Lecture 11:

Bayesian Parameter Learning

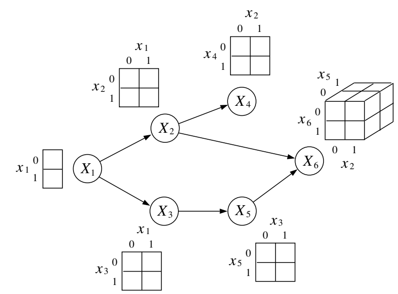
Kevin Murphy

October 25, 2004

MLE for general Bayes nets

- If we assume the parameters for each CPD are globally independent, and all nodes are fully observed, then the log-likelihood function decomposes into a sum of local terms, one per node:

$$\log p(\mathcal{D}|\theta) = \log \prod_m \prod_i p(\mathbf{x}_i^m|\mathbf{x}_{\pi_i}, \theta_i) = \sum_i \sum_m \log p(\mathbf{x}_i^m|\mathbf{x}_{\pi_i}, \theta_i)$$

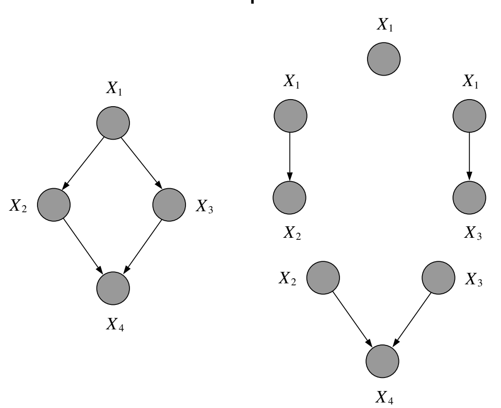


Example: A Directed Model

- Consider the distribution defined by the DAGM:

$$p(\mathbf{x}|\theta) = p(\mathbf{x}_1|\theta_1)p(\mathbf{x}_2|\mathbf{x}_1, \theta_2)p(\mathbf{x}_3|\mathbf{x}_1, \theta_3)p(\mathbf{x}_4|\mathbf{x}_2, \mathbf{x}_3, \theta_4)$$

- This is exactly like learning four separate small DAGMs, each of which consists of a node and its parents.



(a) (b)

MLE for Bayes nets with tabular CPDs

- Assume each CPD is represented as a table (multinomial) where

$$\theta_{ijk} \stackrel{\text{def}}{=} P(X_i = j|X_{\pi_i} = k)$$

- The sufficient statistics are just counts of family configurations

$$N_{ijk} \stackrel{\text{def}}{=} \sum_m I(X_i^m = j, X_{\pi_i}^m = k)$$

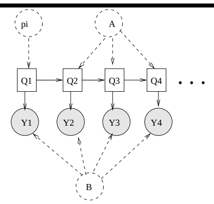
- The log-likelihood is

$$\ell = \log \prod_m \prod_{ijk} \theta_{ijk}^{N_{ijk}} = \sum_m \sum_{ijk} N_{ijk} \log \theta_{ijk}$$

- Using a Lagrange multiplier to enforce so $\sum_j \theta_{ijk} = 1$ we get

$$\hat{\theta}_{ijk}^{ML} = \frac{N_{ijk}}{\sum_{j'} N_{ij'k}}$$

Tied parameters

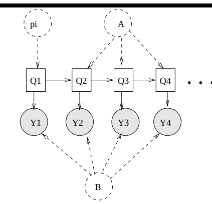


- Consider a time-invariant hidden Markov model (HMM)
  - State transition matrix $A(i,j) \stackrel{\text{def}}{=} P(X_t = j | X_{t-1} = i)$,
  - Discrete observation matrix $B(i,j) \stackrel{\text{def}}{=} P(Y_t = j | X_t = i)$
  - State prior $\pi(i) \stackrel{\text{def}}{=} P(X_1 = i)$.

  The joint is

$$P(X_{1:T}, Y_{1:T}|\theta) = P(X_1|\pi) \prod_{t=2}^{T} P(X_t|X_{t-1}, A) \prod_{t=1}^{T} P(Y_t|X_t; B)$$

Learning a fully observed HMM



- The log-likelihood is

$$\ell(\theta; D) = \sum_m \log P(X_1 = x_1^m|\pi)$$
$$+ \sum_{t=2}^{T} P(X_t = x_t^m | X_{t-1} = x_{t-1}^m, A) + \sum_{t=1}^{T} P(Y_t = y_t^m | X_t = x_t^m, B)$$

- We can optimize each parameter (A, B, π) separately.

