Learning two-node GMs

Probabilistic Graphical Models (10-708)

Lecture 8, Oct 10, 2007

Eric Xing

Reading: J-Chap. 5,6, KF-Chap. 8

Two-node BNs

| $X$ | $Y$ | $p(Y|X)$ |
| --- | --- | --- |
| $\mathbb{R}^n$ | $\mathbb{R}^m$ | regression |
| $\mathbb{R}^n$ | $\{0,1\}^n$ | binary classification |
| $\{0,1\}^n$ | $\{0,1\}$ | binary classification |
| $\mathbb{R}^n$ | $\{1,\ldots,K\}$ | multiclass classification |
| $\{1,\ldots,K\}$ | $\mathbb{R}^n$ | conditional density modeling |
Multimodal models

- A bimodal probability density:

\[
\text{GM:}
\]

Conditional Gaussian

- The data:

\[
\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_N, y_N)\}
\]

- Both nodes are observed:
  - \(Y\) is a class indicator vector
    \[
p(y_n) = \text{multi}(y_n : \pi) = \prod_{k} \pi_k^{y_n}
\]
  - \(X\) is a conditional Gaussian variable with a class-specific mean
    \[
p(x_n | y_n = 1, \mu, \sigma) = \frac{1}{(2\pi \sigma)^{1/2}} \exp \left\{ -\frac{1}{2\sigma^2} (x_n - \mu)^2 \right\}
    \]
    \[
p(x | y, \mu, \sigma) = \prod_n \left( \prod_k \mathcal{N}(x_n : \mu_k, \sigma) \right)
\]
MLE of conditional Gaussian

- Data log-likelihood
  \[ \ell(\theta; D) = \log \prod p(x_n, y_n) = \log \prod p(y_n | \pi) p(x_n | y_n, \mu, \sigma) \]
  \[ = \sum \log \prod \pi_i^y + \sum \log \prod \mathcal{N}(x_n, \mu_i, \sigma)^y \]
  \[ = \sum y_i^x \log \pi_i + \sum y_i^x \frac{1}{2\pi\sigma} (x_n - \mu_i)^2 + C \]

- MLE
  \[ \hat{\pi}_{i, MLE} = \arg \max \ell(\theta; D), \quad \Rightarrow \hat{\pi}_i = \frac{\sum y_i^x}{N} = \frac{n_i}{N} \]
  the fraction of samples of class \(m\)
  \[ \hat{\mu}_{i, MLE} = \arg \max \ell(\theta; D), \quad \Rightarrow \hat{\mu}_i = \frac{\sum y_i^x x_n}{\sum y_i^x} = \frac{\sum y_i^x x_n}{n_i} \]
  the average of samples of class \(m\)

Bayesian estimation of conditional Gaussian

- Prior:
  \[ P(\bar{\pi} | \bar{\alpha}) = \text{Dir}(\bar{\pi} : \bar{\alpha}) \]
  \[ P(\mu_k | \nu) = \text{Normal}(\mu_k : \nu, \tau) \]

- Posterior mean (Bayesian est.)
  \[ \pi_{k, \text{Bayes}} = \frac{N}{N + |d|} \hat{\pi}_{k, MLE} + \frac{|d|}{N + |d|} \frac{\alpha_k + n_k}{N + |d|} \]
  \[ \mu_{k, \text{Bayes}} = \frac{n_k}{n_k / \sigma^2 + 1/\tau^2} \hat{\mu}_{k, MLE} + \frac{1/\tau^2}{n_k / \sigma^2 + 1/\tau^2} \nu, \quad \text{and} \quad \sigma^2_{\text{Bayes}} = \left( \frac{N}{\sigma^2 + 1/\tau^2} \right)^{-1} \]
Classification

- From conditional density modeling to classification:
  - The joint probability of a datum and its label is:
    \[ p(x_n, y_n) = p(y_n = 1 | \mu, \sigma) \times p(x_n | y_n = 1, \mu, \sigma) \]
    \[ = \pi_i \frac{1}{(2\pi\sigma^2)^{n/2}} \exp \left\{ -\frac{1}{2\sigma^2} (x_n - \mu)^2 \right\} \]
  - Given a datum \( x_n \), we predict its label using the conditional probability of the label given the datum:
    \[ p(y_n = 1 | x_n, \mu, \Sigma) = \frac{\pi_i \frac{1}{(2\pi\sigma^2)^{n/2}} \exp \left\{ -\frac{1}{2\sigma^2} (x_n - \mu)^2 \right\}}{\sum_{\tau} \pi_i \frac{1}{(2\pi\sigma^2)^{n/2}} \exp \left\{ -\frac{1}{2\sigma^2} (x_n - \mu)^2 \right\}} \]
  - This is basic inference
    - introduce evidence, and then normalize

