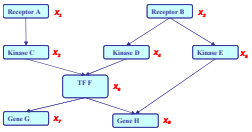
School of Computer Science
Carnegie Mellon

Learning two-node GMs

Probabilistic Graphical Models (10-708)

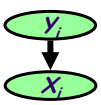
Lecture 8, Oct 10, 2007



Eric Xing

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

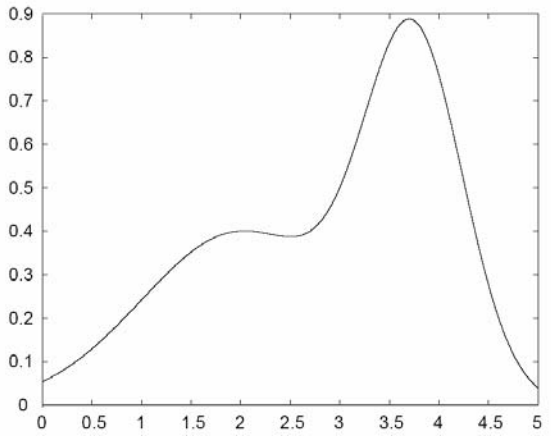
Two-node BNs



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

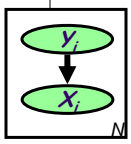


Conditional Gaussian

- The data:

$$\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \cdots, (x_N, y_N)\}$$

GM:



- Both nodes are observed:
  - Y is a class indicator vector

$$p(y_n) = \text{multi}(y_n : \pi) = \prod_k \pi_k^{y_n^k}$$

  - X is a conditional Gaussian variable with a class-specific mean

$$p(x_n \mid y_n^k = 1, \mu, \sigma) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{ \frac{1}{2\sigma^2} (x_n - \mu_k)^2 \right\}$$

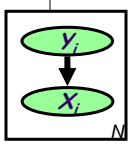
$$p(x \mid y, \mu, \sigma) = \prod_n \left(\prod_k N(x_n : \mu_k, \sigma)^{y_n^k} \right)$$

MLE of conditional Gaussian

- Data log-likelihood

$$\ell(\boldsymbol{\theta}; D) = \log \prod_n p(x_n, y_n) = \log \prod_n p(y_n \mid \pi) p(x_n \mid y_n, \mu, \sigma)$$
$$= \sum_n \log \prod_k \pi_k^{y_n^k} + \sum_n \log \prod_k N(x_n; \mu_k, \sigma)^{y_n^k}$$
$$= \sum_n \sum_k y_n^k \log \pi_k - \sum_n \sum_k y_n^k \tfrac{1}{2\sigma^2} (x_n - \mu_k)^2 + C$$

GM:



- MLE

$$\hat{\pi}_{k,MLE} = \arg\max \ell(\boldsymbol{\theta}; D), \quad \Rightarrow \frac{\partial}{\partial \pi_k} \ell(\boldsymbol{\theta}; D) = 0, \forall k, \quad \text{s.t.} \sum_k \pi_k = 1$$

$$\Rightarrow \hat{\pi}_{k,MLE} = \sum_n y_n^k / N = n_k / N$$

the fraction of samples of class m

$$\hat{\mu}_{k,MLE} = \arg\max \ell(\boldsymbol{\theta}; D), \quad \Rightarrow \hat{\mu}_{k,MLE} = \frac{\sum_n y_n^k x_n}{\sum_n y_n^k} = \frac{\sum_n y_n^k x_n}{n_k}$$

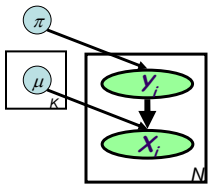
the average of samples of class m

Bsyesian estimation of conditional Gaussian

- Prior:

$$P(\vec{\pi} \mid \vec{\alpha}) = \text{Dir}(\vec{\pi} : \vec{\alpha})$$
$$P(\mu_k \mid \nu) = \text{Normal}(\mu_k : \nu, \tau)$$

