CS 281: Advanced Machine Learning

taught by Sasha Rush

Fall 2017

Contents

CS281: Advanced ML **September 6, 2017**

Lecture 1: Discrete Models

Lecturer: Sasha Rush *Scribes: Anna Sophie Hilgard, Diondra Peck*

- Discrete models take values from a countable set, e.g. {0,1}, {cold, flu, asthma} and are simpler than continuous models.
- We will use simple discrete models to develop our tactics such as marginalization and conditioning.
- Today, we will focus coins as a real-world example.

1.1 Bernoulli model

The likelihood is of the form $p(\text{heads}) = \theta$.

Easy Prior

Assume we know the coin came from one of 3 unknown manufacturers (later, we'll have mixture model estimation, but for now assume these probabilities come from an oracle).

1. $\theta = 0.4$ with probability .1
2. $\theta = 0.5$ with probability .8
3. $\theta = 0.6$ with probability .1

$$p(\theta) = 0.1 \cdot \delta(\theta = 0.4) + 0.8 \cdot \delta(\theta = 0.5) + 0.1 \cdot \delta(\theta = 0.6)$$

Likelihood

Likelihood = $p(\text{data}|\text{parameters})$. For the coin example,

$$p(\text{coin flips}|\theta) = \text{Bin}(N_1|N,\theta) = \binom{N}{N_1}\theta^{N_1}(1-\theta)^{N-N_1} \quad \text{where } N = N_0 + N_1 = \text{number of flips}$$

Note that the last two terms, the "score", is our focus since they are the only terms that depend on θ. The first term normalizes the distribution.

1.2 Inference

Inference 1: $p(\theta|x)$ ($x \in N_0, N_1$). How can we estimate θ?

Maximum Likelihood Estimation (MLE)

$$\theta_{MLE} = \text{argmax}_\theta \; p(N_0, N_1|\theta) = \text{argmax}_\theta \; \log\left[p(N_0, N_1|\theta)\right]$$

$$\theta_{MLE} = \text{argmax}_\theta \; \log\binom{N}{N_1} + N_1\log\theta + N_0\log(1-\theta) \quad \text{Because the first term is not a function of } \theta \text{, we can ignore it.}$$

$$\frac{d}{d\theta} = \frac{N_1}{\theta} + \frac{N_0}{1-\theta}\cdot(-1) \rightarrow \theta_{MLE} = \frac{N_1}{N_0 + N_1}$$

Note that Inference $\neq$ Decision Making. If we asked you to make a bet on the coin, based on this you could either

1. Always take heads if $\theta > .5$. In this case, $p(\text{win}) = \theta$
2. Take heads with probability = θ. In this case, $p(\text{win}) = \theta^2 + (1-\theta)^2$ [p(is heads) * p(choose heads) + ...]

If θ = 0.6, for option 1, p(win) = θ = 0.6. For option 2, p(win) = $\theta^2 + (1-\theta)$ = 0.52. In this case, the additional information used in the calculation of option 2 does not result in a better decision.

Maximizing the Posterior (MAP)

Bayes Rule : $p(\theta|\text{data}) \propto p(\text{data}|\theta)p(\theta)$

- Posterior: $p(\theta|\text{x})$
- Likelihood: $p(\text{x}|\theta)$
- Prior: $p(\theta)$

$$\theta_{MAP} = \text{argmax}_\theta \; p(\theta|\text{x}) = \text{argmax}_\theta \; \log\left[p(\text{x}|\theta)p(\theta)\right] \quad \text{from Bayes' Rule: } p(\theta|\text{x}) \propto p(\text{x}|\theta)p(\theta)$$

Consider an example:

$$p(\theta = 0.4|N_0, N_1) \propto \binom{N}{N_1}(.4)^{N_1}(1-.4)^{N_0}(0.1)$$

$$p(\theta = 0.45|N_0, N_1) = 0 \quad \text{Due to the sparsity of the prior - similar result for } \theta = 0.5 \text{ and } 0.6$$

$\theta_{MAP} = \theta_{MLE}$ when we have a uniform prior since the MLE calculation does not explicitly factor a prior into its calculation.

Full Posterior

Partition or Marginal Likelihood: $p(N_0, N_1) = \int_\theta p(N_0, N_1, \theta)$.

$$p(\theta|N_0, N_1) = \frac{p(x|\theta)p(\theta)}{p(N_0, N_1)} \qquad \text{Note that } p(N_0, N_1) \text{ is a very difficult term to compute.}$$

Beta Prior

$$p(\theta|\alpha_0, \alpha_1) = \frac{\Gamma(\alpha_0 + \alpha_1)}{\Gamma(\alpha_0)\Gamma(\alpha_1)}\theta^{\alpha_1 - 1}(1-\theta)^{\alpha_0 - 1} \qquad \text{support} \in [0, 1]$$

From the image of the beta function for different parameters, we can see that it can ether be balanced, skewed to one side, or tend toward infinity on one side.
With a beta prior:

$$p(\theta|N_0, N_1) = \frac{\Gamma(\alpha_0 + \alpha_1)}{\Gamma(\alpha_0)\Gamma(\alpha_1)}\theta^{\alpha_1 - 1}(1-\theta)^{\alpha_0 - 1}\theta^{N_1}(1-\theta)^{N_0} \cdot \text{(constant normalization term w.r.t } \theta)$$

The key insight is that we get additive terms in the exponent and the resulting distribution looks like another beta. The prior "counts" (pseudocounts) from the hyperparameters can be interpreted as counts we have beforehand.

$$p(\theta|N_0, N_1) = \frac{\Gamma(\alpha_0 + \alpha_1)}{\Gamma(\alpha_0)\Gamma(\alpha_1)}\theta^{N_1 + \alpha_1 - 1}(1-\theta)^{N_0 + \alpha_0 - 1} \cdot \text{(constant normalization term w.r.t } \theta)$$

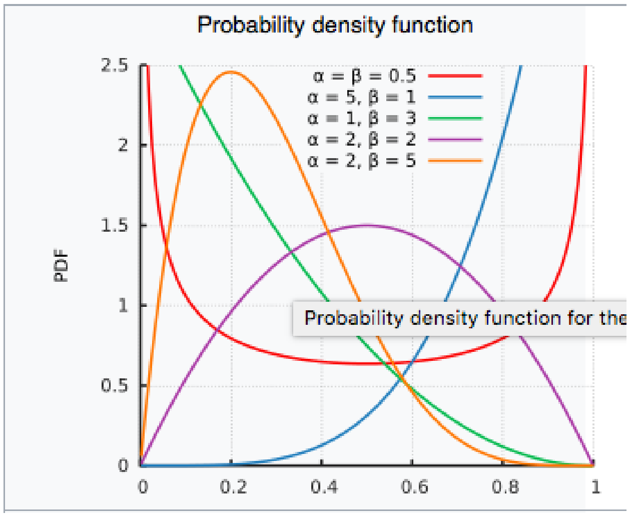


Figure 1.1: Beta Params

To make this distribution sum to 1, use the known beta normalizer

$$p(\theta|N_0, N_1) = \frac{\Gamma(\alpha_0 + \alpha_1 + N_0 + N_1)}{\Gamma(\alpha_0)\Gamma(\alpha_1)\Gamma(N_0)\Gamma(N_1)}\theta^{N_1+\alpha_1-1}(1-\theta)^{N_0+\alpha_0-1} \sim \text{Beta}(\theta|N_0 + \alpha_0, N_1 + \alpha_1) \tag{posterior}$$

The mode of the Beta gives us back θ_{MAP}, but with additional information about the shape of the distribution. What does the prior that tends to infinity at 1 imply? That in the absence of other information, the coin is definitely heads.

Predictive Distribution

$$\begin{aligned} p(\hat{x}|N_0, N_1) &= \int_\theta p(x|\theta, N_0, N_1)p(\theta|N_0, N_1)d\theta \\ &= \int_\theta \theta p(\theta|N_0, N_1)d\theta \\ &= \mathbb{E}_{\theta\sim p(\theta|N_0,N_1)}\theta \end{aligned}$$

This is the expectation under the posterior of θ which is the mean of the Beta distribution. Feel free to prove this as an exercise.

Marginal Likelihood

$$\begin{aligned} p(N_0, N_1) &= \int_\theta p(x_1, \dots x_n|\theta)p(\theta)d\theta \\ &= \int_\theta \frac{\Gamma(\alpha_1 + \alpha_1)}{\Gamma(\alpha_0)\Gamma(\alpha_1)}\theta^{\alpha_1+N_1-1}(1-\theta)^{\alpha_0+N_0-1} \end{aligned}$$

The first term can be moved outside, as it does not depend on θ. After introducing our normalization term and making the distribution insidesum to 1,

$$p(N_0, N_1) = \frac{\Gamma(\alpha_1 + \alpha_1)}{\Gamma(\alpha_0)\Gamma(\alpha_1)}\frac{\Gamma(N_0 + \alpha_0)\Gamma(N_1 + \alpha_1)}{\Gamma(N_0 + N_1 + \alpha_0 + \alpha_1)}$$

1.3 Extensions on the Coin Flip Model: Super Coins

- Many correlated coins: models of binary data, important for discrete graphical models
- Many-sided coins aka dice: models of categorical data, generalization of Bernoulli

1.4 Other Distributions

$$\begin{aligned}
\text{Bernoulli}(x|\theta) &= \theta^x(1-\theta)^x \\
\text{Categorical}(x|\theta) &= \prod_k \theta_k^{x_k} && \text{generalization of Bernoulli} \\
\text{Multinomial}(x|\theta) &= \frac{(\sum x_k)!}{\prod_k x_k!}\prod_k \theta_k^{x_k} && \text{generalization of Binomial} \\
\text{Dirichlet}(x|\alpha) &= \frac{\Gamma(\sum \alpha_k)}{\prod \Gamma(\alpha_k)}\prod_k \theta_k^{\alpha_k - 1} && \text{generalization of Beta, often used as a prior}
\end{aligned}$$

Note that the Dirichlet distribution is the conjugate prior of the Categorical and Multinomial distributions.