Learning a Markov chain transition matrix

- Define $A(i,j) = P(X_t = j|X_{t-1} = i)$.
- A is a stochastic matrix: $\sum_j A(i,j) = 1$
- Each row of A is multinomial distribution.
- So MLE is the fraction of transitions from i to j

$$\hat{A}_{ML}(i,j) = \frac{\#i \rightarrow j}{\sum_k \#i \rightarrow k} = \frac{\sum_m \sum_{t=2}^{T} I(X_{t-1}^m = i, X_t^m = j)}{\sum_m \sum_{t=2}^{T} I(X_{t-1}^m = i)}$$

- If the states X_t represent words, this is called a *bigram language model*.
- Note that $\hat{A}_{ML}(i,j) = 0$ if the particular i, j pair did not occur in the training data; this is called the *sparse data problem*.
- We will solve this using a prior.

Dirichlet priors

- Let $X \in \{1, \ldots, K\}$ have a multinomial distribution

$$P(X|\theta) = \theta_1^{I(X=1)} \theta_2^{I(X=2)} \cdots \theta_K^{I(X=k)}$$

- For a set of data $X^1, \ldots, X^N$, the sufficient statistics are the counts $N_i = \sum_n I(X_n = i)$.
- Consider a Dirichlet prior with hyperparameters α

$$p(\theta|\alpha) = \mathcal{D}(\theta|\alpha) = \frac{1}{Z(\alpha)} \cdot \theta_1^{\alpha_1 - 1} \cdot \theta_2^{\alpha_2 - 1} \cdots \theta_K^{\alpha_K - 1}$$

  where $Z(\alpha)$ is the normalizing constant
- The Dirichlet prior is *conjugate* to (has the same form as) the multinomial likelihood.
- The α_k act like pseudo (virtual) counts.

Normalization constant

- $Z(\alpha)$ is the normalizing constant

$$\begin{aligned} Z(\alpha) &= \int \cdots \int \theta_1^{\alpha_1 - 1} \cdots \theta_K^{\alpha_K - 1} d\theta_1 \cdots d\theta_K \\ &= \frac{\Gamma(\sum_{i=1}^K \alpha_i)}{\prod_{i=1}^K \Gamma(\alpha_i)} \end{aligned}$$

- $\Gamma(\alpha)$ is the gamma function:

$$\Gamma(\alpha) = \int_0^\infty t^{\alpha-1} e^{-t} dt$$

- For integers, $\Gamma(n+1) = n!$

Dirichlet posterior

- Likelihood, prior, posterior:

$$\begin{aligned} P(\vec{N}|\vec{\theta}) &= \prod_{i=1}^K \theta_i^{N_i} \\ p(\theta|\alpha) &= \mathcal{D}(\theta|\alpha) = \frac{1}{Z(\alpha)} \cdot \theta_1^{\alpha_1 - 1} \cdot \theta_2^{\alpha_2 - 1} \cdots \theta_K^{\alpha_K - 1} \\ p(\theta|\vec{N}, \vec{\alpha}) &= \frac{1}{Z(\alpha) p(\vec{N}|\alpha)} \theta_1^{\alpha_1 + N_1} \cdots \theta_K^{\alpha_K + N_k} \\ &= \mathcal{D}(\alpha_1 + N_1, \ldots, \alpha_K + N_K) \end{aligned}$$

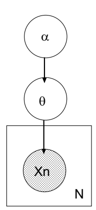
- Marginal likelihood (evidence):

$$P(\vec{N}|\vec{\alpha}) = \int p(\vec{N}|\vec{\alpha}) p(\vec{\theta}|\vec{\alpha}) d^K\theta = \frac{Z(\vec{N} + \vec{\alpha})}{Z(\vec{\alpha})}$$

Hierarchical Bayesian models

- θ are the parameters for the likelihood $p(X|\theta)$
- α are the parameters for the prior $p(\theta|\alpha)$.
- We can have hyper-hyper-parameters, etc.
- We stop when the choice of hyper^n-parameters makes no difference to the marginal likelihood; typically make hyper-parameters constants.
- Type-II maximum likelihood (empirical Bayes) = computing point estimates of α:

$$\hat{\alpha}_{ML} = \arg\max_\alpha p(\vec{\alpha}|\vec{N}) = \arg\max_\alpha p(\vec{N}|\vec{\alpha}) p(\vec{\alpha})$$