GM:

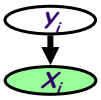


- Posterior mean (Bayesian est.)

$$\pi_{k,Bayes} = \frac{N}{N+|\alpha|} \hat{\pi}_{k,ML} + \frac{|\alpha|}{N+|\alpha|} \frac{\alpha_k}{|\alpha|} = \frac{n_k + \alpha_k}{N+|\alpha|}$$

$$\mu_{k,Bayes} = \frac{n_k/\sigma^2}{n_k/\sigma^2 + 1/\tau^2} \hat{\mu}_{k,ML} + \frac{1/\tau^2}{n_k/\sigma^2 + 1/\tau^2} \nu, \quad \text{and} \quad \sigma^2_{Bayes} = \left(\frac{N}{\sigma^2} + \frac{1}{\tau^2} \right)^{-1}$$

Classification



- From conditional density modeling to classification:
  - The joint probability of a datum and it label is:

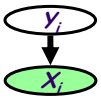
$$p(x_n, y_n^k = 1 \mid \mu, \sigma) = p(y_n^k = 1) \times p(x_n \mid y_n^k = 1, \mu, \sigma)$$
$$= \pi_k \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{ \frac{1}{2\sigma^2} (x_n - \mu_k)^2 \right\}$$

  - Given a datum x_n, we predict its label using the conditional probability of the label given the datum:

$$p(y_n^k = 1 \mid x_n, \mu, \sigma) = \frac{\pi_k \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{ \frac{1}{2\sigma^2} (x_n - \mu_k)^2 \right\}}{\sum_{k'} \pi_{k'} \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{ \frac{1}{2\sigma^2} (x_n - \mu_{k'})^2 \right\}}$$

- This is basic inference
  - introduce evidence, and then normalize

Naïve Bayes Classifier

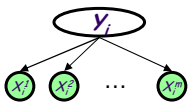


- When X is multivariate-Gaussian vector:
  - The joint probability of a datum and it label is:

$$p(\vec{x}_n, y_n^k = 1 \mid \vec{\mu}, \Sigma) = p(y_n^k = 1) \times p(\vec{x}_n \mid y_n^k = 1, \vec{\mu}, \Sigma)$$
$$= \pi_k \frac{1}{(2\pi|\Sigma|)^{1/2}} \exp\left\{ -\frac{1}{2} (\vec{x}_n - \vec{\mu}_k)^T \Sigma^{-1} (\vec{x}_n - \vec{\mu}_k) \right\}$$

  - The naïve Bayes simplification

$$p(x_n, y_n^k = 1 \mid \mu, \sigma) = p(y_n^k = 1) \times \prod_j p(x_n^j \mid y_n^k = 1, \mu_{k,j}, \sigma_{k,j})$$
$$= \pi_k \prod_j \frac{1}{(2\pi\sigma_{k,j}^2)^{1/2}} \exp\left\{ \frac{1}{2\sigma_{k,j}^2} (x_n^j - \mu_{k,j})^2 \right\}$$

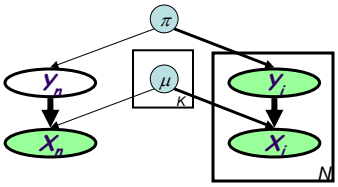


  - More generally: $p(x_n, y_n \mid \eta, \pi) = p(y_n \mid \pi) \times \prod_{j=1}^{m} p(x_n^j \mid y_n, \eta)$
    - Where $p(. \mid .)$ is an arbitrary conditional (discrete or continuous) 1-D density

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 π, μ and σ from data first, and then …

GM:



$$p(y_n^k = 1 \mid x_n, \mu, \sigma, \pi) = \frac{p(y_n^k = 1, x_n \mid \mu, \sigma, \pi)}{p(x_n \mid \mu, \sigma, \pi)} = \frac{\pi_k N(x_n, \mid \mu_k, \sigma)}{\sum_j \pi_j N(x_n, \mid \mu_j, \sigma)}$$