1.5 Example notebook

See Beta.ipynb

CS281: Advanced ML **September 11, 2017**

Lecture 3: Multivariate Normal Distributions

Lecturer: Sasha Rush *Scribes: Christopher Mosch, Lindsey Brown, Ryan Lapcevic*

3.6 Examples

Multivariate gaussians are used for modeling in various applications, where knowing mean and variance is useful:

- radar: mean and variance of approaching objects (like invading aliens)
- weather forecasting: predicting the position of a hurricane, where the uncertainty in the storm's position increases for timepoints farther away
- tracking the likely outcome of a sports game: last year's superbowl is an example of a failure of modeling with multivariate gaussians as the Patriots still won after a large Falcons' lead

3.7 Review: Eigendecomposition

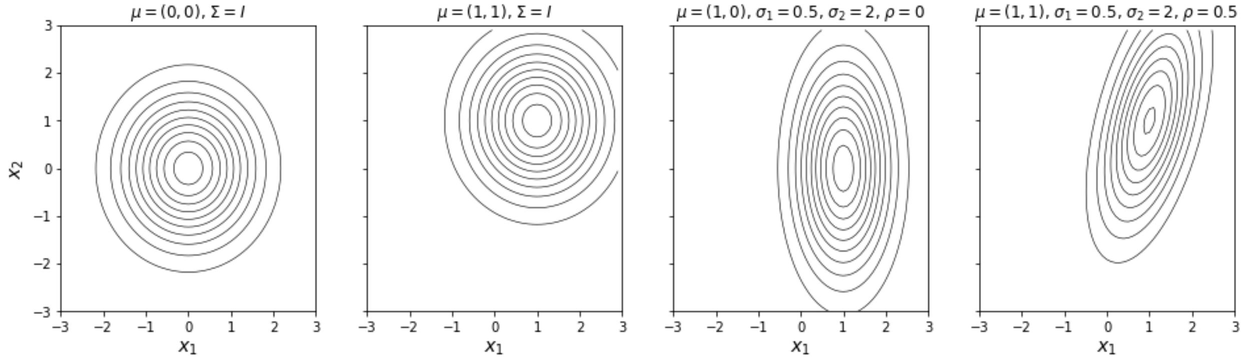
Let $\mathbf{\Sigma}$ be a square, symmetric matrix. Then its eigendecomposition is given by $\mathbf{\Sigma} = \boldsymbol{U}^T \mathbf{\Lambda} \boldsymbol{U}$, where $\boldsymbol{U}$ is an orthogonal matrix and $\mathbf{\Lambda}$ is a diagonal matrix. In the special case that $\mathbf{\Sigma}$ is positive semidefinite (as is the case for covariance matrices), denoted $\mathbf{\Sigma} \succeq 0$, all its eigenvalues are nonnegative, $\Lambda_{ii} \geq 0$, and we can decompose its inverse as $\mathbf{\Sigma}^{-1} = \boldsymbol{U}^T \mathbf{\Lambda}^{-1} \boldsymbol{U}$, where $\Lambda_{ii}^{-1} = 1/\Lambda_{ii}$.

3.8 Multivariate Normal Distributions (MVNs)

Let X be a D-dimensional MVN random vector with mean $\boldsymbol{\mu}$ and covariance matrix $\mathbf{\Sigma}$, denoted $X \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$. Then the pdf of X is

$$p(x) = (2\pi)^{-D/2} \, |\mathbf{\Sigma}|^{-1/2} \exp{[-\frac{1}{2}(x-\boldsymbol{\mu})^T \mathbf{\Sigma}^{-1}(x-\boldsymbol{\mu})]},$$

where for many problems we focus on the quadratic form $(x-\boldsymbol{\mu})^T \mathbf{\Sigma}^{-1}(x-\boldsymbol{\mu})$ (which geometrically can be thought of as distance) and ignore the normalization factor $(2\pi)^{-D/2} \, |\mathbf{\Sigma}|^{-1/2}$. The figure below plots the contours of a bivariate Normal for various $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$ (in the figure, ρ denotes the off-diagonal elements of $\mathbf{\Sigma}$, given by the covariance of x_1 and x_2).



Note that we can decompose $\mathbf{\Sigma}$ as

$$\begin{aligned}\mathbf{\Sigma} &= (x-\boldsymbol{\mu})^T\mathbf{\Sigma}^{-1}(x-\boldsymbol{\mu}) \\ &= (x-\boldsymbol{\mu})^T\left(\boldsymbol{U}^T\mathbf{\Lambda}^{-1}\boldsymbol{U}\right)(x-\boldsymbol{\mu}) \\ &= (x-\boldsymbol{\mu})^T\left(\sum_d \frac{1}{\lambda_d}U_dU^T\right)(x-\boldsymbol{\mu}) \\ &= \sum_d \frac{1}{\lambda_d}(x-\boldsymbol{\mu})^T U_d U_d^T(x-\boldsymbol{\mu}),\end{aligned}$$

where $(x-\boldsymbol{\mu})^T U_d$ can be interpreted as the projection of $(x-\boldsymbol{\mu})$ onto U_d (which can each be thought of as univariate gaussians), the eigenvector corresponding to the eigenvalue λ_d. Since $\mathbf{\Sigma}$ is the weighted sum of the dot product of such projections (with weights being given by $1/\lambda_d$, which can be thought of as the scale $1/\sigma^2$), we can describe the MVN as tiling of univariates.

3.8.1 Manipulating MVNs: Stretches, Rotations, and Shifts

Let $x \sim \mathcal{N}(0,\boldsymbol{I})$ and $y = \boldsymbol{A}x + \boldsymbol{b}$. We want to consider two ways of obtaining the complete distribution of y.

- 'Overkill': We can perform a change of variables[1]. Here, we have $x = \boldsymbol{A}^{-1}$ and $|dx/dy| = |A^{-1}|$, leading to

  $$\begin{aligned}p(y) &= \mathcal{N}\left(\boldsymbol{A}^{-1}(y-\boldsymbol{b})|0,\boldsymbol{I}\right)|\boldsymbol{A}^{-1}| \\ &= \frac{1}{z}\exp\left[(\boldsymbol{A}^{-1}(y-\boldsymbol{b}))^T(\boldsymbol{A}^{-1}(y-\boldsymbol{b}))\right] \\ &= \frac{1}{z}\exp\left[(y-\boldsymbol{b})^T(\boldsymbol{A}^{-1})^T(\boldsymbol{A}^{-1})(y-\boldsymbol{b})\right] \\ &= \mathcal{N}(y|\boldsymbol{b},\boldsymbol{A}\boldsymbol{A}^T),\end{aligned}$$

  where z is the normalizing constant.

- Using the properties of MVN, we know that y is also MVN, so is completely specified by its mean and covariance matrix which can easily be derived,

  $$\mathbb{E}(y) = \mathbb{E}(\boldsymbol{A}x+\boldsymbol{b}) = \boldsymbol{A}\mathbb{E}(x)+\boldsymbol{b} \qquad \text{cov}(y) = \boldsymbol{A}\boldsymbol{A}^T.$$

Thus, we can generate MVN from $\mathcal{N}(0,\boldsymbol{I})$ via the transformation $y = \boldsymbol{A}x+\boldsymbol{b}$, where we set $\boldsymbol{A} = \boldsymbol{U}\mathbf{\Lambda}^{1/2}$, leading to $\mathbf{\Sigma}_Y = \boldsymbol{U}^T\mathbf{\Lambda}\boldsymbol{U}$. Then shifts are represented by $\boldsymbol{b}$, stretches by $\mathbf{\Lambda}$, and rotations by $\boldsymbol{U}$.

3.8.2 Detour: MVN in High-Dimensions ($D \gg 0$)

Let x be a D-dimensional random vector, distributed as $\mathcal{N}(0, I/D)$, where I is the identity. The expected length of x is given by

$$\mathbb{E}\left(\|x\|^2\right) = \mathbb{E}\left(\sum_d x_d^2\right) = D\sigma_d^2 = 1,$$

which means that x is expected to be on the boundary of a unit sphere centered at the origin. Moreover, the variance of the length is

$$\text{var}\left(\|x\|^2\right) = D\cdot\left(\mathbb{E}(x^4) - \mathbb{E}\left(x^2\right)^2\right) = D\cdot(3\sigma^4-\sigma^4) = 2D/D^2 = 2/D$$

[1] A change of variables can be done in the following way: Let $y = f(x)$ and assume f is invertible so that $x = f^{-1}(y)$. Then $p(y) = p(x)|dx/dy|$. This is a technique which will be used often in this course

Thus, it is not only expected that x lies on the boundary but as D increases most of its realizations will in fact fall on the boundary[2].

3.8.3 Key Formulas for MVN: Marginalization and Conditioning

Let $X \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with

$$X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}, \quad \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix}.$$

Note that Σ is written in block matrix form, rather than scalar entries. It turns out that the marginals, X_1 and X_2, are also MVN, and their mean and covariance matrice are given by $\boldsymbol{\mu}_1$ and $\boldsymbol{\Sigma}_{11}$ and $\boldsymbol{\mu}_2$ and $\boldsymbol{\Sigma}_{22}$ respectively. A sketch of the proof is provided below.

$$p(x_1) = \int_{x_2} N(x|\boldsymbol{\mu}, \boldsymbol{\Sigma}) dx_2,$$

which can be written as

$$0.5 \int_{x_2} \exp\left[(x_1 - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_{11}^{-1}(x_1 - \mu_1) + 2(x_1 - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_{12}^{-1}(x_2 - \mu_2) + (x_2 - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}_{22}^{-1}(x_2 - \mu_2)\right] dx_2.$$

Note that this equals

$$p(x_1) \int_{x_2} p(x_2|x_1) dx_2,$$

implying that $X_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11})$.

While the marginals have a simple form, the conditionals are more complicated. (For a complete derivation, which requires matrix inversion lemmas, refer to Murphy.) It can be shown that $X_1|X_2 \sim \boldsymbol{\mu}_{\infty|\in}, \boldsymbol{\Sigma}_{\infty|\in}$ with

$$\boldsymbol{\mu}_{1|2} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(x_2 - \boldsymbol{\mu}_2), \quad \boldsymbol{\Sigma}_{1|2} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}.$$

3.8.4 Information Form

An alternative formulation, called information form, uses the precision matrix (inverse variance) $\Lambda = \Sigma^{-1}$. Partitioning Λ as

$$\boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12} \\ \boldsymbol{\Lambda}_{21} & \boldsymbol{\Lambda}_{22} \end{pmatrix},$$

the covariance matrices of the conditional distributions have a simple form. For example, the covariance matrix of X_1 given X_2 is $\Lambda_{1|2} = \Lambda_{11}$. However, the simplicity of the conditional precision comes at the cost of marginalization (which was simple when using Σ) becoming a more complicated expression (see Murphy subsection 4.3 for more details).

[2] It is left as an exercise to show that this formula holds. Hint: Use the fact that we assumed no covariance.

CS281: Advanced ML **September 13, 2017**

Lecture 4: Linear Regression

Lecturer: Sasha Rush *Scribes: Kojin Oshiba, Michael Ge, Aditya Prasad*

4.9 Multivariate Normal (MVN)

The multivariate normal distribution of a D-dimensional random vector X is defined as:

$$N(X|\mu,\Sigma) \sim (2\pi)^{-\frac{D}{2}}|\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right)$$

Note:

- $(2\pi)^{-\frac{D}{2}}|\Sigma|^{-\frac{1}{2}}$ and $-\frac{1}{2}$ are constants we can ignore in MLE and MAP calculations.
- $(X-\mu)^T\Sigma^{-1}(X-\mu)$ is a quadratic term.

There are three types of inference we're interested in doing: MLE, MAP, and prediction.