Beta priors

- Consider a coin toss $X \in \{h, t\}$.
- The Dirichlet distribution becomes the beta distribution:

$$p(\theta) = \frac{1}{Z(\alpha)} \theta^{\alpha_h - 1} (1 - \theta)^{\alpha_t - 1}$$

- If $\alpha_h = \alpha_t = 1$, we have a uniform (Laplace) prior.
- The posterior mean (predicted probability of heads) is

$$\begin{aligned} P(X = h|\alpha_h, \alpha_t) &= \int_0^1 d\theta \ \ P(X = 1|\theta) p(\theta) \\ &= \int_0^1 d\theta \ \ \theta p(\theta) = \frac{\alpha_h}{\alpha_h + \alpha_t} \end{aligned}$$

- Hence α_h is the number of virtual heads we have seen in our prior "database"; similarly for α_t.
- The strength of the prior is measured by the equivalent sample size $\alpha_h + \alpha_t$.

Sequential Bayesian updating

- Start with beta prior $p(\theta|\alpha_h, \alpha_t) = \mathcal{B}(\theta; \alpha_h, \alpha_t)$.
- Observe N trials with N_h heads and N_t tails. Posterior becomes

$$p(\theta|\alpha_h, \alpha_t, N_h, N_t) = \mathcal{B}(\theta; \alpha_h + N_h, \alpha_t + N_t) = \mathcal{B}(\theta; \alpha'_h, \alpha'_t)$$

- Observe another N' trials with N'_h heads and N'_t tails. Posterior becomes

$$\begin{aligned} p(\theta|\alpha'_h, \alpha'_t, N'_h, N'_t) &= \mathcal{B}(\theta; \alpha'_h + N'_h, \alpha'_t + N'_t) \\ &= \mathcal{B}(\theta; \alpha_h + N_h + N'_h, \alpha_t + N_t + N'_t) \end{aligned}$$

- So sequentially absorbing data in any order is equivalent to batch update.

Effect of prior strength

- Let $N = N_h + N_t$ be number of samples (observations).
- Let N' be the number of pseudo observations (strength of prior) and define the prior means

$$\alpha_h = N'\alpha'_h, \quad \alpha_t = N'\alpha'_t, \quad \alpha'_h + \alpha'_t = 1$$

- Then posterior mean is a convex combination of the prior mean and the MLE:

$$\begin{aligned} P(X = h|\alpha_h, \alpha_t, N_h, N_t) &= \frac{\alpha_h + N_h}{\alpha_h + N_h + \alpha_t + N_t} \\ &= \frac{N'\alpha'_h + N_h}{N + N'} \\ &= \frac{N'}{N + N'}\alpha'_h + \frac{N}{N + N'}\frac{N_h}{N} \\ &= \lambda\alpha'_h + (1 - \lambda)\frac{N_h}{N} \end{aligned}$$

where $\lambda = N'/(N + N')$.

Effect of prior strength

- Suppose we have a uniform prior $\alpha'_h = \alpha'_t = 0.5$, and we observe $N_h = 3$, $N_t = 7$.
- Weak prior $N' = 2$. Posterior prediction:

$$P(X = h|\alpha_h = 1, \alpha_t = 1, N_h = 3, N_t = 7) = \frac{3 + 1}{3 + 1 + 7 + 1} = \frac{1}{3} \approx 0.33$$

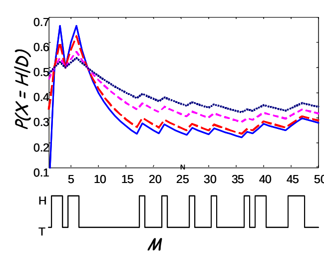
- Strong prior $N' = 20$. Posterior prediction:

$$\frac{3 + 10}{3 + 10 + 7 + 10} = \frac{13}{30} \approx 0.43$$

- However, if we have enough data, it washes away the prior. e.g., $N_h = 300$, $N_t = 700$. Estimates are $\frac{300+1}{1000+2}$ and $\frac{300+10}{1000+20}$, both of which are close to 0.3