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

The predictive distribution

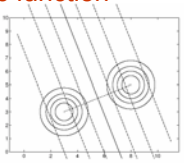
- Understanding the predictive distribution

$$p(y_n^k = 1 \mid x_n, \mu, \sigma, \pi) = \frac{p(y_n^k = 1, x_n \mid \mu, \sigma, \pi)}{p(x_n \mid \mu, \sigma)} = \frac{\pi_k N(x_n, \mid \mu_k, \sigma)}{\sum_j \pi_j N(x_n, \mid \mu_j, \sigma)} \quad *$$

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

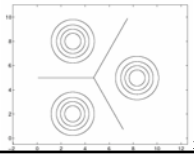
$$p(y_n^1 = 1 \mid x_n) = \frac{1}{1 + \frac{\pi_2 \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{\frac{1}{2\sigma^2}(x_n - \mu_2)^2\right\}}{\pi_1 \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{\frac{1}{2\sigma^2}(x_n - \mu_1)^2\right\}}} = \frac{1}{1 + \exp\left\{-x_n \frac{1}{\sigma^2}(\mu_1 - \mu_2) + \log\frac{\pi_2}{\pi_1}\right\}}$$

$$= \frac{1}{1 + e^{-\theta^T x_n}}$$



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

$$p(y_n^k = 1 \mid x_n) = \frac{e^{-\theta_k^T x_n}}{\sum_j e^{-\theta_j^T x_n}}$$

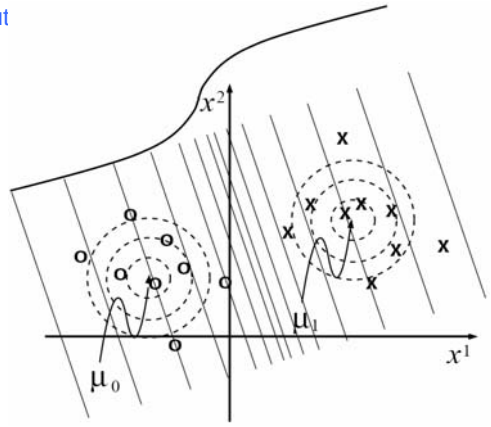


Discussion

- We've seen how to learning two-node model $p(y_n, x_n)$, but in certain problems the goal is to learning $p(y_n \mid x_n)$
- Can we model $p(y_n \mid x_n)$ directly?
- How?

Generative and discriminative classifiers

- Generative:
  - Modeling the joint distribut of all data
- Discriminative:
  - Modeling only points at the boundary
  - How? Regression!



Linear regression

- The data:

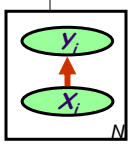
$$\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \cdots, (x_N, y_N)\}$$

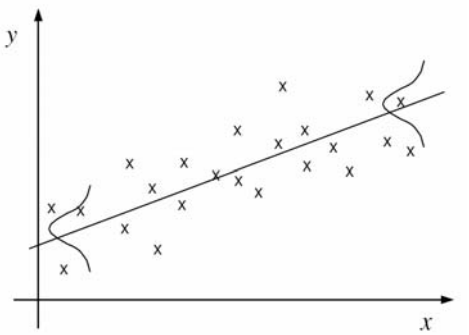
- Both nodes are observed:
  - X is an input vector
  - Y is a response vector

  (we first consider y as a generic continuous response vector, then we consider the special case of classification where y is a discrete indicator)

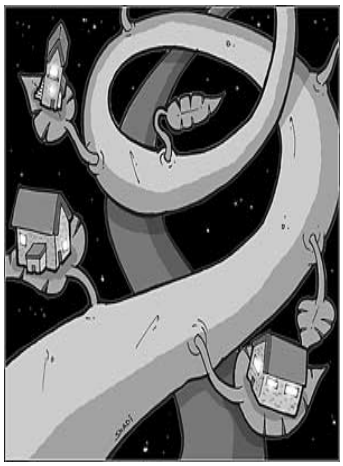
- A regression scheme can be used to model $p(y|x)$ directly, rather than $p(x,y)$

GM:





Apartment hunting



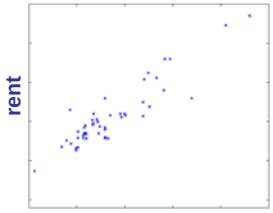
- Now you've moved to Pittsburgh!!