Naïve Bayes Classifier

- When \( X \) is multivariate-Gaussian vector:
  - The joint probability of a datum and its label is:
    \[ p(\tilde{x}_n, y_n) = p(y_n = 1 | \mu, \Sigma) \times p(\tilde{x}_n | y_n = 1, \mu, \Sigma) \]
    \[ = \pi_i \frac{1}{(2\pi\Sigma)^{n/2}} \exp \left\{ -\frac{1}{2\Sigma} (\tilde{x}_n - \tilde{\mu})^T \Sigma^{-1} (\tilde{x}_n - \tilde{\mu}) \right\} \]
  - The naïve Bayes simplification
    \[ p(x_n, y_n) = p(y_n = 1 | \mu, \Sigma) \times \prod_j p(x_n^j | y_n = 1, \mu_{ij}, \sigma_{ij}) \]
    \[ = \pi_i \prod_j \frac{1}{(2\pi\sigma_{ij})^{1/2}} \exp \left\{ -\frac{1}{2\sigma_{ij}} (x_n^j - \mu_{ij})^2 \right\} \]
  - More generally:
    \[ p(x_n, y_n | \eta, \pi) = p(y_n | \pi) \times \prod_j p(x_n^j | y_n, \eta) \]
    - Where \( p(\cdot | \cdot) \) is an arbitrary conditional (discrete or continuous) 1-D density
Transductive classification

- Given $X_n$, what is its corresponding $Y_n$ when we know the answer for a set of training data?

- Frequentist prediction:
  - we fit $\pi$, $\mu$ and $\sigma$ from data first, and then ...

  $$ p(y^k_n = 1 | x_n, \mu, \sigma, \pi) = \frac{p(y^k_n = 1, x_n | \mu, \sigma, \pi)}{p(x_n | \mu, \sigma, \pi)} = \frac{\pi_k N(x_n | \mu_k, \sigma)}{\sum_j \pi_j N(x_n | \mu_j, \sigma)} $$

- Bayesian:
  - we compute the posterior dist. of the parameters first …
  - Do you want to make it a homework (say, just assume that $\pi$ and $\mu$ are uncertain)?

GM:

The predictive distribution

- Understanding the predictive distribution

  $$ p(y^k_n = 1 | x_n, \mu, \sigma, \pi) = \frac{p(y^k_n = 1, x_n | \mu, \sigma, \pi)}{p(x_n | \mu, \sigma)} = \frac{\pi_k N(x_n | \mu_k, \sigma)}{\sum_j \pi_j N(x_n | \mu_j, \sigma)} $$

- For two class (i.e., $K=2$), * turns out to be the logistic function

  $$ p(y^k_n = 1 | x_n) = \frac{1}{1 + e^{-\theta^T x_n}} $$

- For multiple class (i.e., $K>2$), * correspond to a softmax function

  $$ p(y^k_n = 1 | x_n) = \frac{e^{\theta^T x_n}}{\sum_j e^{\theta^T x_n}} $$
Discussion

- We've seen how to learning two-node model $p(y_n, x_n)$, but in certain problems the goal is to learning $p(y_n | x_n)$.

- Can we model $p(y_n | x_n)$ directly?

- How?

Generative and discriminative classifiers

- Generative:
  - Modeling the joint distribution of all data

- Discriminative:
  - Modeling only points at the boundary

- How? Regression!
Linear regression

- The data:
  \[ \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_N, y_N)\} \]

- Both nodes are observed:
  - \(X\) is an input vector
  - \(Y\) is a response vector
  
  (we first consider \(y\) as a generic continuous response vector, then we consider the special case of classification where \(y\) is a discrete indicator)

- A regression scheme can be used to model \(p(y|x)\) directly, rather than \(p(x,y)\)

### Apartment hunting

- Now you’ve moved to Pittsburgh!!
  And you want to find the most reasonably priced apartment satisfying your needs:
  - square-ft., # of bedroom, distance to campus …

<table>
<thead>
<tr>
<th>Living area (ft²)</th>
<th># bedroom</th>
<th>Rent ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>230</td>
<td>1</td>
<td>600</td>
</tr>
<tr>
<td>506</td>
<td>2</td>
<td>1000</td>
</tr>
<tr>
<td>433</td>
<td>2</td>
<td>1100</td>
</tr>
<tr>
<td>109</td>
<td>1</td>
<td>500</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>270</td>
<td>1.5</td>
<td>?</td>
</tr>
</tbody>
</table>
The learning problem

- **Features:**
  - Living area, distance to campus, # bedroom …
  - Denote as \( x = [x_1, x_2, ..., x_k] \)

- **Target:**
  - Rent
  - Denoted as \( y \)

- **Training set:**

\[
\begin{bmatrix}
\text{rent} \\
\text{Location} \\
\text{Living area}
\end{bmatrix} =
\begin{bmatrix}
\begin{array}{c}
\text{living area of house}_{1} \\
\text{living area of house}_{2} \\
\text{living area of house}_{n}
\end{array}
\end{bmatrix}
\]

\[
\begin{bmatrix}
\text{rent} \\
\text{Location} \\
\text{Living area}
\end{bmatrix} =
\begin{bmatrix}
\begin{array}{c}
\text{rent of house}_{1} \\
\text{rent of house}_{2} \\
\text{rent of house}_{n}
\end{array}
\end{bmatrix}
\]

- Our goal:

Linear Regression

- Assume that \( Y \) (target) is a linear function of \( X \) (features):
  - e.g.: \( \hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \)
  - Let’s assume a vacuous “feature” \( X_0 = 1 \) (this is the intercept term, why?), and define the feature vector to be:

\[
\begin{bmatrix}
1 \\
x_1 \\
x_2 \\
\vdots \\
x_n
\end{bmatrix}
\]

- Then we have the following general representation of the linear function:

\[
\hat{y}_i = \theta_0 + \theta_1 x_{1i} + \theta_2 x_{2i} + \cdots + \theta_k x_{ki}
\]

- Our goal is to pick the optimal \( \theta \). How!
  - We seek \( \theta \) that minimize the following cost function:

\[
J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2
\]
The Least-Mean-Square (LMS) method

- The Cost Function:
  \[ J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (x_i^T \theta - y_i)^2 \]

- Consider a gradient descent algorithm:
  \[ \theta_j^{t+1} = \theta_j^t - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \]

Now we have the following descent rule:

\[ \theta_j^{t+1} = \theta_j^t + \alpha \sum_{i=1}^{n} (y_n - x_n^T \theta^t)x_{n,j} \]

- For a single training point, we have:
  
  - This is known as the LMS update rule, or the Widrow-Hoff learning rule
  - This is actually a "stochastic", "coordinate" descent algorithm
  - This can be used as an on-line algorithm
The Least-Mean-Square (LMS) method

- Steepest descent
  - Note that:
    \[ \nabla_{\theta} J = \left[ \frac{\partial}{\partial \theta_1} J, \ldots, \frac{\partial}{\partial \theta_k} J \right]^T = -\sum_{i=1}^{n} (y_n - \mathbf{x}_n^T \mathbf{\theta}) \mathbf{x}_n \n\]
    \[ \theta^{t+1} = \theta^t + \alpha \sum_{i=1}^{n} (y_n - \mathbf{x}_n^T \theta^t) \mathbf{x}_n \n\]
  - This is as a batch gradient descent algorithm

Some matrix derivatives

- For \( f : \mathbb{R}^{m \times n} \rightarrow \mathbb{R} \), define:
  \[ \nabla_{A} f(A) = \left[ \begin{array}{ccc} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{array} \right] \n\]
- Trace:
  \[ \text{tr}A = \sum_{i=1}^{n} A_{ii} , \quad \text{tr}a = a , \quad \text{tr}ABC = \text{tr}CAB = \text{tr}BCA \n\]
- Some fact of matrix derivatives (without proof)
  \[ \nabla_{A} \text{tr}AB = B^T , \quad \nabla_{A} \text{tr}ABA^T C = CAB + C^T AB^T , \quad \nabla_{A} |A| = |A(A^{-1})^T | \]
The normal equations

- Write the cost function in matrix form:
  \[ J(\theta) = \frac{1}{2} n \sum_{i=1}^{n} (x_i^T \theta - y_i)^2 \]
  \[ = \frac{1}{2} \sum_{i} (X\theta - \tilde{y})^T (X\theta - \tilde{y}) \]
  \[ = \frac{1}{2} (\theta^T X^T X \theta - \theta^T X^T \tilde{y} - \tilde{y}^T X \theta + \tilde{y}^T \tilde{y}) \]

- To minimize \( J(\theta) \), take derivative and set to zero:
  \[ \nabla_{\theta} J = \frac{1}{2} \nabla_{\theta} \text{tr}(\theta^T X^T X \theta - \theta^T X^T \tilde{y} - \tilde{y}^T X \theta + \tilde{y}^T \tilde{y}) \]
  \[ = \frac{1}{2} \left( \nabla_{\theta} \text{tr}(\theta^T X^T X \theta) - 2 \nabla_{\theta} \text{tr}(\theta^T X \tilde{y}) \theta + \nabla_{\theta} \text{tr}(\tilde{y}^T \tilde{y}) \right) \]
  \[ = \frac{1}{2} \left( X^T X \theta + X^T \tilde{y} - 2 \tilde{y}^T \tilde{y} \right) \]
  \[ = X^T \theta - X^T \tilde{y} = 0 \]

\[ \Rightarrow X^T X \theta = X^T \tilde{y} \]

A recap:

- LMS update rule
  \[ \theta_{j+1} = \theta_j + \alpha (y_n - x_n^T \theta) x_n \]
  - Pros: on-line, low per-step cost
  - Cons: coordinate, maybe slow-converging

- Steepest descent
  \[ \theta_{j+1} = \theta_j + \alpha \sum_{i=1}^{n} (y_n - x_n^T \theta) x_n \]
  - Pros: fast-converging, easy to implement
  - Cons: a batch,

- Normal equations
  \[ \theta^* = \left( X^T X \right)^{-1} X^T \tilde{y} \]
  - Pros: a single-shot algorithm! Easiest to implement.
  - Cons: need to compute pseudo-inverse \((X^T X)^{-1}\), expensive, numerical issues (e.g., matrix is singular ..)
Geometric Interpretation of LMS