4.10 Maximum Likelihood of MVN

Let $\theta = (\mu, \Sigma)$, where Σ can be approximated as a diagonal/low rank matrix. If there are $x_1, \ldots, x_n$ observations, the MLE estimate of μ is

$$\begin{aligned}\mu^* &= \arg\max_{\mu} - \sum_n logN(x_n|\mu,\Sigma)\\ &= \arg\max_{\mu} \quad log(constant) - \sum_n (x_n-\mu)^T\Sigma^{-1}(x_n-\mu)\\ &= \arg\max_{\mu} - \sum_n (x_n-\mu)^T\Sigma^{-1}(x_n-\mu)\end{aligned}$$

Let $L = \sum_n (x_n-\mu)^T\Sigma^{-1}(x_n-\mu)$.

$$\frac{dL}{d\mu} = \Sigma_n\Sigma^{-1}(x_n-\mu) = 0$$

$$\Leftrightarrow \mu^*_{MLE} = \frac{\Sigma_{X_n}}{N}$$

Similarly,

$$\frac{dL}{d\Sigma} = (exercise) = 0$$

$$\Leftrightarrow \Sigma^*_{MLE} = \frac{1}{N}\sum_n x_n x_n^T = \frac{1}{N}X^\top X$$

For calculating $\frac{dL}{d\Sigma}$ as an exercise , the following might be helpful:

- $\frac{d}{dA}ln|A| = A^{-1}$
- $\frac{d}{dA}tr(BA) = B^T$
- $tr(ABC) = tr(BCA)$

4.11 Linear-Gaussian Models

Let x be a vector of affine, noisy observations with a prior distribution:

$$x \sim N(m_0, S_0)$$

Let y be the outputs:

$$y|x \sim N(Ax + b, \Sigma_y)$$

4.11.1 $p(x|y)$

We are interested in calculating the posterior distribution: $p(x|y)$.

$$\begin{aligned} p(x|y) &\propto p(x)p(y|x) \\ &= \frac{1}{2}\exp\begin{cases} (x - m_0)^\top S_0^{-1}(x - m_0) \\ +(y - (Ax + b))^\top \Sigma_y^{-1}(y - (Ax + b)) \end{cases} \\ &= \frac{1}{2}\exp\begin{cases} \underline{x^\top S_0^{-1} x}^{\star\star} - \underline{2x^\top S_0^{-1} m_0}^{\star} + \ldots \\ +\underline{x^\top (A^\top \Sigma_y^{-1} A) x}^{\star\star} - \underline{2x^\top (A^\top \Sigma_y^{-1}) y}^{\star} + \underline{2x^\top (A^\top \Sigma_y^{-1}) b}^{\star} + \ldots \end{cases} \end{aligned}$$

The terms containing x are underlined. Double-starred ($\star\star$) terms are quadratic in x, while single-starred ($\star$) terms are linear in x. The remaining terms are constants that are swallowed up by the proportionality. By Gaussian-Gaussian conjugacy, we know the resulting distribution should be Gaussian. To find the parameters, we'll modify $p(x|y)$ to fit the form of a Normal. This requires completing the square!

4.11.2 Completing the Square

$$ax^2 + bx + c \to a(x - h)^2 + k, h = \frac{-b}{2a}, k = c - \frac{b^2}{4a}$$

We ignore the k term since it too is swallowed up in the proportionality. In application to our problem, we group the quadratic and linear terms together to calculate our terms for completing the square.

- "a" is $S_N^{-1} = S_0^{-1} + A^\top \Sigma_y^{-1} A$
- "h" is $m_N = S_N \left[S_0^{-1} m_0 + A^\top \Sigma_y^{-1}(y - b) \right]$

In this more "intuitive" representation, we find that $p(x|y)$ has the form of $N(m_N, S_N)$. Murphy also has a more explicit representation:

- $\Sigma_{x|y} = \Sigma_x^{-1} + A^\top \Sigma_y^{-1} A$
- $\mu_{x|y} = \Sigma_{x|y}[\Sigma_x^{-1}\mu_x + A^\top \Sigma_x^{-1}(y - b)]$

4.11.3 p(y)

We now calculate the normalizer term, $p(y)$. Now, x is fixed. y follows the linear model:

$$y = Ax + b + \epsilon$$

The result is that y follows a Normal distribution with the following form:

$$p(y) = N(y|Am_0 + b, \Sigma_y + A\Sigma_x A^\top)$$

4.11.4 Prior (just for μ)

$$p(\mu) = N(\mu|m_0, S_0)$$

where m_0, S_0 are pseudo mean, pseudo variance. $p(\mu)$ is defined Gaussian because Gaussian is the conjugate prior of itself. A prior is called a conjugate prior if it has the same distribution as the posterior distribution.

4.11.5 Posterior (just for μ)

$$p(\mu|X) \propto p(\mu)p(X|\mu) = N(\mu|m_0, s_0)N(X|\mu, \Sigma)$$

This is a special case of linear regression. Recall,

- "a" is $S_N^{-1} = S_0^{-1} + A^\top \Sigma_y^{-1} A$
- "h" is $m_N = S_N \left[S_0^{-1} m_0 + A^\top \Sigma_y^{-1} (y - b) \right]$

We let $b = 0$ and $A = I$. Then we obtain,

$$S_N^{-1} = S_0^{-1} + \Sigma^{-1}$$

$$m_N = S_N [S_0^{-1} m_0 + \Sigma^{-1} X]$$

Hence,

$$p(\mu|X) = N(m|m_N, S_N)$$

4.11.6 Unknown Variance

Similar to μ, we can also define a conjugate prior on Σ, which is Inverse Wishart distribution. It is defined as:

$$IW(\Sigma|S, \nu) = \frac{1}{2} |\Sigma|^{-(\nu-(D+1)/2)} exp\{\frac{1}{2} tr(S^{-1}\Sigma^{-1})\}$$

- distribution over positive semi definite Σ with two parameters S, ν.
- pseudo info $S = \Sigma X X^T$ is a psuedo scatter matrix called the scale matrix. $\nu = n\mu$ is degrees of freedom where $\nu - (D+1)$ is the number of observations.

4.12 Linear Regression

In an undergraduate version of the class, we might define the problem as follows: We are given "fixed" set of inputs, $\{\boldsymbol{x}_i\}$. We want to "predict" the outputs.

Here, we define the problem as attempting to compute $p(y \mid \boldsymbol{x}, \boldsymbol{\theta})$. Consider the following example. We assume that our data is generated as follows:

$$y = \boldsymbol{w}^T \boldsymbol{x} + \text{noise}$$

Further, we assume that the noise (denoted by ϵ) is distributed as Gaussian with mean 0; that is:

$$\epsilon \sim \mathcal{N}(0, \sigma^2)$$

Then, we have:

$$p(y \mid \boldsymbol{x}, \theta) = \mathcal{N}(y \mid \boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x}, \sigma^2)$$

Note that the bias term here is included as a dimension in $\boldsymbol{w}$, w_0.

4.12.1 Log Likelihood

Consider a data-set that looks like $\{(\boldsymbol{x_i}, y_i)\}_{i=1}^N$. The log-likelihood $\mathcal{L}(\boldsymbol{\theta})$ is given by:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \log p(\text{data} \mid \boldsymbol{\theta}) \\ &= \sum_{n=1}^{N} \log p(y_n \mid \boldsymbol{x_n}, \boldsymbol{\theta}) \\ &= \sum_{n=1}^{N} \log(\text{constant}) - \frac{1}{2\sigma^2}(y_n - \boldsymbol{w}^T \boldsymbol{x_n})^2\end{aligned}$$

Note that data here refers to just the y_i's. The y_n's are called the target; the $\boldsymbol{w}$ represents the weights; and the x_n's are the observations. The term $(y_n - \boldsymbol{w}^T \boldsymbol{x_n})^2$ is essentially just the residual sum of squares.

4.12.2 Computing MLE

We want the argmax of the log-likelihood. We therefore have:

$$\begin{aligned}\text{argmax}_w \mathcal{L}(\boldsymbol{w}) &= \text{argmax} - \sum_{n=1}^{N} \frac{1}{2\sigma^2}(y_n - \boldsymbol{w}^T \boldsymbol{x_n})^2 \\ &= \text{argmax}_w - [\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}]^T [\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}] \\ &= \text{argmax}_w \left[\boldsymbol{w}^T \boldsymbol{X^T} \boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{w}^T \boldsymbol{X^T} \boldsymbol{y} + \text{constant}\right]\end{aligned}$$

There is an analytical solution to this, and we obtain it by simply computing the gradient and setting it to $\mathbf{0}$.

$$\partial_w \left[\boldsymbol{w}^T \boldsymbol{X^T} \boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{w}^T \boldsymbol{X^T} \boldsymbol{y}\right] = 2\boldsymbol{X^T}\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{X^T}\boldsymbol{y}$$

Setting this to $\mathbf{0}$, we obtain:

$$\boldsymbol{w}_{MLE} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

As we will see in homework 1, $(\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$ can be viewed as the projection of $\boldsymbol{y}$ onto the column space of $\boldsymbol{X}$.

4.13 Bayesian Linear Regression

In the Bayesian framework, we also introduce a probability distribution on the weights. Here, we choose:

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{m_0}, \boldsymbol{S_0})$$

Thus, we have:

$$p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \boldsymbol{\mu}, \sigma^2) = \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{\mu} + \boldsymbol{X^T}\boldsymbol{w}, \sigma^2 \boldsymbol{I})$$

We assume that $\boldsymbol{\mu} = 0$.

The posterior then is of the form:

$$p(\boldsymbol{w} \mid \ldots) \propto \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{m_0}, \boldsymbol{S_0}) \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{X}^T \boldsymbol{w}, \sigma^2 \boldsymbol{I})$$

Applying the results obtained above with the linear Gaussian results, with:

$$\begin{aligned}b &= 0 \\ \boldsymbol{A} &= \boldsymbol{X}^T \\ \boldsymbol{\Sigma_y} &= \sigma^2 \boldsymbol{I}\end{aligned}$$

Thus, we have:

$$\boldsymbol{S_N}^{-1} = \boldsymbol{S_0}^{-1} + \frac{1}{\sigma^2}\boldsymbol{X^T X}$$
$$\boldsymbol{m_N} = \boldsymbol{S_N}\left[\boldsymbol{S_0^{-1} m_0} + \boldsymbol{X^T y}\frac{1}{\sigma^2}\right]$$

Now, we compute the posterior predictive:

$$p(y \mid \boldsymbol{x}, y) = \int \mathcal{N}(y \mid \boldsymbol{w}^T\boldsymbol{x}, \sigma^2)\mathcal{N}(\boldsymbol{w} \mid \boldsymbol{m_N}, \boldsymbol{S_N})d\boldsymbol{w}$$

Using the form for the marginal derived earlier, we have:

$$p(y \mid \boldsymbol{x}, y) = \mathcal{N}(y \mid \boldsymbol{X^T m_N}, \sigma^2 + \boldsymbol{X^T S_0 X})$$

The variance term is particularly interesting because now the variance has dependence on the actual data; thus, the Bayesian method has thus produced a different result. The mean, however, is the same as the MAP estimate ($\boldsymbol{x}^T\boldsymbol{m_N}$)

4.14 Non Linear Regression

All the examples done so far have been in linear space. To define an adaptive basis, we simply transform point x with the transormation of our choice:

$$x \to \phi(x)$$

Examples include:

- $\phi_1(x) = \sin(x)$
- $\phi_2(x) = \sin(\lambda x)$
- $\phi_3(x) = \max(0, x)$
- $\phi(x; w) = \max(0, w' \top x)$

The last example is the core of neural networks and deep learning where the weights are learned for each level of w.