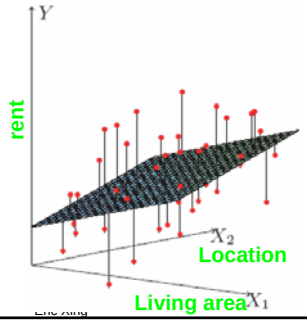
  And you want to find the **most reasonably priced** apartment satisfying your **needs:**

  square-ft., # of bedroom, distance to campus …

Living area (ft²)	# bedroom	Rent ($)
230	1	600
506	2	1000
433	2	1100
109	1	500
…		
150	1	?
270	1.5	?

The learning problem



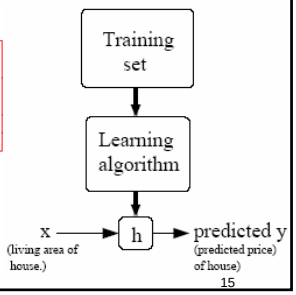


- Features:
  - Living area, distance to campus, # bedroom …
  - Denote as $\mathbf{x}=[x_1, x_2, \ldots x_k]$
- Target:
  - Rent
  - Denoted as y
- Training set:

$$\mathbf{X} = \begin{bmatrix} -- & \mathbf{x}_1 & -- \\ -- & \mathbf{x}_2 & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n & -- \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,k} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,k} \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} - & \mathbf{y}_1 & - \\ - & \mathbf{y}_2 & - \\ \vdots & \vdots & \vdots \\ - & \mathbf{y}_n & - \end{bmatrix}$$

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:
  - then we have the following general representation of the linear function:
- Our goal is to pick the optimal θ . How!
  - We seek θ that minimize the following **cost function**:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(\hat{y}_i(\bar{x}_i) - y_i)^2$$

The Least-Mean-Square (LMS) method

- The Cost Function:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^T\theta - y_i)^2$$

- Consider a **gradient descent** algorithm:

$$\theta_j^{t+1} = \theta_j^t - \alpha\frac{\partial}{\partial\theta_j}J(\theta)\Big|_t$$

The Least-Mean-Square (LMS) method

- Now we have the following descent rule:

$$\theta_j^{t+1} = \theta_j^t + \alpha\sum_{i=1}^{n}(y_n - \mathbf{x}_n^T\theta^t)x_{n,i}$$

- For a single training point, we have:

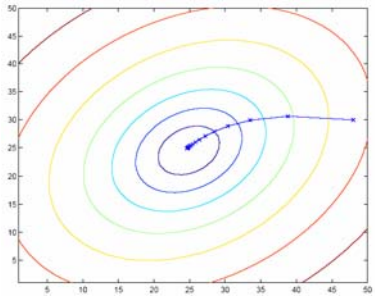
  - This is known as the LMS update rule, or the Widrow-Hoff learning rule
  - This is actually a "**stochastic**", "**coordinate**" descent algorithm
  - This can be used as a **on-line** algorithm

The Least-Mean-Square (LMS) method

- Steepest descent
  - Note that:

$$\nabla_\theta J = \left[\frac{\partial}{\partial\theta_1}J,\ldots,\frac{\partial}{\partial\theta_k}J\right]^T = -\sum_{i=1}^{n}(y_n - \mathbf{x}_n{}^T\theta)\mathbf{x}_n$$

$$\theta^{t+1} = \theta^t + \alpha\sum_{i=1}^{n}(y_n - \mathbf{x}_n{}^T\theta^t)\mathbf{x}_n$$