- The predictions on the training data are:
  \[ \hat{y} = X\theta^* = X\left(X^TX\right)^{-1}X^T\tilde{y} \]

- Note that
  \[ \hat{y} - \tilde{y} = (X^TX)^{-1}X^T(\tilde{y} - \tilde{y}) \]
  and
  \[ X^T(\hat{y} - \tilde{y}) = X^T(X^TX)^{-1}X^T(\tilde{y} - \tilde{y}) = (X^TX)^{-1}X^T(\tilde{y} - \tilde{y}) = 0 \]
  \( \hat{y} \) is the orthogonal projection of \( \tilde{y} \) into the space spanned by the column of \( X \)

Probabilistic Interpretation of LMS

- Let us assume that the target variable and the inputs are related by the equation:
  \[ y_i = \theta^T x_i + \epsilon_i \]
  where \( \epsilon \) is an error term of unmodeled effects or random noise

- Now assume that \( \epsilon \) follows a Gaussian \( N(0, \sigma) \), then we have:
  \[ p(y_i | x_i; \theta) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(y_i - \theta^T x_i)^2}{2\sigma^2}\right) \]

- By independence assumption:
  \[ L(\theta) = \prod_{i=1}^{n} p(y_i | x_i; \theta) = \left(\frac{1}{\sqrt{2\pi\sigma}}\right)^n \exp\left(-\frac{\sum_{i=1}^{n} (y_i - \theta^T x_i)^2}{2\sigma^2}\right) \]
Probabilistic Interpretation of LMS, cont.

- Hence the log-likelihood is:

$$l(\theta) = n \log \frac{1}{\sqrt{2\pi\sigma}} - \frac{1}{\sigma^2} \frac{1}{2} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2$$

- Do you recognize the last term?

Yes it is: $$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (x_i^T \theta - y_i)^2$$

- Thus under independence assumption, LMS is equivalent to MLE of $\theta$!

Multivariate Linear Regression

- Consider vector-valued input $X \in \mathbb{R}^k$ leading to vector-valued output $Y \in \mathbb{R}^d$ via regression matrix $A \in \mathbb{R}^{k \times d}$:

$$p(y \mid x) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (y - Ax)^T \Sigma^{-1} (y - Ax) \right\}$$

- Log-(conditional-) likelihood

$$\ell = -\frac{1}{2} \sum_n |\Sigma| - \frac{1}{2} \sum_n (y_n - Ax_n)^T \Sigma^{-1} (y_n - Ax_n) + c$$

- To take derivatives wrt a matrix, we use the following identity

$$\frac{\partial \left( (Ma + b)^T C(Ma + b) \right)}{\partial M} = (C + C^T)(Ma + b)a^T$$

where $M = A, a = -x_n, b = y_n$ and $C = \Sigma^{-1}$
Multivariate Linear Regression

- Log-(conditional-) likelihood
  \[ \ell = -\frac{1}{2} \sum_n [y_n - \sum_n (y_n - A x_n) \Sigma^{-1} (y_n - A x_n)] + c \]

- Using
  \[ \frac{\partial ((Ma + b)^T C(Ma + b))}{\partial M} = (C + C^T)(Ma + b)a^T \]

we have
  \[ \frac{\partial \ell}{\partial A} = \frac{1}{2} \sum_n 2 \Sigma^{-1} (y_n - A x_n) x_n^\top \]

  \[ = \Sigma^{-1} \left( \sum_n y_n x_n^\top - A \sum_n x_n x_n^\top \right) \]

  \[ = \Sigma^{-1} (S_{YX} - A S_{XX}) = 0 \]

where \( S_{XX} \) and \( S_{YY} \) are the sufficient statistics.

Hence
  \[ A = S_{XX}^{-1} S_{XY} \]

1-D linear regression

\[ A = S_{XX}^{-1} S_{XY} \]

- In the special case of scalar outputs, let \( A = \theta^\top \), and the design matrix \( X = [x_1, \ldots, x_N] \) as a row vector and \( Y = [y_1, \ldots, y_N] \) as a column vector. Then we get the normal equations

  \[ \theta = (X^T X)^{-1} X^T Y \]
Bayesian linear regression

The joint likelihood:

\[ p(y_i, \theta | x_i) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(y_i - \theta^T x_i)^2}{2\sigma^2}\right) \times \frac{\lambda}{2} \exp\left(-\lambda|\theta|_1\right) \]

The "regularized" regression cost function

\[ J(\theta) = (y_i - \theta^T x_i)^2 + \lambda|\theta|_1 \]

Laplace Prior and Sparsity

- The Laplace prior:
  \[ p(\theta_k | \lambda) = \frac{\lambda}{2} \exp(-\lambda|\theta_k|) \]
  \[ p(\theta | \lambda) = \frac{\lambda}{2} \exp(-\lambda|\theta|_1) \]

- The joint likelihood:

- The "regularized" regression cost function
### L1 regularization

- The "regularized" cost:

\[ J(\theta) = (y_i - \theta^T x_i)^2 + \lambda |\theta|_1 \]