CS281: Advanced ML **September 18, 2017**

Lecture 5: Linear Classification

Lecturer: Sasha Rush *Scribes: Demi Guo, Artidoro Pagnoni, Luke Melas-Kyriazi, Florian Berlinger*

5.15 Classification Introduction

Last time we saw linear regression. In linear regression we were predicting $y \in \mathbb{R}$, in classification instead we deal with a discrete set, for example $y \in \{0, 1\}$ or $y \in \{1, \dots, C\}$. This distinction only matters for this lecture, starting from next class we will generalize the topics and treat them as the same thing.

Among the many applications, linear classification is used in sentiment analysis, spam detection, and facial and image recognition. We will use generative models of the data, which means that we will model both the x and the y explicitly, and we are not keeping x fixed. In the case of the spam filter earlier, x is the email body, and y is the label {spam, not spam}. A generative model of the email and labels, we would model the distribution of x, of the text in the email itself, and not only the distribution of the category y.

We will explore the basic method of Naïve Bayes in detail. Even with a very simple method like Naïve Bayes with basic features it is possible to perform extremely well on many classification tasks when large training data sets are available. For example, this simple model performs almost as well (one percent point difference) as very complex methods on spam detection.

5.16 Naïve Bayes

Note that the term ”Bayes” in Naïve Bayes (NB) does not have to do with Bayesian modeling, or the presence of priors on parameters. We won’t have any priors for the moment. General Naïve Bayes takes the following form:

$$\begin{aligned} y &\sim \text{Cat}(\pi) && \text{[class distribution]} \\ \boldsymbol{x} | y &\sim \prod_j p(x_j \mid y) && \text{[class conditional]} \end{aligned}$$

where y is the class label and comes from a categorical distribution, and x_j is a dimension of the input x.

In Naïve Bayes, the form of the class distribution is fixed and parametrized independently from the class conditional distribution. The ”Naïve” term in ”Naïve Bayes” precisely refers to the conditional independence between y and $x_j|y$. Depending of what the data looks like we can choose a different form for the class conditional distribution.

Here we present three possible choices for the class conditional distribution:

- **Multivariate Bernoulli Naïve Bayes**:

  $$x_j | y \sim \text{Bern}(\mu_{jc}) \quad \text{if } y = c$$

  Here y takes values in a set of classes, and μ_{jc} is a parameter associated with a specific feature (or dimension) in the input and a specific class. We use multivariate Bernoulli when we only allow two possible values for each feature, therefore $x_j|y$ follows a Bernoulli distribution.

  We can think of x as living in a hyper cube, with each dimension j having an associated μ for each class c. From here we get the name multivariate Bernoulli distribution.

- **Categorical Naïve Bayes**:

  $$x_j | y \sim \text{Cat}(\mu_{jc}) \quad \text{if } y = c$$

  We use the Categorical Naïve Bayes when we allow different classes for each feature j, so $x_j|y$ follows a Categorical distribution.

- **Multivariate Normal Naïve Bayes**

  $$\boldsymbol{x}|y \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}^c_{diag})$$

  Note that here we use $\boldsymbol{x}$ vector and not a specific feature. Since we impose that $\boldsymbol{\Sigma}^c$ is a diagonal matrix, we have no covariance between features, so this comes down to having an independent normal for each feature (or dimension) of the output. This is also required by the "Naïve" assumption of conditional independence. We would use MVN Naïve Bayes when the features take continuous values in $\mathbb{R}^n$.

5.17 General Naïve Bayes

We consider the data points $\{(\boldsymbol{x}_n, y_n)\}$, without specifying a particular generative model. The likelihood of each data point is:

$$\begin{aligned} p(\boldsymbol{x}_n, y_n|\text{param}) &= p(y_n|\boldsymbol{\pi}) \prod_j p(x_{nj}|y_n, \text{param}) && (5.1) \\ &= \prod_c \pi_c^{(y_n=c)} \prod_j \prod_c p(x_{nj}|y_n)^{(y_n=c)} && (5.2) \end{aligned}$$

where in equation (5.2) we assume conditional independence (the "Naïve" assumption). The term $p(x_{nj}|y_n)$ depends on the generative model used for x and also on the class y_n.

We can then solve for the parameters maximizing the likelihood, which is equivalent to maximizing the log likelihood.

$$\begin{aligned} (\pi_{\text{MLE}}, \mu_{\text{MLE}}) &= \underset{(\boldsymbol{\pi},\boldsymbol{\mu})}{\text{argmax}} \sum_n \log p(\boldsymbol{x}_n, y_n|\text{param}) && (5.3) \\ &= \underset{(\boldsymbol{\pi},\boldsymbol{\mu})}{\text{argmax}} \sum_c N_c \log \pi_c + \sum_i \sum_c \sum_{n:y_n=c} \log p(x_{nj}|y_n) && (5.4) \\ &= \left(\underset{(\boldsymbol{\pi},\boldsymbol{\mu})}{\text{argmax}} \sum_c N_c \log \pi_c \right) + \left(\underset{(\boldsymbol{\pi},\boldsymbol{\mu})}{\text{argmax}} \sum_i \sum_c \sum_{n:y_n=c} \log p(x_{nj}|y_n) \right) && (5.5) \end{aligned}$$

Where $N_c = \sum_n \mathbb{1}(y_n = c)$, and $N =$ the number of data points.

This factors into two parts (5.10), the first only depending on $\boldsymbol{\pi}$ the other is the MLE for the class condition distribution on each feature or dimension of the input. This factorization allows to solve for the maximizing $\boldsymbol{\pi}$ and the maximizing parameters for the class conditional separately.

For example, if we use a Multivariate Bernoulli Naïve Bayes generative model we would get the following parameters from MLE:

$$\begin{aligned} \pi_c &= \frac{N_c}{N} && (5.6) \\ \mu_{jc} &= \frac{\sum_{n:y_n=c} x_{nj}}{N_c} = \frac{N_{cj}}{N_c} && (5.7) \end{aligned}$$

Again, where $N_c = \sum_n \mathbb{1}(y_n = c)$, $N_{cj} = \sum_n \mathbb{1}(y_n = c)x_{nj}$ and $N =$ number of data points.

5.18 Bayesian Naive Bayes: Add a Prior

Here, instead of working with a single distribution, we are working with multiple distributions. For simplicity, let's use the following factored **prior**:

$$p(\pi, \mu) = p(\pi) \prod_j \prod_c p(\mu_{jc})$$

where $p(\pi)$ represents the prior on class distribution and $\prod_j \prod_c p(\mu_{jc})$ represents prior on class conditional distribution.

Now, **what prior should we use?**

1. π: Dirichlet (goes with Categorical)
2. μ_{jc}:
    (a) Beta (goes with Bernoulli)
    (b) Dirichlet (goes with Categorical)
    (c) Normal (goes with Normal)
    (d) Inverse-Wishart (Iw) (goes with Normal)

    Here, what distribution we choose depends on our choice of class conditional distribution.

Recall that we want to use conjugate priors to have a natural update (that's why we pair them up!). By using conjugate priors, we will have:

$$p(\pi|\text{data}) = Dir(N_1 + \alpha_1, \cdots, N_c + \alpha_c)$$
$$p(\mu_{jc}|\text{data}) = \beta((N_c - N_{jc}) + \beta_0, N_c + \beta_1)$$

5.18.1 Intuition

You can think of the α_i above as initial pseudocounts. Those pseudocounts give nonzero probability to features we haven't seen before, which is crucial for NLP. For unseen features, you could have a pseudocount of 1 or 0.5 (Laplace term) or something.

Because of this property, a Bayesian model helps prevents overfitting by introducing such priors: consider the spam email classification problem mentioned before. Say the word "subject" (call it feature j) always occurs in both classes ("spam" and "not spam"), so we estimate $\hat{\theta}_{jc} = 1$ (we overfit!) What will happen if we encounter a new email which does not have this word in it? Our algorithm will crash and burn! This is another manifestation of the black swan paradox discussed in Book Section 3.3.4.1. Note that this will not happen if we introduce pseudocounts to all features!

5.19 Posterior Predictive

$$p(\hat{y}, \hat{x} \mid \text{data}) = \text{(integrate over parameters)}$$
$$\pi_c{}^{\text{MAP}} = \frac{N_c + \alpha_c}{N + \sum_c \alpha_c} \text{(Dirichlet MAP)}$$
$$\mu_{jc}{}^{\text{MAP}} = \frac{N_{jc} + \beta_1}{N_c + \beta_1 + \beta_0} \text{(Beta MAP)}$$

(How to derive this? Good exercise!)

5.20 More on Predictive

Now, let's consider a little bit more about what's happening in our predictive. Consider the email spam classification problem: given some features of an email, we want to predict if the email is a spam or not a spam. We have:

$$\begin{aligned} p(y = c|x, data) &\propto \pi_c \prod_j p(x_j|y) \text{ (try to generate observations from class)} \\ &= \pi_c \prod_j \mu_{jc}^{x_j}(1 - \mu_{jc})^{(1-x_j)} \text{ (informal parametrization)} \\ &= \exp(\log \pi_c + \sum_j x_j \log \mu_{jc} + (1 - x_j)\log(1 - \mu_{jc})) \text{ (take exp of log)} \\ &= \exp\left(\log \pi_c + \sum_j \log(1 - \mu_{jc}) + \sum_j x_j \log \frac{\mu_{jc}}{1 - \mu_{jc}}\right) \end{aligned}$$

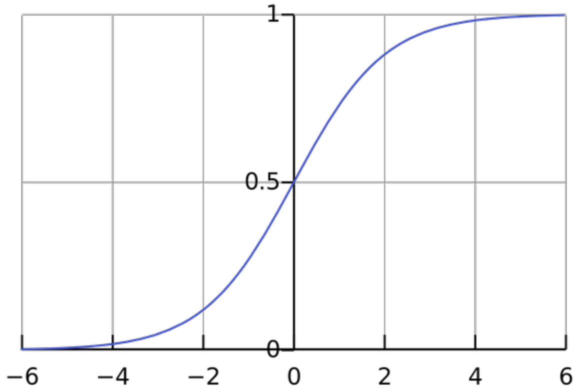


Figure 5.2: The Sigmoid Function

where the first two term $log\pi_c + \sum_j log(1-\mu_{jc})$ is a constant (we call it b for bias), and the last term $\sum_j x_j log\frac{\mu_{jc}}{1-\mu_{jc}}$ is linear (we call it θ).

5.21 Multivariate Bernoulli Naive Bayes

For Multivariate Bernoulli NB, we will have:

$$\theta_{jc} = \log\frac{\mu_{jc}}{1-\mu_{jc}}$$
$$b_c = \log\pi_c + \log(1-\mu_{jc})$$

So, we have:

$$p(y=c \mid x) \propto \exp(\theta_c^T x + b_c)$$

Thus, in order to determine which class (”spam” or ”not spam”), for each class we simply compute a linear function with respect to x, and compare the two. Our $\theta x + b$ is going to be associated with a linear separator of the data. Even better, for prediction, we can simply compute θ and β (as shown above) using closed form for both MAP and MLE cases.