Prior smooths parameter estimates

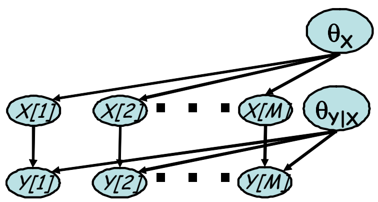
- The MLE can change dramatically with small sample sizes.
- The MAP estimate changes much more smoothly (depending on the strength of the prior).
- This is called regularization.
- Lower blue=MLE, red = beta(1,1), pink = beta(5,5), upper blue = beta(10,10)



Bayesian parameter estimation for general BNs

- Defn 13.4.1: *global parameter independence* means $p(\theta) = \prod_i p(\theta_i)$, where θ_i are the parameters for CPD for X_i.
- If we assume global parameter independence, and have fully observed data, then the parameter posterior decomposes into a sum of local terms, one per node:

$$\log p(\theta|\mathcal{D}) = \sum_i \sum_m \log p(\mathbf{x}_i^m | \mathbf{x}_{\pi_i}, \theta_i) + \log p(\theta_i)$$

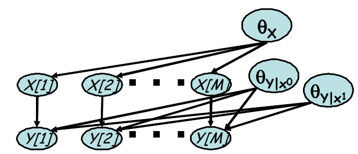


Bayesian parameter estimation for BNs with tabular CPDs

- Defn 13.4.4: *local parameter independence* means $p(\theta_i) = \prod_k p(\theta_{i,\cdot,k})$, where $\theta_{i,j,k} = P(X_i = j | X_{\pi_i} = k)$ is the row of the CPT corresponding to conditioning case k.
- If we assume global and local parameter independence, and have fully observed data, then the parameter posteriors are

$$P(\theta_{i,\cdot,k}|D) = \mathcal{D}(\alpha_{i,1,k} + N_{i,1,k}, \ldots, \alpha_{i,S,k} + N_{i,S,k})$$

- Posterior for $\theta_{y|x^0}$ and $\theta_{y|x^1}$ is factorized despite v-structure on y_m because CPT acts like a multiplexer.



Where do the priors come from?

- We can define $\alpha'_{ijk} = N' P'(X_i = j | X_{\pi_i} = k)$, where N' is the strength of our prior and P' is some Bayes net that summarizes our virtual database of pseudo counts.
- This is called the BDe (Bayesian-Dirichlet likelihood equivalent) prior.
- Type-II ML = learning P' from data.

Example of Bayesian parameter learning

- Suppose we draw $X_{1:37}^{1:N} \sim P(X_{1:37}|\theta^*)$ from the ICU-Alarm BN.
- Then we estimate

$$\hat{\theta} = \arg\max_\theta P(X^{1:N}|\theta) P(\theta|\alpha', N')$$

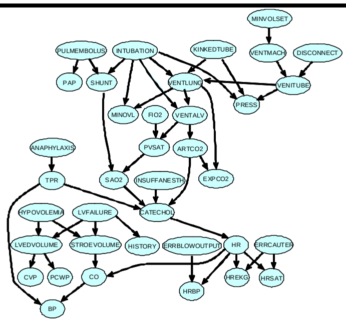
for different sample sizes N and prior strengths N' (with uniform prior $\alpha'_{ijk} = 1/|X_i|$).

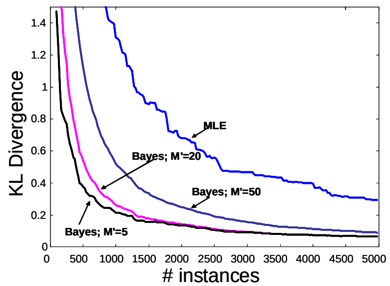
- We compare answers using the Kullback-Leibler divergence

$$KL\left(P(X|\theta^*)||P(X|\hat{\theta})\right) = \sum_x P(x|\theta^*) \log \frac{P(x|\theta^*)}{P(x|\hat{\theta})}$$

where $KL(P||Q) \geq 0$ measures the "distance" of the approximation Q from truth P.

Example of Bayesian parameter learning





- If $N_{ijk} = 0$ in training but $P(X_i = j|X_\pi = k, \theta^*) > 0$, then $KL(P^*||\hat{P}) = \infty$, since

$$KL\left(P(X|\theta^*)||P(X|\hat{\theta})\right) = \sum_x P(x|\theta^*) \log \frac{P(x|\theta^*)}{P(x|\hat{\theta})}$$

- Dirichlet smoothing helps a lot!
- Optimal prior strength = 5.