- The regularization term penalizes all factors equally. This makes the \( \theta \) *SPARSE*
- A sparse \( \theta \) means reduced complexity
- Can be viewed as a selection of relevant/important features
- \( J(\theta) \) is Non-differentiable
  - Can transform into convex quadratic problem, and use standard convex optimization methods to solve, but these usually cannot handle large practical problems
  - \( J(\theta) \) is piece-wise differentiable, \( \rightarrow \) piece-wise gradient

\[
p(\theta | \lambda) = \left\{ \begin{array}{ll}
\frac{1}{2} \exp\left(-\lambda(\theta_i + |\theta_i|)\right) & \theta_i \geq 0 \\
\frac{1}{2} \exp\left(-\lambda(-\theta_i + |\theta_i|)\right) & \theta_i < 0
\end{array} \right.
\]

- Known as Lasso regression in Statistics

### Effects of L1-Regularization

Select \( \lambda \) by cross-validation
L2 regularization

- Let
  \[ p(\theta | \lambda) = \left(\frac{\lambda}{\pi}\right)^{N/2} \exp\left(-\lambda(\theta - \theta_0)^T(\theta - \theta_0)^T\right) \]

- The joint likelihood:
  \[ p(y, \theta | x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(y - \theta^T x)^2}{2\sigma^2}\right) \times \left(\frac{\lambda}{\pi}\right)^{N/2} \exp\left(-\lambda|\theta|^2\right) \]

- The "regularized" regression cost function
  \[ J(\theta) = (y - \theta^T x)^2 + \lambda|\theta|^2 \]
  - Regularization term restricts large value components
  - Smooth and convex,
  - Can be computed directly (O(n^3))
  - Or can use iterative methods (e.g. conjugate gradients method)

Recall the condition-Gaussian classifier

- So we have seen a new scheme based on LMS (ML) to learn two node GM: \( p(y | x; \theta) = \mathcal{N}(y; \theta^T x, \sigma^2) \) discriminatively
  - Gradient descent
  - Normal equation

- How can we use this scheme to learning the conditional Gaussian classifier discriminatively?
  - Recall that
    \[ p(y | x) = \mu(x)^y(1 - \mu(x))^{1-y} \]
    where \( \mu(x) = \frac{1}{1 + e^{-\theta^T x}} \)
Logistic regression (sigmoid classifier)

- The condition distribution: a Bernoulli
  \[ p(y | x) = \mu(x)^y (1 - \mu(x))^{1-y} \]
  where \( \mu \) is a logistic function
  \[ \mu(x) = \frac{1}{1 + e^{-\theta^T x}} \]

- We can use the brute-force gradient method as in LR

- But we can also apply generic laws by observing the \( p(y | x) \) is an exponential family function, more specifically, a generalized linear model (see next lecture!)

Summary

- Conditional Density Est.
- Classification
  - Generative classifier
  - Discriminative classifier
- Linear Regression
  - Algorithms
    - LMS
    - Steepest descent
    - Normal equation
  - Regularized regression vs. Bayesian regression
Exponential family

- For a numeric random variable $\mathbf{X}$
  \[ p(\mathbf{x} | \eta) = \frac{1}{Z(\eta)} h(\mathbf{x}) \exp \left\{ \eta^T \mathbf{T}(\mathbf{x}) - A(\eta) \right\} \]
  \[ = \frac{1}{Z(\eta)} h(\mathbf{x}) \exp \left\{ \eta^T \mathbf{T}(\mathbf{x}) \right\} \]
  is an exponential family distribution with natural (canonical) parameter $\eta$.

- Function $\mathcal{T}(\mathbf{x})$ is a sufficient statistic.
- Function $A(\eta) = \log Z(\eta)$ is the log normalizer.
- Examples: Bernoulli, multinomial, Gaussian, Poisson, gamma, ...

Multivariate Gaussian Distribution

- For a continuous vector random variable $\mathbf{X} \in \mathbb{R}^k$:
  \[ p(\mathbf{x} | \mu, \Sigma) = \frac{1}{(2\pi)^{k/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) \right\} \]
  \[ = \frac{1}{(2\pi)^{k/2}} \exp \left\{ -\frac{1}{2} \text{tr} (\Sigma^{-1} \mathbf{x} \mathbf{x}^T) + \mu^T \Sigma^{-1} \mathbf{x} - \frac{1}{2} \mu^T \Sigma^{-1} \mu + \log |\Sigma| \right\} \]

- Exponential family representation
  \[ \eta = [\Sigma^{-1} \mu; -\frac{1}{2} \text{vec}(\Sigma^{-1})] = [\eta_1, \text{vec}(\eta_2)]. \]
  $\eta_1 = \Sigma^{-1} \mu$ and $\eta_2 = -\frac{1}{2} \Sigma^{-2}$
  \[ \mathbf{T}(\mathbf{x}) = [\mathbf{x}; \text{vec}(\mathbf{x} \mathbf{x}^T)] \]
  \[ A(\eta) = \frac{1}{2} \mu^T \Sigma^{-1} \mu + \log |\Sigma| = -\frac{1}{2} \text{tr} (\eta_2 \eta_1^T) - \frac{1}{2} \log (-2\eta_2) \]
  \[ h(x) = (2\pi)^{-k/2} \]