5.22 The Sigmoid Function

Before proceeding, we should name our variables to speak about them more easily.

We call μ the ”informal parameters” and θ the ”scores.” In the case of a Multivariate Bernouli model, we have the map $\theta_{jc} = \log\frac{\mu_{jc}}{1-\mu_{jc}}$, which we call the ”log odds.” We may also invert this relationship to find μ as a function of θ:

$$\theta = \log\frac{\mu}{1-\mu} \implies \mu = \frac{e^\theta}{1+e^\theta} = \frac{1}{1+e^{-\theta}} = \sigma(\theta)$$

We denote this function $\sigma(\theta)$ as the sigmoid function. The sigmoid function is a map from the real line to the interval $[0,1]$, so is useful as a representation of probability. It is also a common building block in constructing neural networks, as we will see later in the course.

5.23 The Softmax Function

We will now return to the predictive $p(y=c|x) \propto \exp(\theta_c^T x + c_c)$ to try to compute the normalizer Z:

$$p(y=c|x) = \frac{1}{Z}\exp(\theta_c^T x + b_c)$$

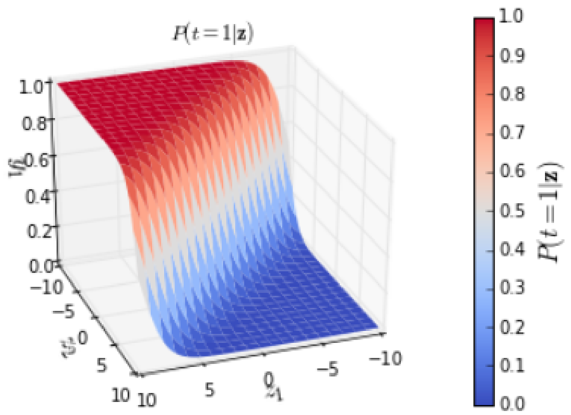


Figure 5.3: The Softmax Function

In general, we can compute the normal by summing over all our classes.

$$Z(\theta) = \sum_{c'} = \exp(\theta_{c'}^T x + b_{c'})$$

In practice, this summation is often computationally expensive. However, it is not necessary to compute this sum if we are only interested in the most likely class label given an input.

We call the resulting probability density function the *softmax*:

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_{i'} \exp(z_{i'})}$$

This function generalizes the sigmoid function to multiple classes/dimensions. We call it the ”softmax” because we may think of it as a smooth, differentiable version of the function which simply returns 1 for the most likely class (or argmax).

5.24 Discriminative Classification

We may apply the mathematical tools developed in the generative classification setting discussed above to perform discriminative classification. In discriminative classification, we assume that our inputs x are fixed, rather than coming from some probability distribution.

We take the maximum likelihood estimate, as in linear regression, given that $p(y = c|x) \propto \exp(\theta_c^T x)$:

$$\text{MLE} : \underset{\theta}{\text{argmax}}\, p(y|x, \theta) = \underset{\theta}{\text{argmax}} \sum_n \log \text{softmax}(\theta_c^T x_n) c_n$$

What are the advantages and disadvantages of this approach? The primary disadvantage compared to methods we have seen earlier is that this maximum likelihood estimate has *no closed form*. It is also not clear how we might incorporate our prior (although there is recent work in this area). On the other hand, this equation is convex and it is easy (at least mathematically, not necessarily computationally) to compute gradients, so we may use gradient descent.

$$\frac{d(\cdot)}{d\theta_c} = \sum_n x_n \cdot \begin{cases} 1 - \text{softmax}(\theta_c^T x) & \text{if } y_n = c \\ \text{softmax}(\theta_c^T x) & \text{otherwise} \end{cases} \tag{5.8}$$

This model is known as **logistic regression** (even though it is used for classification, not regression) and is widely used in practice.

More Resources on Optimization

- Convex Optimization by Lieven Vandenberghe and Stephen P. Boyd

CS281: Advanced ML **September 20, 2017**

Lecture 6: Exponential Families

Lecturer: Sasha Rush *Scribes: Meena Jagadeesan, Yufeng Ling, Tomoka Kan, Wenting Cai*

6.25 Introduction

(Wainwright and Jordan (textbook) presents a more detailed coverage of the material in this lecture.)

This lecture, we will unify all of the fundamentals presented so far:

$p(\theta)$	$p(x)$	$p(y \mid x)/p(x,y)$
Beta, Dir	Discrete	Classification
MVN, IW	MVN	Linear Regression
	Exponential Families	**Generalized Linear Models**
	Undirected Graphic Models	Conditional UGM
	Variational Inference	

We will focus on coming up with a general form for Discrete and MVN through exponential families. We will also come up with a general form for classification and linear regression through generalized linear models.

6.26 Definition of Exponential Family

The definition is

$$\begin{aligned} p(x \mid \theta(\mu)) &= \frac{1}{Z(\theta)} h(x) \exp\{\theta^T \phi(x)\} \\ &= h(x) \exp \theta^T \phi(x) - A(\theta) \end{aligned}$$

where

μ	mean parameters
$\theta(\mu)$	natural / canonical / exponential parameters
$Z(\theta) A(\theta)$	also written as $Z(\theta(\mu))$ or $Z(\mu)$, the partition function and log partition
$\phi(x)$	sufficient statistics of x, potential functions, "features"
$h(x)$	scaling term, in most cases, we have $h(x) = 1$

Note that there is "minimal form" and "overcomplete form".

6.27 Examples of Exponential Families

6.27.1 Bernoulli/Categorical

First, we consider the Bernoulli as an exponential family. Like last lecture, we rewrite the distribution as an exp of log.

$$\begin{aligned} \text{Ber}(x|\mu) &= \mu^x (1-\mu)^{(1-x)} \\ &= \exp x \log \mu + (1-x) \log(1-\mu) \\ &= \underbrace{}_{h(x)} \exp \underbrace{\log\left(\frac{\mu}{1-\mu}\right)}_{\theta} \underbrace{x}_{\phi(x)} + \underbrace{\log(1-\mu)}_{-A(\mu)} \end{aligned}$$

For the **minimal form**, we have

$$\begin{aligned}
h(x) &= 1 \\
\phi_1(x) &= x \\
\theta_1(\mu) &= \log\frac{\mu}{1-\mu}(\text{“log odds”}) \\
\mu &= \sigma(\theta) \\
A(\mu) &= -\log(1-\mu) \\
A(\theta) &= -\log(1-\sigma(\theta)) = \theta + \log(1+e^{-\theta})
\end{aligned}$$

For the **overcomplete form**, we have

$$\begin{aligned}
\phi(x) &= \begin{bmatrix} x \\ 1-x \end{bmatrix} \\
\theta &= \begin{bmatrix} \log\mu \\ \log(1-\mu) \end{bmatrix}
\end{aligned}$$

For the Categorical/Multinouilli distribution, we have

$$\theta = \begin{bmatrix} \log\mu_1 \\ \vdots \\ \log\mu_n \end{bmatrix}$$

where $\sum_c \mu_c = 1$.

Side note: Writing out in overcomplete form usually comes with some restraints.

6.27.2 Univariate Gaussians

$$\begin{aligned}
\mathcal{N}(x \mid \mu, \sigma^2) &= (2\pi\sigma^2)^{1/2}\exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\} \\
&= \underbrace{(2\pi\sigma^2)^{-\frac{1}{2}}}_{A(\mu,\sigma^2)}\exp\{\underbrace{-\frac{1}{2\sigma^2}x^2+\frac{\mu}{\sigma^2}x}_{\theta^T\phi(x)}-\underbrace{\frac{1}{2\sigma^2}\mu^2\}}_{A(\mu,\theta^2)}
\end{aligned}$$

$$\begin{aligned}
\phi(x) &= \begin{bmatrix} x \\ x^2 \end{bmatrix} \\
\theta &= \begin{bmatrix} \frac{\mu}{\sigma^2} \\ -\frac{1}{2\sigma^2} \end{bmatrix} \\
A(\mu,\sigma^2) &= \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\mu^2 \\
\mu &= -\frac{\theta_1}{2\theta_2} \\
\sigma^2 &= -\frac{1}{2\theta_2} \\
A(\theta) &= -\frac{1}{2}\log(-2\theta_2) - \frac{\theta_1^2}{4\theta_2}
\end{aligned}$$

6.27.3 Bad distributions

Two simple distributions that do not fit this form are the uniform distribution Uniform$(0,1)$ (check this as an exercise), and the Student-T distribution.

6.28 Properties of Exponential Families

Most inference problems involve a mapping between natural parameters and mean parameters, so this is a natural framework.

Here are three properties of exponential families:

Property 1 Derivatives of $A(\theta)$ provide us the cumulants of the distribution $\mathbb{E}(\phi(x))$, $\mathrm{var}(\phi(x))$:

Proof. For univariate, first order:

$$\begin{aligned}
\frac{dA}{d\theta} &= \frac{d}{d\theta}(\log Z(\theta)) \\
&= \frac{d}{d\theta}\log \underbrace{\left(\int \exp\{\theta\phi\}h(x)dx\right)}_{\text{needed to integrate to 1}} \\
&= \frac{\int \phi \exp\{\theta\phi\}h(x)dx}{\int \exp(\theta\phi)h(z)dx} \\
&= \frac{\int \phi \exp\{\theta\phi\}h(x)dx}{\exp(A(\theta))} \\
&= \int \phi(x) \underbrace{\exp(\theta\phi(x) - A(\theta))h(x)}_{p(x)} dx \\
&= \int \phi(x)p(x)dx \\
&= \mathbb{E}(\phi(x))
\end{aligned}$$

The same property holds for multivariates (refer to textbook for proof). □

Bernoulli:

$$A(\theta) = \theta + \log(1 + e^{-\theta})$$

$$\frac{dA}{d\theta} = 1 - \frac{e^{-\theta}}{1 + e^{-\theta}} = \underbrace{\frac{1}{1 + e^{-\theta}}}_{\text{sigmoid}} = \sigma(\theta) = \mu$$

Univariate Normal Left as exercise.

Property 2 MLE has a nice form (through "moment matching")

Proof.

$$\begin{aligned}
\underset{\theta}{\operatorname{argmax}} \log p(\text{data} \mid \theta) &= \underset{\theta}{\operatorname{argmax}} \left(\sum_d \theta^T \phi(x_d)\right) - NA(\theta) \\
&= \underset{\theta}{\operatorname{argmax}} \theta^T \underbrace{\left(\sum_d \phi(x_d)\right)}_{\text{sum of sufficient statistics}} - \underbrace{NA(\theta)}_{\text{amount of points}}
\end{aligned}$$

We take a derivative to obtain:

$$\begin{aligned}
\frac{d(.)}{d\theta} &= \sum_d \phi(x_d) - N\frac{dA(\theta)}{d\theta} \\
&= \sum_d \phi(x_d) - N\mathbb{E}(\phi(x)) \\
&= 0
\end{aligned}$$

$$E(\phi(x)) = \underbrace{\frac{\sum \phi(x_d)}{N}}_{\text{set mean parameter to sample means that gives us MLE}}$$

□

Property 3 Exponential families have conjugate priors.