  - This is as a **batch** gradient descent algorithm

Some matrix derivatives

- For $f:\mathbb{R}^{m\times n}\mapsto\mathbb{R}$, define:

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial}{\partial A_{11}}f & \cdots & \frac{\partial}{\partial A_{1n}}f \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial A_{1m}}f & \cdots & \frac{\partial}{\partial A_{mn}}f \end{bmatrix}$$

- Trace:

$$\mathrm{tr}A = \sum_{i=1}^{n}A_{ii}\ , \qquad \mathrm{tr}a = a\ , \qquad \mathrm{tr}ABC = \mathrm{tr}CAB = \mathrm{tr}BCA$$

- Some fact of matrix derivatives (without proof)

$$\nabla_A \mathrm{tr}AB = B^T\ , \qquad \nabla_A \mathrm{tr}ABA^TC = CAB + C^TAB^T\ , \qquad \nabla_A|A| = |A|\left(A^{-1}\right)^T$$

The normal equations

- Write the cost function in matrix form:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^T\theta - y_i)^2 = \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) = \frac{1}{2}\left(\theta^T X^T X\theta - \theta^T X^T\vec{y} - \vec{y}^T X\theta + \vec{y}^T\vec{y}\right)$$

$$\mathbf{X} = \begin{bmatrix} -- & \mathbf{x}_1 & -- \\ -- & \mathbf{x}_2 & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n & -- \end{bmatrix} \qquad \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

- To minimize $J(\theta)$, take derivative and set to zero:

$$\nabla_\theta J = \frac{1}{2}\nabla_\theta \mathrm{tr}\left(\theta^T X^T X\theta - \theta^T X^T\vec{y} - \vec{y}^T X\theta + \vec{y}^T\vec{y}\right) = \frac{1}{2}\left(\nabla_\theta \mathrm{tr}\,\theta^T X^T X\theta - 2\nabla_\theta \mathrm{tr}\vec{y}^T X\theta + \nabla_\theta \mathrm{tr}\vec{y}^T\vec{y}\right) = \frac{1}{2}\left(X^T X\theta + X^T X\theta - 2X^T\vec{y}\right) = X^T X\theta - X^T\vec{y} = 0$$

$$\Rightarrow \boxed{X^T X\theta = X^T\vec{y}} \quad \text{The normal equations}$$

$$\Downarrow$$

$$\theta^* = \left(X^T X\right)^{-1} X^T\vec{y}$$

A recap:

- LMS update rule

$$\theta_j^{t+1} = \theta_j^t + \alpha(y_n - \mathbf{x}_n^T\theta^t)x_{n,i}$$

  - Pros: on-line, low per-step cost
  - Cons: coordinate, maybe slow-converging

- Steepest descent

$$\theta^{t+1} = \theta^t + \alpha\sum_{i=1}^{n}(y_n - \mathbf{x}_n^T\theta^t)\mathbf{x}_n$$

  - Pros: fast-converging, easy to implement
  - Cons: a batch,

- Normal equations

$$\theta^* = \left(X^T X\right)^{-1} X^T\vec{y}$$

  - Pros: a single-shot algorithm! Easiest to implement.
  - Cons: need to compute pseudo-inverse $(X^TX)^{-1}$, expensive, numerical issues (e.g., matrix is singular ..)