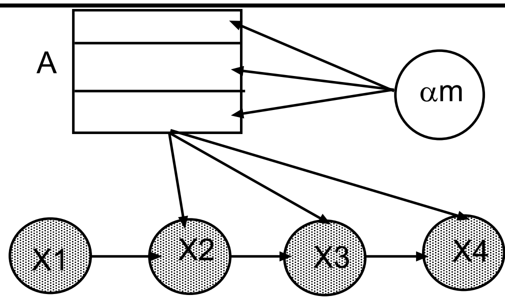
Application: language modeling

- A bigram model predicts $P(X_t = j|X_{t-1} = i, \theta) = \theta_{ij}$.
- Often the data is sparse so $N_{ij} = 0$ so $\theta_{ij} = 0$.
- A standard hack is to use *backoff smoothing* or *deleted interpolation*:

$$\hat{P}(X_t|X_{t-1}) = \lambda f_{x_t} + (1-\lambda) f_{x_t|x_{t-1}}$$

  where λ is set by cross valdiation and f_i and $f_{j|i}$ are empirical frequencies.
- A similar effect can be gotten using a hierarchical prior.

Application: language modeling

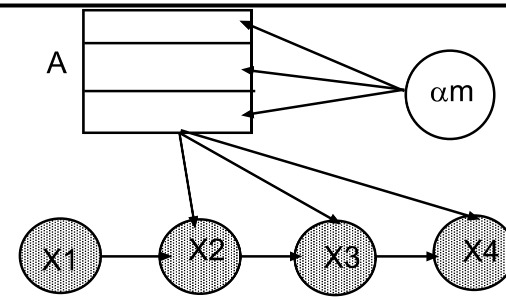


- Assign the same Dirichlet prior αm_i to each row of the transition matrix.
- So the prediction is

$$P(i|j, D, \alpha m) = \frac{f_{i|j} + \alpha m_i}{\sum_{i'} f_{i'|j} + \alpha m_{i'}} = \lambda_j m_i + (1-\lambda_j) f_{i|j}$$

  where $\lambda_j = \frac{\alpha}{f_j + \alpha}$.
- This is like adaptive deleted interpolation.

Application: language modeling



- We can optimize the hyperparameters using numerical methods (e.g., conjugate gradient), which is faster than cross validation.

$$(\alpha m)^{MAP} = \arg\max P(D|\alpha m)$$

- We could consider more realistic priors, e.g., mixtures of Dirichlets to account for types of words (adjectives, verbs, etc.)

CPDs for continuous nodes

- So far we have considered the case where $p(y|x,\theta)$ can be represented as a multinomial (table).
- Now we consider the case where some nodes may be continuous.

X	Y	$p(Y\|X)$
$\mathbb{R}^n$	$\mathbb{R}^m$	regression
$\mathbb{R}^n$	$\{0,1\}$	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

Exponential Family

- For a numeric random variable $\mathbf{x}$

$$p(\mathbf{x}|\eta) = h(\mathbf{x})\exp\{\eta^\top T(\mathbf{x}) - A(\eta)\}$$
$$= \frac{1}{Z(\eta)} h(\mathbf{x})\exp\{\eta^\top T(\mathbf{x})\}$$

is an exponential family distribution with *natural (canonical) parameter* η.

- Function $T(\mathbf{x})$ is a *sufficient statistic*.
- Function $A(\eta) = \log Z(\eta)$ is the log normalizer.
- Examples: Bernoulli, multinomial, Gaussian, Poisson, gamma,...
- A distribution $p(x)$ has finite sufficient statistics (independent of number of data cases) iff it is in the exponential family.