Proof. We first introduce some notations.

$$\begin{aligned}
&\eta \text{ - parameters} \\
&\bar{s} = \sum_d \phi(x_d)/N \\
p(\text{data} \mid \eta) &\propto \exp[(N\bar{s})\eta - NA(\eta)] \\
p(\eta \mid N_0, s_0) &\propto \exp[(N_0, \bar{s}_0)\eta - N_0 \underbrace{A(\eta)}_{\text{not log partition, which has to be a function strictly of parameters}}] \\
p(\eta|\text{data}) &\propto \exp((N\bar{s} + N_0\bar{s}_0)^T\eta - (N_0 + N)A(\eta))
\end{aligned}$$

The above two distributions have the same sufficient statistics – so we have a conjugate prior. It also tells us that it is not a coincidence that we kept obtaining pseudo counts. (More references will be put up to describe this).

□

6.29 Definition of Generalized Linear Models

While exponential families generalize $p(x)$, GLMs generalize $p(y|x)$.

$$p(y|x,w) = h(y)\exp\{\theta(\underbrace{\mu(x)}_{\text{predict mean}})^T\phi(y) - A(\theta)\}$$

where $\mu(x) = \underbrace{g^{-1}}_{\text{squashing const}}(w^Tx + b)$ where g is an appropriate linear transformation.

This can be summarized through the following sequence of transformations:

$$x \overset{g^{-1}(w^Tx+b)}{\longrightarrow} \mu \to \theta \to p(y \mid x).$$

6.30 Examples of Generalized Linear Models

We present three examples:

Example 1 Exponential family - Normal distribution with $\sigma^2 = 1$ and g^{-1} is the identity function. This gives us the linear regression

$$\mu = w^Tx + b \qquad \mathbb{R} \to \mathbb{R}.$$

Example 2 Exponential family - Bernoulli distribution and g^{-1} is the sigmoid function $\sigma : \mathbb{R} \to (0,1)$. Now, $\mu = \sigma(w^Tx + b)$ and $\theta = \log\left(\frac{\mu}{1-\mu}\right)$. This is how we define logistic regression. This gives us

$$p(y \mid x) = \sigma(w^Tx + b)^y(1 - \sigma(w^Tx + b))^{1-y}$$

Example 3 Exponential family - Categorical distribution with g^{-1} as the softmax function.

$\mu_c = \text{softmax}(w_c^Tx + b_c)_c$

$\theta_c = \log \mu_c$

CS281: Advanced ML **September 25, 2017**

Lecture 7: Neural Networks

Lecturer: Sasha Rush *Scribes: Juntao Wang, Alexander Wei, Kevin Zhang, Aron Szanto*

7.30.1 Introduction

Neural networks have been a hot topic recently in machine learning. But everything we will cover today has essentially been known since '70s and '80s. Since then, there has been in increased focus on this subject due to its successes after improved computing power, larger datasets, better neural network architectures, and more careful study in academia. Neural networks have also seen wide adoption in industry in recent years. Lately, there has also been work trying to integrate other methods of inference into neural networks—we will take a look at this topic later in this course. We cover neural networks now as a tangentially-related introduction to graphical models and as an example of combining traditional inference with deep models.

7.30.2 Review of General Linear Models

In the last lecture, we saw that we can learn a mapping between mean parameters and natural parameters for many general linear models and use squashing functions to change natural parameters into mean parameters. In general, we describe these models using a transformation $\mu = g^{-1}(w^\top x + b)$ for some function g and a distribution $y \mid x$.

Example 1 (Linear Regression)**.** Here we have $g(x) = x$ (the identity function) with $y \mid x \sim \mathcal{N}(w^\top x + b, \sigma^2)$. This is the classic model we have already seen.

Example 2 (Linear Classification)**.** In this case, g^{-1} is the sigmoid function σ, so that $\mu = \sigma(w^\top x + b)$ with $y \mid x \sim \text{Bern}(\sigma(w^\top x + b))$. We can think of the sigmoid function as a smooth approximation to an indicator variable, so that $\sigma(w^\top x + b)$ is simply an estimation of the class of $w^\top x + b$.

Example 3 (Softmax Classification)**.** Here g^{-1} is the softmax function, so that $\mu_c = \text{softmax}(Wx + b)_c$ where W is some matrix rather than a vector. Remember that the softmax is defined by $\text{softmax}(z)_c = \frac{\exp(z_c)}{\sum_{c'} \exp(z_{c'})}$. Think of the softmax function as a sigmoid approximation to multi-dimension.

7.31 Basis Functions

It can be advantageous to apply models after modifying the data set using a transformation called a *basis*. We can have a huge variety of basis functions (some of which we have seen on the problem set), e.g., $\phi_j(x) = \sin(x), \tanh(x), \text{ReLU}(x)$, and so on. Vector examples (i.e., functions on vectors) include $\phi_j(x) = \max\{x_1, x_2\}$ and $\phi_j(x) = x_1 x_2 + x_1^2$. Figuring out a good basis within which to represent data is an important problem in machine learning.

Example 4. (Basis Function in Speech Recognition) A snippet of speech might be a waveform, and one way to extract features is to chunk the waveform by time, for each chunk applying a Fourier transform. Then we would take as features some values of each transformed chunk in the frequency domain. Typically this process gives 13 features per chunk. These features are then passed to a learning model.

We now consider general linear models in combination with basis functions. Suppose $y|x \sim \mathcal{N}(w^\top \phi(x) + b, \sigma^2)$, and $y|x \sim \text{Bern}(\sigma(w^\top \phi(x)) + b)$, where ϕ gives rise to a basis. We can do MLE just as before, i.e., compute $\text{argmax}_w \sum_n \log p(y_n | x_n, w)$. In general, these will be solvable just as before, e.g., with numerical optimization—iterative gradient calculation and updates. The form of the MLE depends on the distribution of $y \mid x$. When it is normal, the optimization becomes over sum of squares, when it is Bernoulli, the optimization becomes over cross-entropy (as discussed in previous classes).

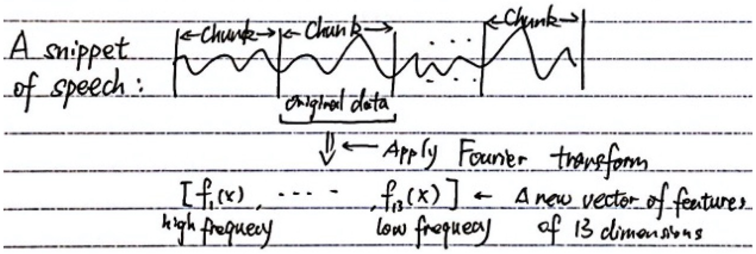

A snippet of speech:
←Chunk→
←Chunk→
←Chunk→
...
original data
← Apply Fourier transform
[$f_1(x)$, ... , $f_{13}(x)$] ← A new vector of features of 13 dimensions
high frequecy
low frequecy

Figure 7.4: Fourier transform in speech recognition in Example 4

7.32 Now Let's Talk About Neural Networks

An adaptive basis function is a model with parameters in the basis functions. Neural networks are specific adaptive basis functions with particular structures, which we will describe below.

One can think of the function of a neural net as learning the correct basis function(s) for the data—having the computer come up with the best representation of the data over features of the form $\phi(x; w, b) = g^{-1}(Wx + b)$, where $g^{-1} =$ relu, σ, softmax, or another nonlinear function. This procedure can be applied recursively, e.g., we can define the x that lives inside this basis function to also have its own basis function, e.g., $\phi(x; w, b) = g^{-1}(w\phi'(x; w', b') + b)$, and so on. This is also why neural networks are often referred to as "deep learning." This allows us to do non-linear regression and classification with parameters. Now, when we do regression and classification, we can have complex models such as

$$y|x \sim \mathcal{N}(w^\top \tanh(w'x + b') + b, \sigma^2)$$

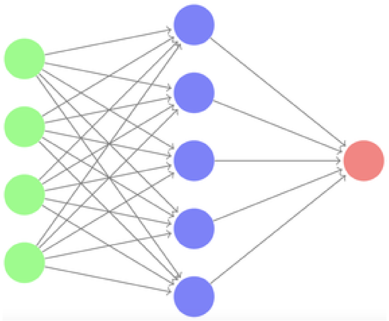
When we do MLE, we have to take the same argmax over the parameters w, w', b', b. All that's changing is that the function we are optimizing is non-linear, with many parameters, and non-convex. So when we optimize such functions, we might end up at a local optimum instead of the global optimum. We will see many techniques for combating the complexities of non-convex optimization.

7.33 Demo

See iPython notebook for demo.

7.34 Graphical Representation

Consider the adaptive basis $\sigma(w^\top \sigma(Wx + b') + b)$. We can represent this graphically with a two-layer, fully-connected network:



In the literature, the circles are called "neurons," matrices are "fully connected," each column is a "layer" with implied squashing, each line is a parameter. The goal of these networks is to find μ.

"Personally, I find this part—the 'it's like a brain!'—pretty silly. It's just linear algebra separated by nonlinear transformations." - S. Rush

7.34.1 Application Architectures for Neural Networks

In a typical neural network, we have $x \to$ Layer 1 $\to$ Layer 2 $\to \cdots \to$ Output, where each arrow is a linear map, and in each layer is a non-linear function. In the class before, we talked about classifying the MNIST data set—for this simple model, we had 8000 parameters(!). "But that's nothing—just yesterday I was working with a model with 1.2 billion parameters."

Although some of the power of neural networks comes from this flexibility in parameters, much of the interesting work is done in trying to find better neural network architectures that capture more of the essence of the data with fewer parameters. For example, the modern approaches to digit classification are done by convolutional neural networks, where the architecture captures some of the "local" information of images.

Example 5. [Speech Recognition] Suppose we want to map sounds into classes of saying the digits "one," "two," and so on. Recall that the typical approach is to split speech into chunks and perform Fourier transforms to extract features from each chunk. The problem here is that individual chunks don't necessarily map to single digits, since there's no guarantee the chunk even corresponds to an entire word in speech!

Instead, what is typically done in this case is convolution using a *kernel* (equivalently, a single weight vector called w^{tile}) that spans several chunks. Rather than applying learning on the full $\mathbb{R}^{n \cdot 13}$ data set (where n is the number of chunks), we multiply each k-chunk stretch of speech by the kernel to obtain $\approx n - k$ chunks. Based on our choice of kernel, we can take advantage of sparsity to improve structure in the data set. This is known as a one-dimensional convolution between the kernel, w^{tile}, and the input $\phi(x)$.