Geometric Interpretation of LMS

- The predictions on the training data are:

$$\hat{\vec{y}} = X\theta^* = X\left(X^T X\right)^{-1} X^T \vec{y}$$

- Note that

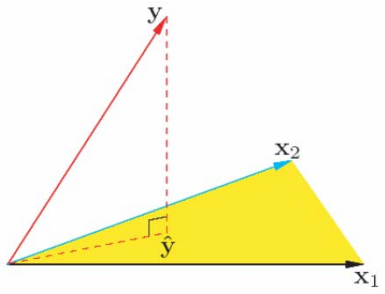
$$\hat{\vec{y}} - \vec{y} = \left(X\left(X^T X\right)^{-1} X^T - I\right)\vec{y}$$

and

$$\begin{aligned} X^T\left(\hat{\vec{y}} - \vec{y}\right) &= X^T\left(X\left(X^T X\right)^{-1} X^T - I\right)\vec{y} \\ &= \left(X^T X\left(X^T X\right)^{-1} X^T - X^T\right)\vec{y} \\ &= 0 \quad !! \end{aligned}$$

$\hat{\vec{y}}$ is the orthogonal projection of $\vec{y}$ into the space spanned by the column of X

$$\mathbf{X} = \begin{bmatrix} -- & \mathbf{x}_1 & -- \\ -- & \mathbf{x}_2 & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n & -- \end{bmatrix}$$

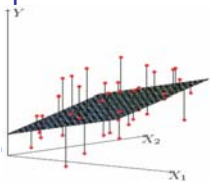


Probabilistic Interpretation of LMS

- Let us assume that the target variable and the inputs are related by the equation:

$$y_i = \theta^T \mathbf{x}_i + \varepsilon_i$$

where ε is an error term of unmodeled effects or random n



- Now assume that ε follows a Gaussian $N(0,\sigma)$, then we have:

$$p(y_i \mid x_i;\theta) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2}\right)$$

- By independence assumption:

$$L(\theta) = \prod_{i=1}^{n} p(y_i \mid x_i;\theta) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{\sum_{i=1}^{n}(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2}\right)$$

Probabilistic Interpretation of LMS, cont.

- Hence the log-likelihood is:

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

- Do you recognize the last term?

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

- Thus under independence assumption, LMS is equivalent to MLE of θ !

Multivariate Linear Regression

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

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

- 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

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

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

Multivariate Linear Regression

- Log-(conditional-) likelihood

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

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

  we have $\frac{\partial \ell}{\partial A} = -\frac{1}{2}\sum_n 2\Sigma^{-1}(y_n - Ax_n)x_n^T$

$$= \Sigma^{-1}\left(\sum_n y_n x_n^T - A\sum_n x_n x_n^T\right) \stackrel{\text{def}}{=} \Sigma^{-1}(S_{YX'} - AS_{XX'}) = 0$$

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

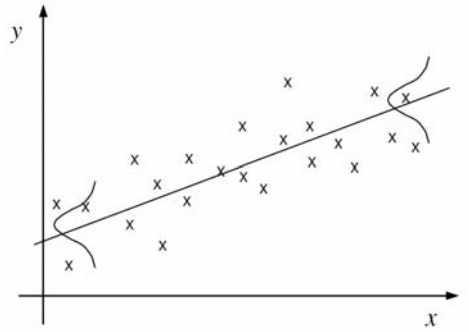
Hence $A = S_{YX'}S_{XX'}^{-1}$

1-D linear regression

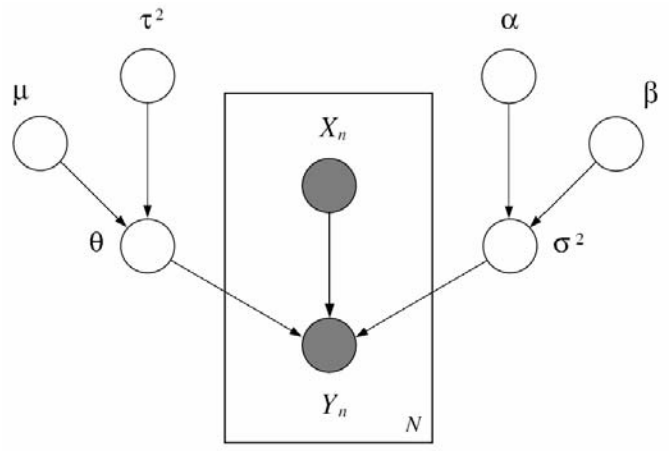
$A = S_{YX'}S_{XX'}^{-1}$

- In the special case of scalar outputs, let $A = \theta^T$, 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