MLE for Exponential Family

- For iid data, the log-likelihood is

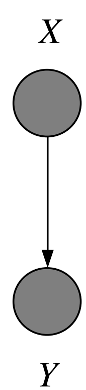
$$\ell(\eta;\mathcal{D}) = \log \prod_m h(x^m)\exp\left(\eta^T T(x^m) - A(\eta)\right)$$
$$= \left(\sum_m \log h(\mathbf{x}^m)\right) - MA(\eta) + \left(\eta^\top \sum_m T(\mathbf{x}^m)\right)$$

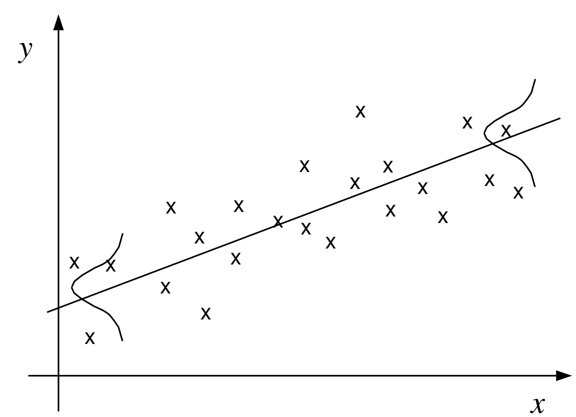
- Take derivatives and set to zero:

$$\frac{\partial \ell}{\partial \eta} = \textstyle\sum_m T(\mathbf{x}^m) - M\frac{\partial A(\eta)}{\partial \eta} = 0$$
$$\Rightarrow \frac{\partial A(\eta)}{\partial \eta} = \frac{1}{M}\textstyle\sum_m T(\mathbf{x}^m)$$
$$\hat{\mu}_{\mathrm{ML}} = \frac{1}{M}\textstyle\sum_m T(\mathbf{x}^m)$$

- This amounts to moment matching.
- We can infer the canonical parameters using $\hat{\eta}_{ML} = \psi(\hat{\mu}_{ML})$

Linear Regression





Multivariate Linear Regression

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

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

- Log-likelihood

$$\ell = -\frac{1}{2}\sum_m \log|\Sigma| - \frac{1}{2}\sum_m (y_m - Ax_m)^T \Sigma^{-1}(y_m - Ax_m)$$

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

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

where $A = M$, $a = -x_m$ and $b = y_m$.

Multivariate Linear Regression

- Log-likelihood:

$$\ell = -\frac{1}{2}\sum_m \log|\Sigma| - \frac{1}{2}\sum_m (y_m - Ax_m)^T \Sigma^{-1}(y_m - Ax_m)$$

- Using

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

we have

$$\begin{aligned}\frac{\partial \ell}{\partial A} &= -\frac{1}{2}\sum_m 2\Sigma^{-1}(y_m - Ax_m)x_m^T \\ &= -\Sigma^{-1}\sum_m y_m x_m^T - A\sum_m x_m x_m^T \\ &\stackrel{\text{def}}{=} -\Sigma^{-1}S_{YX'} - AS_{XX'} = 0\end{aligned}$$

where $S_{YX'}$ and $S_{XX'}$ are the sufficient statistics. Hence

$$A = S_{YX'}S_{XX'}^{-1}$$

1D Linear Regression

- For the vector case,

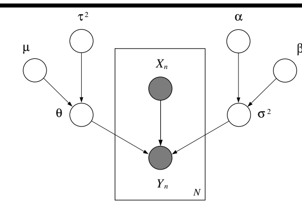
$$A = S_{YX'}S_{XX'}^{-1}$$

where $S_{YX'} = \sum_m y_m x_m^T$ and $S_{XX'} = \sum_m x_m x_m^T$.

- In the special case of scalar outputs, let $A = \theta^T$, and the design matrix $X = [x_m^T]$ stacked as rows and $Y = [y_m]$ a column vector. Then we get the normal equations

$$\theta = (X^T X)^{-1} X^T Y$$

Bayesian 1D Linear Regression



- For scalar (1D) output

$$p(y_n|x_n, \theta, \sigma^2)p(\theta|\mu, \tau^2)p(\sigma^2|\alpha, \beta)$$

Gaussian × Gaussian × Gamma

- For vector output

$$p(y_n|x_n, A, \Sigma)p(A|\mu, \tau^2)p(\Sigma|\alpha, \beta)$$

Gaussian × matrix-Gaussian × Wishart

MLE for Generalized Linear Models

- GLIM with scale parameter ϕ and canonical parameter $\eta = \theta^T x$:

$$p(y|x,\theta,\phi) = h(y,\phi)\exp\left(\frac{\eta^T y - A(\eta)}{\phi}\right)$$

