots \max_{x_N} \phi_{1,2}(x_1, x_2) \phi_{2,3}(x_2, x_3) \dots \phi_{N-1,N}(x_{N-1}, x_N) \tag{10.10}$$

We really just want to apply the "pushing in" principal to the max operators. We can also incorporate the same message passing scheme to further improve speed.

Key Formulae:

$$\begin{aligned}\mu_{x\to f}(x) &= \sum_{\ell\in ne(x)} \mu_{f_\ell\to x}(x) \\ \mu_{f_\ell\to x} &= \max_{x_1,\ldots,x_m} [\log(\phi_\ell(x_1,\ldots,x_m)) + \sum_{m\in ne(f)} \mu_{x_m\to f}(x_m)]\end{aligned} \tag{10.11}$$

It's also generally a good idea to run this algorithm in the log domain (hence the logs above).

Initialization We initialize the Viterbi algorithm from the leaves of the tree.

1. If the leaf is a **factor node**: $\mu_{f\to x}(x) = \log f(x)$.
2. If the leaf is a **varaible node**: $\mu_{x\to f}(x) = \log(1) = 0$

The "upward" messages to the root node are the easiest. The backward messages are more difficult.

The Algorithm We let $w_{n+1}(z_{n+1}) = \mu_{f_{n+1}\to z_{n+1}}(z_n)$

$$w_{n+1}(z_{n+1}) = \max_{z_n}[\ln(P(z_{n+1}|z_n)) + w_n(z_n)] + \ln(P(x_{n+1}|z_{n+1})) \tag{10.12}$$

We run this recursively starting at factor node h. This is our "forward pass". For each node we traverse, we store a "dictionary" relating each $z_{n+1} \to \arg\max_{z_n} \ln(P(z_{n+1})|z_n) + w_n(z_n)$. Then, we trace back starting at z_N using these dictionaries.

Interpretation: $w_n(z_n) = \max_{z_{1:n-1}} \ln(P(x_{1:n}, z_{1:n}))$.