- Note: a $k$-dimensional Gaussian is a $(d+\frac{d(d+1)}{2})$-parameter distribution with a $(d+\frac{d(d+1)}{2})$-element vector of sufficient statistics (but because of symmetry and positivity, parameters are constrained and have lower degree of freedom).
**Multinomial distribution**

- For a binary vector random variable $\mathbf{X} \sim \text{multi}(\mathbf{X} | \pi)$, 
  
  \[ p(\mathbf{x} | \pi) = \pi_1^{x_1} \pi_2^{x_2} \cdots \pi_k^{x_k} = \exp \left\{ \sum_{i=1}^k x_i \ln \pi_i \right\} \]
  
  \[ = \exp \left\{ \sum_{i=1}^k x_i \ln \pi_i + \left( 1 - \sum_{i=1}^k \pi_i \right) \ln \left( 1 - \sum_{i=1}^k \pi_i \right) \right\} \]
  
  \[ = \exp \left\{ \sum_{i=1}^k x_i \ln \left( \frac{\pi_i}{1 - \sum_{i=1}^k \pi_i} \right) + \ln \left( 1 - \sum_{i=1}^k \pi_i \right) \right\} \]
  
- Exponential family representation

  \[ \eta = \begin{bmatrix} \ln \left( \frac{\pi_1}{\pi_k} \right) \\ 0 \end{bmatrix}, \quad T(x) = [x] \]

  \[ A(\eta) = -\ln \left( 1 - \sum_{i=1}^k \pi_i \right) = \ln \left( \sum_{i=1}^k \pi_i \right) \]

  \[ h(x) = 1 \]

---

**Why exponential family?**

- Moment generating property

  \[ \frac{dA}{d\eta} = \frac{d}{d\eta} \log Z(\eta) = \frac{1}{Z(\eta)} \frac{d}{d\eta} Z(\eta) \]

  \[ = \frac{1}{Z(\eta)} \left[ h(x) \exp \left( \eta^T T(x) \right) \right] dx \]

  \[ = \left[ \frac{T(x) h(x) \exp \left( \eta^T T(x) \right)}{Z(\eta)} \right] dx \]

  \[ = \mathbb{E}[T(x)] \]

  \[ \frac{d^2 A}{d\eta^2} = \int \left( \frac{T^2(x)}{Z(\eta)} \right) h(x) \exp \left( \eta^T T(x) \right) dx - \left[ \frac{h(x) \exp \left( \eta^T T(x) \right) \exp \left( \eta^T T(x) \right)}{Z(\eta)} \right] dx \]

  \[ = \int \left( \frac{T^2(x)}{Z(\eta)} \right) h(x) \exp \left( \eta^T T(x) \right) dx - \mathbb{E}\left[ T^2(x) \right] \]

  \[ = \text{Var}[T(x)] \]
Moment estimation

- We can easily compute moments of any exponential family distribution by taking the derivatives of the log normalizer $A(\eta)$.
- The $q^{th}$ derivative gives the $q^{th}$ centered moment.

\[
\frac{dA(\eta)}{d\eta} = \text{mean} \\
\frac{d^2 A(\eta)}{d\eta^2} = \text{variance} \\
\ldots
\]

- When the sufficient statistic is a stacked vector, partial derivatives need to be considered.

Moment vs canonical parameters

- The moment parameter $\mu$ can be derived from the natural (canonical) parameter

\[
\frac{dA(\eta)}{d\eta} = E[T(x)] \overset{\text{def}}{=} \mu
\]

- $A(\eta)$ is convex since

\[
\frac{d^2 A(\eta)}{d\eta^2} = Var[T(x)] > 0
\]

- Hence we can invert the relationship and infer the canonical parameter from the moment parameter (1-to-1):

\[
\eta = \psi(\mu)
\]

- A distribution in the exponential family can be parameterized not only by $\eta$ – the canonical parameterization, but also by $\mu$ – the moment parameterization.
MLE for Exponential Family

- For iid data, the log-likelihood is
  \[\ell(\eta; D) = \log \prod_{x} h(x_{x}) \exp \left( \eta' T(x_{x}) - A(\eta) \right)\]
  \[= \sum_{x} \log h(x_{x}) + \left( \eta' \sum_{x} T(x_{x}) \right) - N A(\eta)\]
- Take derivatives and set to zero:
  \[\frac{\partial \ell}{\partial \eta} = \sum_{x} T(x_{x}) - N \frac{\partial A(\eta)}{\partial \eta} = 0\]
  \[\Rightarrow \frac{\partial A(\eta)}{\partial \eta} = \frac{1}{N} \sum_{x} T(x_{x})\]
  \[\Rightarrow \hat{\eta}_{MLE} = \frac{1}{N} \sum_{x} T(x_{x})\]
- This amounts to **moment matching**.
- We can infer the canonical parameters using \(\hat{\eta}_{MLE} = \psi(\hat{\mu}_{MLE})\)