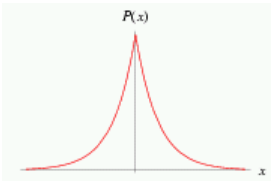


Laplace Prior and Sparsity

- The Laplace prior:

$$p(\theta_k \mid \lambda) = \frac{\lambda}{2}\exp\left(-\lambda|\theta_k|\right)$$

$$p(\theta \mid \lambda) = \frac{\lambda}{2}\exp\left(-\lambda|\theta|_1\right)$$



- The joint likelihood:

$$p(y_i, \theta \mid x_i) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2}\right) \times \frac{\lambda}{2}\exp\left(-\lambda|\theta|_1\right)$$

- The "regularized" regression cost function

$$J(\theta) = (y_i - \theta^T \mathbf{x}_i)^2 + \lambda|\theta|_1$$

L1 regularization

- The "regularized" cost:

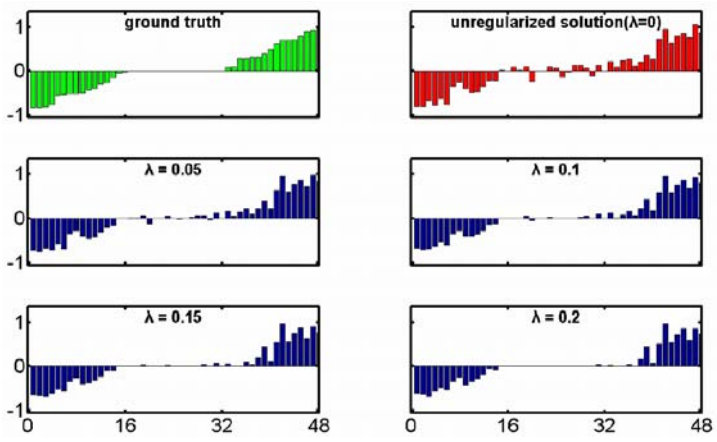
$$J(\theta) = (y_i - \theta^T \mathbf{x}_i)^2 + \lambda |\theta|_1$$

  - The regularization term penalizes all factors equally. This makes the θ *SPARSE*
  - A sparse θ means reduced complexity
  - Can be viewed as a selection of relevant/important features
  - J(θ) 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(θ) is piece-wise differentiable, → piece-wise gradient

$$p(\theta \mid \lambda) = \begin{cases} \frac{\lambda}{2} \exp\left(-\lambda(\theta_k + |\theta_{-k}|_1)\right) & \theta_k \geq 0 \\ \frac{\lambda}{2} \exp\left(-\lambda(-\theta_k + |\theta_{-k}|_1)\right) & \theta_k < 0 \end{cases}$$

  - Known as Lasso regression in Statistics

Effects of L1-Regularization



Select λ by cross-validation

L2 regularization

- Let $$p(\theta \mid \lambda) = \left(\frac{\lambda}{\pi}\right)^{N/2} \exp\left(-\lambda(\theta - 0)^T(\theta - 0)^T\right)$$
- The joint likelihood:
$$p(y_i, \theta \mid x_i) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2}\right) \times \left(\frac{\lambda}{\pi}\right)^{N/2} \exp\left(-\lambda|\theta|_2^2\right)$$
- The "regularized" regression cost function
$$J(\theta) = (y_i - \theta^T \mathbf{x}_i)^2 + \lambda|\theta|_2^2$$
  - 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: $p(y \mid x; \theta) = \mathcal{N}\left(y; \theta^T x, \sigma^2\right)$ discriminatively
  - Gradient descent
  - Normal equation
- How can we use this scheme to learning the conditional Gaussian classifier discriminatively?
  - Recall that $p(y \mid x) = \mu(x)^y (1 - \mu(x))^{1-y}$

where $$\mu(x) = \frac{1}{1 + e^{-\theta^T x}}$$

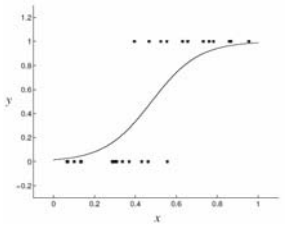
Logistic regression (sigmoid classifier)