- Log-likelihood

$$\ell = \sum_n \log h(y_n) + \frac{1}{\phi}\sum_n \left(\theta^T x_n y_n - A(\eta_n)\right)$$

- Derivative of Log-likelihood

$$\begin{aligned}\frac{d\ell}{d\theta} &= \frac{1}{\phi}\sum_n \left(x_n y_n - \frac{dA(\eta_n)}{d\eta_n}\frac{d\eta_n}{d\theta}\right) \\ &= \frac{1}{\phi}\sum_n (y_n - \mu_n)\, x_n \\ &= \frac{1}{\phi}X^T(y-\mu)\end{aligned}$$

Batch learning for canonical GLIMs

- Hessian

$$\begin{aligned}H &= \frac{d^2\ell}{d\theta d\theta^T} = \frac{d}{d\theta^T}\frac{1}{\phi}\sum_n x_n(y_n-\mu_n) = -\frac{1}{\phi}\sum_n x_n \frac{d\mu_n}{d\theta^T} \\ &= -\frac{1}{\phi}\sum_n x_n \frac{d\mu_n}{d\eta_n}\frac{d\eta_n}{d\theta^T} \\ &= -\frac{1}{\phi}\sum_n x_n \frac{d\mu_n}{d\eta_n} x_n^T \text{ since } \eta_n = \theta^T x_n \\ &= -\frac{1}{\phi}X^T W X\end{aligned}$$

where $X = [x_n^T]$ is the design matrix and

$$W = \text{diag}\left(\frac{d\mu_1}{d\eta_1}, \ldots, \frac{d\mu_N}{d\eta_N}\right)$$

Online learning for canonical GLIMs

- Derivative of Log-likelihood

$$\frac{d\ell}{d\theta} = \frac{1}{\phi}\sum_n (y_n - \mu_n)\, x_n$$

- Stochastic gradient ascent = least mean squares (LMS) algorithm:

$$\theta^{t+1} = \theta^t + \rho(y_n - \mu_n^t)x_n$$

where $\mu_n^t = \theta^{(t)T} x_n$ and ρ is a step size.

Iteratively Reweighted Least Squares (IRLS)

$$\begin{aligned}\nabla_\theta \ell &= \frac{1}{\phi}X^T(y-\mu) \\ H &= -\frac{1}{\phi}X^T W x \\ \theta^{t+1} &= \theta^T + H^{-1}\nabla_\theta \ell \\ &= (X^T W^t X)^{-1}\left[X^T W^t X\theta^t + X^T(y-\mu^t)\right] \\ &= (X^T W^t X)^{-1} X^T W^t z^t\end{aligned}$$

where the adjusted response is

$$z^t = X\theta^t + (W^t)^{-1}(y-\mu^t)$$

We iteratively reoptimize

$$\theta^{t+1} = \arg\min_\theta (z - X\theta)^T W (z - X\theta)$$

This Newton-Raphson procedure will (usually) find the global optimum starting from $\theta = 0$.

IRLS for logistic regression (sigmoid classifier)

$$\mu = \sigma(\eta) = \frac{1}{1+e^{-\eta}} = \sigma(\theta^T x) = p(y=1|x,\theta)$$

$$\frac{d\mu}{d\eta} = \mu(1-\mu)$$

$$W = \begin{pmatrix} \mu_1(1-\mu_1) & & \\ & \ddots & \\ & & \mu_n(1-\mu_n) \end{pmatrix}$$

Logistic regression: practical issues

- It is very common to use penalized maximum likelihood.

$$p(y=\pm 1|x,\theta) = \sigma(y\theta^T x) = \frac{1}{1+exp(-y\theta^T x)}$$

$$p(\theta) \sim \mathcal{N}(0, \lambda^{-1} I)$$

$$\ell(\theta) = \sum_n \log \sigma(y_n \theta^T x_n) - \frac{\lambda}{2}\theta^T\theta$$

- IRLS takes $O(Nd^2)$ per iteration, where N = number of training cases and d = size of input x.
- Quasi-Newton methods, that approximate the Hessian, work faster.
- Conjugate gradient takes $O(Nd)$ per iteration, and usually works best in practice.
- Stochastic gradient descent can also be used if N is large c.f. perceptron rule:

$$\nabla_\theta \ell(\theta) = (1-\sigma(y_n\theta^T x_n))y_n x_n - \lambda\theta$$