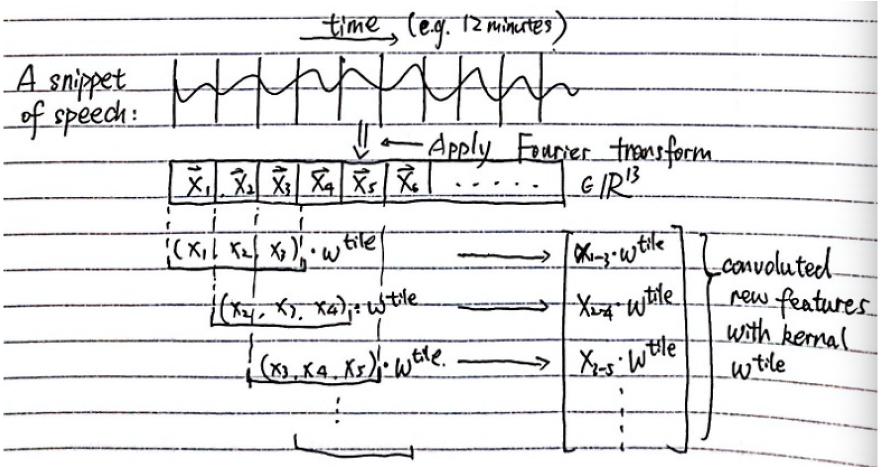

time, (e.g. 12 minutes)
A snippet of speech:
Apply Fourier transform
$\vec{x}_1$ $\vec{x}_2$ $\vec{x}_3$ $\vec{x}_4$ $\vec{x}_5$ $\vec{x}_6$ . . . . .
$\in \mathbb{R}^{13}$
$(x_1, x_2, x_3) \cdot w^{tile}$
$(x_2, x_3, x_4) \cdot w^{tile}$
$(x_3, x_4, x_5) \cdot w^{tile}$
$x_{1-3} \cdot w^{tile}$
$x_{2-4} \cdot w^{tile}$
$x_{3-5} \cdot w^{tile}$
convoluted new features with kernal w^{tile}

Figure 7.5: Convolution architecture in speech recognition in Example 5

Example 6 (Image Classification)**.** For the case of images, we can do the same as above with two-dimensional convolution, where have blocks in the image instead of tiles. This lets us pick up on information that is very local—e.g., or edges or corners in images, informatoin which can then be recombined in later layers with spectacular success.

Example 7 (Language Classification)**.** Suppose we want to determine whether a movie review was good or bad. Consider the review "The movie was not very good." One way to do this to convert words to vector representations (e.g., via word2vec or glove), since discrete words are difficult to deal with, but vectors let us have a more continuous approach while taking into account the meaning. We can do things like add these vectors up over the course of the review (e.g., a bag-of-words approach). An alternative is to take blocks of words and use a one-dimensional convolution. One advantage of the latter is that it allows you to pick up on structures such as "not very good," which wouldn't be observed in a bag-of-words model, which may pick up on the words "very" and "good" instead.

Remark. All of these convolution methods exist in PyTorch under `nn.conv`.

CS281: Advanced ML **September 27, 2017**

Lecture 8: Backpropagation & Directed Graphical Models

Lecturer: Sasha Rush *Scribes: Giridhar Anand, Michael Xueyuan Han, Ana-Roxana Pop*

8.35 Backpropagation in Neural Networks

8.35.1 Neural networks review

In the last lecture, we defined the mean parameter of a neural network as follows:

$$\mu = \sigma(w^T \text{ReLU}(Wx))$$

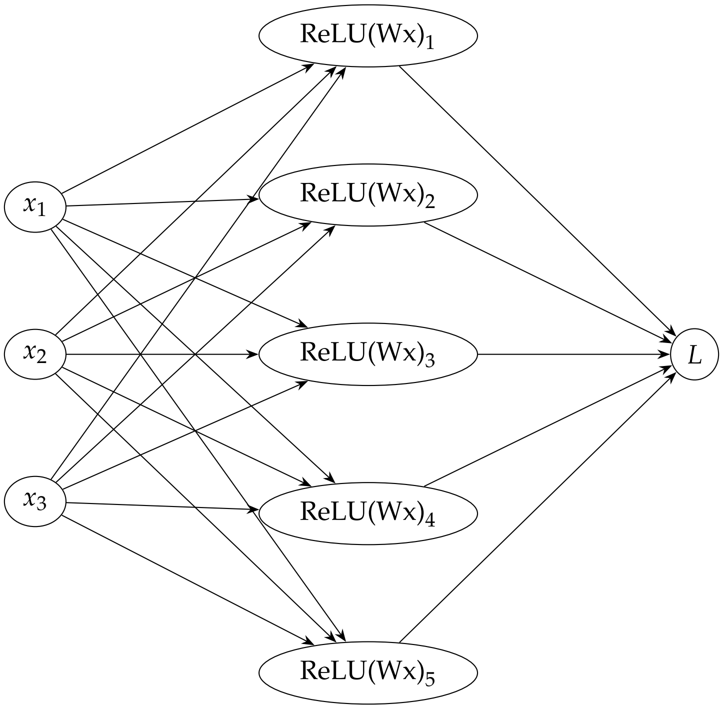
Here, μ parameterizes a Bernoulli distribution, $\text{Ber}(\mu)$. Suppose we want to find μ such that it maximizes the likelihood of a single data example (x, y). Then we compute

$$\mu = \underset{\mu}{\text{argmax}} \log p(y|x) = \arg\min_{\mu} (-\log p(y \mid x)) = \arg\min_{\mu} L$$

where L is the loss of the neural network.

8.35.2 Chain rule and backpropagation

We can represent the neural network graphically as follows:



In order to generate this graph, we must perform the following computational operations in order:

$$\begin{array}{ccccccccccc} v^{(0)} & \rightarrow & v^{(1)} & \rightarrow & v^{(2)} & \rightarrow & v^{(3)} & \rightarrow & v^{(4)} & \rightarrow & L \\ x & & Wv^{(0)} & & \text{ReLU}(v^{(1)}) & & w^T v^{(2)} & & \sigma(v^{(3)}) & & -\log v^{(4)} \end{array}$$

We would like to get the gradient terms $\dot{v}^{(i)} \equiv \frac{dL}{dv^{(i)}}$ for any i, which tell us how each part of the neural network affects our loss. We can do this by applying the chain rule (of calculus) to get a recursive solution (by convention, the derivative of a scalar with respect to a vector is represented as a column vector):

$$\frac{dL}{dv^{(i)}} = \left(\frac{dL}{dv^{(i+1)}}\right)^T \frac{dv^{(i+1)}}{dv^{(i)}}$$

$$\frac{\partial L}{\partial v_k^{(i)}} = \sum_j \frac{\partial L}{\partial v_j^{(i+1)}} \frac{\partial v_j^{(i+1)}}{\partial v(i)}$$

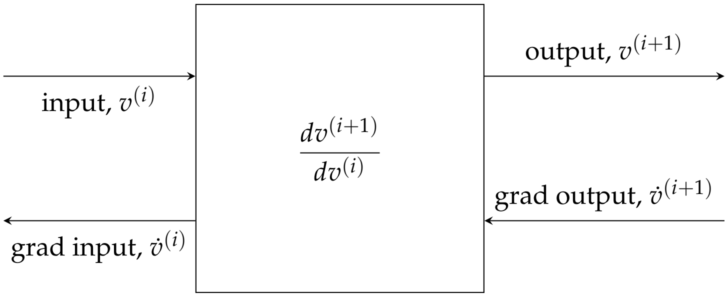
Since the gradient of each term depends on the gradient of the subsequent term, we can compute the gradients in reverse while applying the chain rule. This method is known as backpropagation. For each backward step, we need to remember everything that was computed in the corresponding forward step, namely $v^{(i)}$, $\dot{v}^{(i+1)}$, and $\frac{dv^{(i+1)}}{dv^{(i)}}$:

$$\begin{array}{ccccccccccc}
v^{(0)} & \rightarrow & v^{(1)} & \rightarrow & v^{(2)} & \rightarrow & v^{(3)} & \rightarrow & v^{(4)} & \rightarrow & L \\
x & & Wv^{(0)} & & \text{ReLU}(v^{(1)}) & & w^T v^{(2)} & & \sigma(v^{(3)}) & & -\log v^{(4)} \\
\\
\dot{v}^{(0)} & \leftarrow & \dot{v}^{(1)} & \leftarrow & \dot{v}^{(2)} & \leftarrow & \dot{v}^{(3)} & \leftarrow & \dot{v}^{(4)} & \leftarrow & \\
... & & ... & & (\dot{v}^{(2)})^T W & & \dot{\sigma}(v^{(3)})\dot{v}^{(4)} & & -\frac{1}{v^{(4)}} & &
\end{array}$$

8.35.3 Writing software for neural networks

- "blocks" style neural network (e.g. Torch)

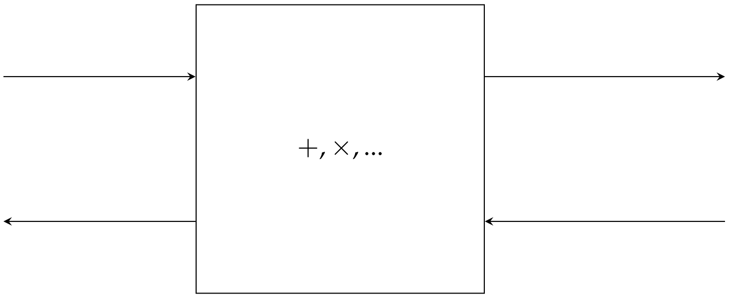
  Computation is done in "blocks" which are black boxes f that implement the following contract:

  
  input, $v^{(i)}$
output, $v^{(i+1)}$
$\frac{dv^{(i+1)}}{dv^{(i)}}$
grad output, $\dot{v}^{(i+1)}$
grad input, $\dot{v}^{(i)}$

  We can also augment the black boxes. For instance, if we let f take in parameter W, we can also compute $\frac{dL}{dW}$ within this function.

- computational graph (e.g. Theano, TensorFlow)

  Everything is implemented in terms of primitives, so there are no black boxes:

  This allows us to optimize the neural network once and run it on many examples.

- imperative/autograd systems

  These are tape-based systems in which the computational graph can look different for different examples, but we can still compute gradients using backpropagation. Torch is built on an autograd core, but higher level functions like the Linear module take on a "blocks" style approach.

8.36 Graphical Models

8.36.1 Directed Graphical Models

The goal of using directed graphical models (DGMs) is to separate out two parts of a model:

1. Conditional Independence
2. Parameters and Parametrization

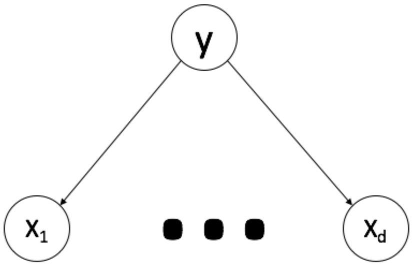


Figure 8.6: The graphical model of Naive Bayes

In the case of Naive Bayes (see Figure 8.6), we know from a previous lecture that:

- $p(y)$ is probably categorical.
- $p(x_j|y)$ could be one of many different distributions, including Categorical, Gaussian, Bernoulli, etc.

We are interested in the following distributions from the underlying data that we have:

- p(y, x): joint distribution
- $p(x_j)$: marginal distribution, or $p(y \mid x)$: conditional distribution

The structure of the model will often determine the difficulty of inference. This is the motivation of why we want to draw these graphs.

On a high level, given $p(A, B, C)$, we can always apply the chain rule (in probability):

$$p(A, B, C) = p(A \mid B, C)\, p(B \mid C)\, p(C)$$

However, if we write $p(A, B, C)$ in the way above, we basically assume that all variables depend on each other. In some cases, this is not necessarily true, and we want to find a factorization as below.