Sufficiency

- For \(p(x | \theta)\), \(T(x)\) is **sufficient** for \(\theta\) if there is no information in \(X\) regarding \(\theta\) beyond that in \(T(x)\).
  - We can throw away \(X\) for the purpose of inference w.r.t. \(\theta\).
  - Bayesian view
    \[\begin{array}{c}
    X \\
    \xrightarrow{\text{}} T(x) \\
    \xrightarrow{\theta} \text{ } p(\theta | T(x), x) = p(\theta | T(x))
    \end{array}\]
  - Frequentist view
    \[\begin{array}{c}
    X \\
    \xrightarrow{\theta} \text{ } p(x | T(x), \theta) = p(x | T(x))
    \end{array}\]
  - The Neyman factorization theorem
    \[\begin{array}{c}
    X \\
    \xrightarrow{\text{ }} T(x) \\
    \xrightarrow{\theta}
    \end{array}\]
- \(T(x)\) is **sufficient** for \(\theta\) if
  \[p(x, T(x), \theta) = \psi_{1}(T(x), \theta) \psi_{2}(x, T(x))\]
  \[\Rightarrow p(x | \theta) = g(T(x), \theta) h(x, T(x))\]
Examples

- **Gaussian:**
  \[ \eta = \left[ \Sigma^{-\frac{1}{2}} \mu - \frac{1}{2} \text{vec}(\Sigma^{-1}) \right] \]
  \[ T(x) = \left[ x^T \text{vec}(xx^T) \right] \]
  \[ A(\eta) = \frac{1}{2} \mu^T \Sigma^{-1} \mu + \frac{1}{2} \log|\Sigma| \]
  \[ h(x) = (2\pi)^{-\frac{1}{2}} \]
  \[ \Rightarrow \mu_{MLE} = \frac{1}{N} \sum x \]

- **Multinomial:**
  \[ \eta = \left[ \ln \left( \frac{\pi_k}{\pi_k} \right) \right] \]
  \[ T(x) = \left[ x \right] \]
  \[ A(\eta) = -\ln \left( 1 - \sum_{i=1}^{K} \pi_i \right) - \ln \left( \sum_{i=1}^{K} e^{\eta} \right) \]
  \[ h(x) = 1 \]
  \[ \Rightarrow \mu_{MLE} = \frac{1}{N} \sum x \]

- **Poisson:**
  \[ \eta = \log \lambda \]
  \[ T(x) = x \]
  \[ A(\eta) = \lambda = e^\eta \]
  \[ h(x) = \frac{1}{x!} \]
  \[ \Rightarrow \mu_{MLE} = \frac{1}{N} \sum x \]

Generalized Linear Models (GLIMs)

- **The graphical model**
  - Linear regression
  - Discriminative linear classification
  - Commonality:
    - model \( E(Y|X) = f(\theta^T X) \)
    - What is \( f() \), the cond. dist. of \( Y \)?
    - What is \( f() \), the response function?

- **GLIM**
  - The observed input \( X \) is assumed to enter into the model via a linear combination of its elements \( \xi = \theta^T X \)
  - The conditional mean \( \mu \) is represented as a function \( f(\xi) \) of \( \xi \), where \( f \) is known as the response function
  - The observed output \( y \) is assumed to be characterized by an exponential family distribution with conditional mean \( \mu \).
GLIM, cont.

\[ p(y | \eta) = h(x) \exp{\eta^T (x)y - A(\eta)} \]

\[ \Rightarrow p(y | \eta) = h(x) \exp{\frac{1}{2} \left( \eta^T (x)y - A(\eta) \right)} \]

- The choice of exp family is constrained by the nature of the data \( Y \)
  - Example: \( y \) is a continuous vector \( \rightarrow \) multivariate Gaussian
  \( y \) is a class label \( \rightarrow \) Bernoulli or multinomial

- The choice of the response function
  - Following some mild constraints, e.g., \([0,1]\). Positivity …
  - Canonical response function: \( f = \psi^{-1}(\cdot) \)
    - In this case \( \frac{\partial}{\partial \theta} x \) directly corresponds to canonical parameter \( \eta \).

MLE for GLIMs with natural response

- Log-likelihood
  \[ \ell = \sum_n \log h(y_n) + \sum_n \left( \theta^T x_n y_n - A(\eta_n) \right) \]

- Derivative of Log-likelihood
  \[ \frac{d\ell}{d\theta} = \sum_n \left( x_n y_n - A(\eta_n) \frac{dA(\eta_n)}{d\eta_n} \frac{d\eta_n}{d\theta} \right) \]
  \[ = \sum_n (y_n - \mu_n) x_n \]
  \[ = X^T (y - \mu) \]

- Online learning for canonical GLIMs
  - Stochastic gradient ascent = least mean squares (LMS) algorithm:
    \[ \theta^{t+1} = \theta^t + \rho (y_n - \mu_n^t) x_n \]
    where \( \mu_n^t = \theta^T x_n \) and \( \rho \) is a step size

This is a fixed point function because \( \mu \) is a function of \( \theta \)
Batch learning for canonical GLIMs