8.36.2 Factorization

If we have the case presented in Figure 8.7, we can rewrite $p(A \mid B, C) \to p(A \mid B)$. Having A only depend on one variable (B) is better than having it depend on two variables (B and C).

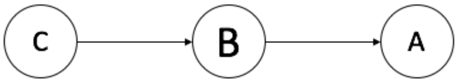


Figure 8.7: A graph where factorization is possible.

8.36.3 Formalism of DGMs

Formally, for directed graphical models (DGMs) (or *Bayes Nets*, or *causal graphs*) we have:

- A graph $G = (V, E)$ where $(s, t) \in E, s \neq t$ (V are vertices, E are edges)
- Each node is a random variable.
- Each edge is a conditioning decision.
- The graph is topologically ordered and it is a directed acyclic graph (DAG).
- Notation: $\text{pa}(x)$ represents x's parents.

8.36.4 Parents notation

Here, A and B are independent of each other, and they are C's parents (as in Figure 8.8). We can then write:

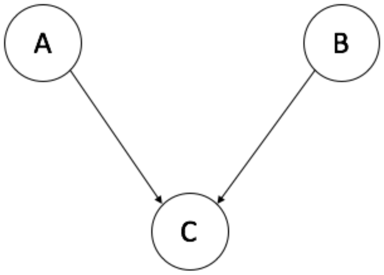


Figure 8.8: A graph to illustrate the use of parents notation.

$$p(A, B, C) = p(A)\, p(B)\, p(C \mid A, B) = p(A)\, p(B)\, p(C \mid \text{pa}(C))$$

8.36.5 Plate Notation

When we have lots of exchangeable variables (i.e., order is not important), we can use the *plate* notation. We want to graphically represent Naive Bayes on examples $(x_j^{(n)}, y^{(n)})$, in which

- y is parameterized on π: $p(y^{(n)} \mid \pi)$, and
- x_j depends on y and μ: $p(x_j^{(n)} \mid y, \mu)$

Since we have n samples of (x, y), we can use the plate notation, as shown in Figure 8.9.

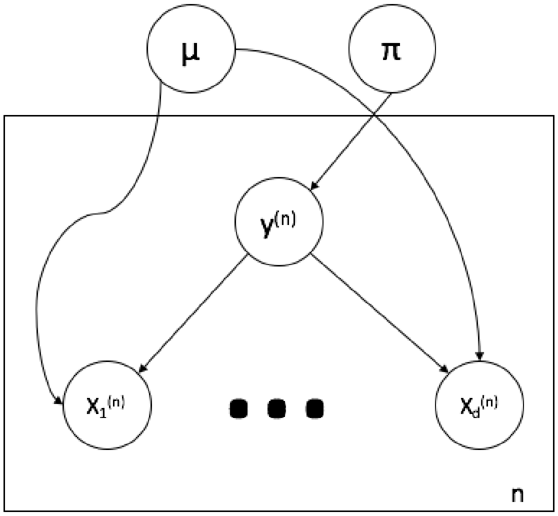


Figure 8.9: A graph to illustrate the use of plate notation.

8.36.6 Caching probabilities

You can save "probabilities" in the model, which is simply *caching* the values of probabilities with the graph. In this case (Figure 8.10), variables are discrete. We call C's probability table *conditional probability table (CPT)* since it is conditioned on A, B. Note that the values do not tell us anything about how the distribution is parameterized. Those are simply the probability values that you can read off from the graph.

Note also that the CPT of $p(x_i \mid x_1, \cdots, x_{i-1}) = O(\prod_i |x_i|)$ (i.e., exponential growth with the number of conditional terms).

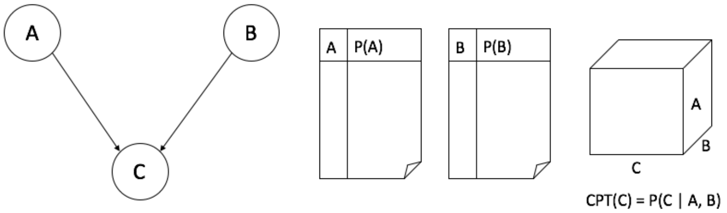


Figure 8.10: A model with probability tables. The CPT of C is a three-dimensional table.

8.36.7 Examples of Directed Graphical Models

Example 8 (Markov Chain)**.** Figure 8.11 shows an example of a Markov chain graphical model.

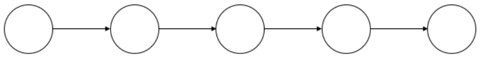


Figure 8.11: Markov Chain Graphical Model

Example 9 (Second Order Markov Chain)**.** Figure 8.12 shows an example of a second order Markov chain graphical model.

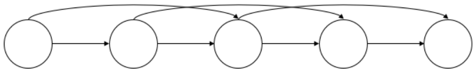


Figure 8.12: Second Order Markov Chain Graphical Model

Example 10 (Hidden Markov Model)**.** Figure 8.13 shows an example of a hidden Markov graphical model.

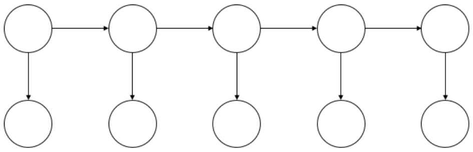


Figure 8.13: Hidden Markov Graphical Model

Example 11 (Navie Bayes)**.** Figure 8.14 shows an example of a Naive Bayes graphical model.

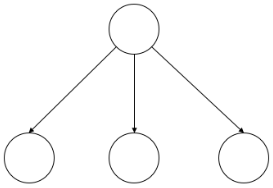


Figure 8.14: Naive Bayes Graphical Model

As we have seen in Figure 8.9, we can also incorporate parameters in the DGMs, as it is illustrated in the next example.

Example 12. In this case (Figure 8.15), we use the same Naive Bayes example with parameters. Here, we incorporate parameters $\alpha \sim$ Dirichlet. This is interesting because it combines two types of distributions: some of them are discrete, but in this example α and π are drawn from continuous distributions, as marked in the figure below.

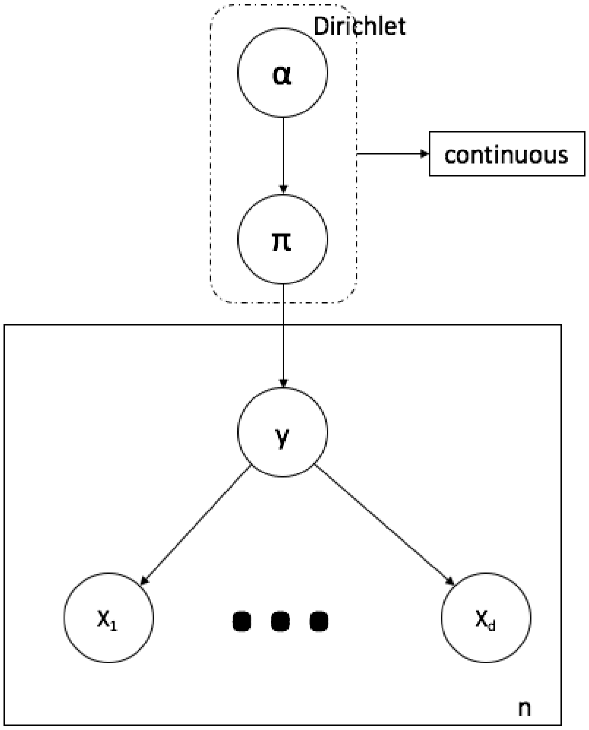


Figure 8.15: Naive Bayes Graphical Model with Parameters

Figure 8.15 corresponds to a single example. If we have multiple examples, we can use a plate-in-plate representation as in Figure 8.16.

8.37 Gaussian Directed Models

Gaussian directed models are a special case of DGMs where every one of the variables has the following distribution:

$$p(x_i \mid \mathrm{pa}(x_i)) = \mathcal{N}(x_i \mid \mu_i + \sum_{j=\mathrm{pa}(x_i)} W_{ij}(x_j - \mu_j),\ \ \sigma_i^2)$$

In the above equation, we transform each of the μ_i based on the starting mean plus a linear transformation of their parents (and to simplify things, we subtract the mean of each parent). We have an underlying generative process where each one of our random variables is a draw from a Gaussian and its children are a linear transformations of that draw.

This means that, we can rewrite x_i as:

$$x_i = \mu_i + \sum_j W_{ij}(x_j - \mu_j) + \sigma_i z_i \qquad \forall i \ \ z_i \sim \mathcal{N}(0,1)$$

Notice that $\sigma_i z_i$ is just Gaussian random noise.

If we define $S = \mathrm{diag}(\sigma)$ (where each term will contribute a different corresponding σ_i), we can rewrite the above equation in a matrix form:

$$(x_i - \mu_i) = W(x - \mu) + Sz$$

where $\mu = [\mu_1, \cdots, \mu_d]$ is a vector containing each individual means.

By rearranging the terms, we find:

$$Sz = (I - W)(x - \mu)$$
$$x - \mu = (I - W)^{-1} Sz$$

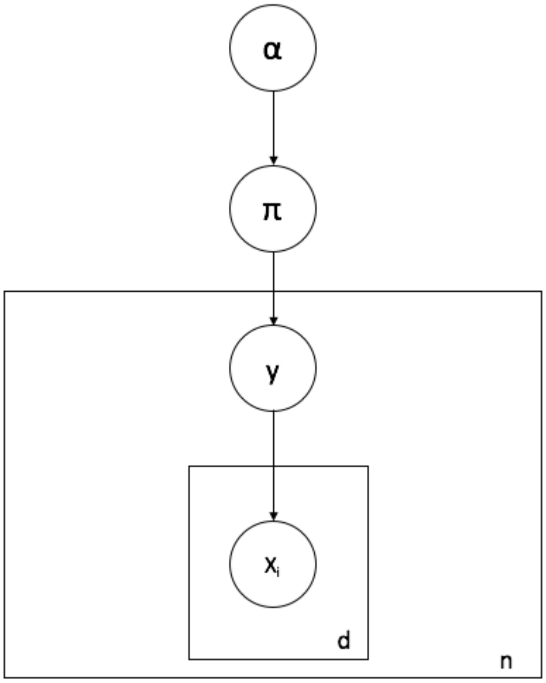


Figure 8.16: Naive Bayes Graphical Model with Parameters and Plate-in-Plate Notation

This tells us how the x random variable differs from the mean at each of the different positions.

We know that the Σ term for our covariant matrix is defined as:

$$\begin{aligned}\Sigma \equiv \operatorname{cov}[x-\mu] &= \operatorname{cov}\left[(I-W)^{-1}S\,z\right]\\ &= (I-W)^{-1}S\operatorname{cov}[z]\;S((I-W)^{-1})^T\\ &= (I-W)^{-1}\,S^2\,((I-W)^{-1})^T\end{aligned}$$

which means that, in general, for Gaussian DGMs we have:

$$\text{Gaussian DGM} \sim \mathcal{N}(\mu,\,(I-W)^{-1}\,S^2\,((I-W)^{-1})^T)$$

Remark. We will talk about *D-Separation* in the next lecture.