- The Hessian matrix

\[ H = \frac{d^2 \mathcal{L}}{d \theta d \theta'} = \frac{d}{d \theta'} \sum_{n} (y_n - \mu_n) x_n = \sum_{n} x_n \frac{d \mu_n}{d \theta'} \]

\[ = -\sum_{n} x_n \frac{d \mu_n}{d \eta_n} \frac{d \eta_n}{d \theta'} \]

\[ = -\sum_{n} x_n \frac{d \mu_n}{d \eta_n} \zeta_n \text{ since } \eta_n = \theta^T x_n \]

\[ = -X^T W X \]

where \( X = [x_n^T] \) is the design matrix and

\[ W = \text{diag} \left( \frac{d \mu_1}{d \eta_1}, \ldots, \frac{d \mu_n}{d \eta_n} \right) \]

which can be computed by calculating the 2nd derivative of \( \mathcal{A}(\eta_n) \)

Iteratively Reweighted Least Squares (IRLS)

- Recall Newton-Raphson methods with cost function \( J \)

\[ \theta^{t+1} = \theta^t - H^{-1} \nabla_{\theta} J \]

- We now have

\[ \nabla_{\theta} J = X^T (y - \mu) \]

\[ H = -X^T W X \]

- Now:

\[ \theta^{t+1} = \theta^t + H^{-1} \nabla_{\theta} \ell \]

\[ = (X^T W^T X)^{-1} \left[ X^T W^T X \theta^t + X^T (y - \mu) \right] \]

\[ = (X^T W^T X)^{-1} X^T W^T z' \]

where the adjusted response is

\[ z' = X \theta^t + (W^t)^{-1} (y - \mu^t) \]

- This can be understood as solving the following "Iteratively reweighted least squares" problem

\[ \theta^{t+1} = \arg \min_{\theta} (z - X \theta)^T W (z - X \theta) \]
Example 1: logistic regression (sigmoid classifier)

- The condition distribution: a Bernoulli
  \[ p(y \mid x) = \mu(x)^y (1 - \mu(x))^{1-y} \]
  where \( \mu \) is a logistic function
  \[ \mu(x) = \frac{1}{1 + e^{-\eta(x)}} \]
- \( p(y \mid x) \) is an exponential family function, with
  - mean: \( E[y \mid x] = \mu = \frac{1}{1 + e^{-\eta(x)}} \)
  - and canonical response function \( \eta = \xi = \theta^T x \)
- IRLS
  \[ \frac{d\mu}{d\eta} = \mu(1 - \mu) \]
  \[ W = \left( \begin{array}{c}
\mu_1(1 - \mu_1) \\
\vdots \\
\mu_N(1 - \mu_N)
\end{array} \right) \]

Logistic regression: practical issues

- It is very common to use regularized maximum likelihood.
  \[ p(y = \pm 1 \mid x, \theta) = \frac{1}{1 + e^{\sigma(y \theta^T x)}} = \sigma(y \theta^T x) \]
  \[ p(\theta) = \text{Normal}(\theta \mid 0, \lambda^{-1} I) \]
  \[ l(\theta) = \sum_x \log(\sigma(y, \theta^T x)) - \frac{\lambda}{2} \theta^T \theta \]
  - IRLS takes \( \mathcal{O}(N d^3) \) per iteration, where \( N \) = number of training cases and \( d \) = dimension of input \( x \).
  - Quasi-Newton methods, that approximate the Hessian, work faster.
  - Conjugate gradient takes \( \mathcal{O}(Nd) \) per iteration, and usually works best in practice.
  - Stochastic gradient descent can also be used if \( N \) is large c.f. perceptron rule:
  \[ \nabla_\theta \ell = \left( 1 - \sigma(y, \theta^T x) \right) y \, x - \lambda \theta \]
Example 2: linear regression

- The condition distribution: a Gaussian
  
  \[ p(y | x, \theta, \Sigma) = \frac{1}{(2\pi)^{y/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (y - \mu(x))^T \Sigma^{-1} (y - \mu(x)) \right\} \]

- Rescale \( h(x) \exp \left\{ -\frac{1}{2} \Sigma^{-1} (\eta^T (x) y - A(\eta)) \right\} \)

  where \( \mu \) is a linear function

  \[ \mu(x) = \theta^T x = \eta(x) \]

- \( p(y | x) \) is an exponential family function, with
  
  \- mean: \( E[y | x] = \mu = \theta^T x \)
  
  \- and canonical response function \( \eta_1 = \xi = \theta^T x \)

- IRLS

  \[ \frac{d\mu}{d\eta} = \begin{bmatrix} 1 \\ \eta_2 \end{bmatrix} = \begin{bmatrix} \theta^T x \eta^T x^T z \\ \eta_2 \end{bmatrix} \Rightarrow \frac{d\mu}{d\eta} = \begin{bmatrix} \theta^T x \eta^T x^T (\theta + (y - \mu)) \\ \eta_2 \end{bmatrix} \Rightarrow \theta = (X^T X)^{-1} X^T y \]

  Steepest descent

  Normal equation