- The condition distribution: a Bernoulli

$$p(y \mid x) = \mu(x)^y (1-\mu(x))^{1-y}$$

where μ is a logistic function

$$\mu(x) = \frac{1}{1+e^{-\theta^T x}}$$



- We can used 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!)

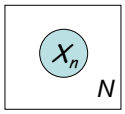
Summary

- 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 X

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



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

- Function $T(x)$ 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 $X \in \mathbb{R}^k$:

$$p(x|\mu,\Sigma) = \frac{1}{(2\pi)^{k/2}|\Sigma|^{1/2}}\exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$$
$$= \frac{1}{(2\pi)^{k/2}}\exp\left\{-\tfrac{1}{2}\mathrm{tr}\left(\Sigma^{-1}xx^T\right) + \mu^T\Sigma^{-1}x - \tfrac{1}{2}\mu^T\Sigma^{-1}\mu - \log|\Sigma|\right\}$$

Moment parameter

- Exponential family representation

Natural parameter

$$\eta = \left[\Sigma^{-1}\mu; -\tfrac{1}{2}\mathrm{vec}\left(\Sigma^{-1}\right)\right] = [\eta_1, \mathrm{vec}(\eta_2)], \quad \eta_1 = \Sigma^{-1}\mu \text{ and } \eta_2 = -\tfrac{1}{2}\Sigma^{-1}$$
$$T(x) = \left[x; \mathrm{vec}\left(xx^T\right)\right]$$
$$A(\eta) = \tfrac{1}{2}\mu^T\Sigma^{-1}\mu + \log|\Sigma| = -\tfrac{1}{2}\mathrm{tr}(\eta_2\eta_1\eta_1^T) - \tfrac{1}{2}\log(-2\eta_2)$$
$$h(x) = (2\pi)^{-k/2}$$

- Note: a k-dimensional Gaussian is a $(d+d^2)$-parameter distribution with a $(d+d^2)$-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 $x \sim \text{multi}(x \mid \pi)$,

$$p(x|\pi) = \pi_1^{x^1}\pi_2^{x^2}\cdots\pi_K^{x^K} = \exp\left\{\sum_k x^k \ln \pi_k\right\}$$
$$= \exp\left\{\sum_{k=1}^{K-1} x^k \ln \pi_k + \left(1-\sum_{k=1}^{K-1} x^K\right)\ln\left(1-\sum_{k=1}^{K-1}\pi_k\right)\right\}$$
$$= \exp\left\{\sum_{k=1}^{K-1} x^k \ln\left(\frac{\pi_k}{1-\sum_{k=1}^{K-1}\pi_k}\right) + \ln\left(1-\sum_{k=1}^{K-1}\pi_k\right)\right\}$$

- Exponential family representation

$$\eta = \left[\ln\left(\pi_k / \pi_K\right); 0\right]$$
$$T(x) = [x]$$
$$A(\eta) = -\ln\left(1-\sum_{k=1}^{K-1}\pi_k\right) = \ln\left(\sum_{k=1}^{K} e^{\eta_k}\right)$$
$$h(x) = 1$$

Why exponential family?

- Moment generating property

$$\frac{dA}{d\eta} = \frac{d}{d\eta}\log Z(\eta) = \frac{1}{Z(\eta)}\frac{d}{d\eta}Z(\eta)$$
$$= \frac{1}{Z(\eta)}\frac{d}{d\eta}\int h(x)\exp\{\eta^T T(x)\}dx$$
$$= \int T(x)\frac{h(x)\exp\{\eta^T T(x)\}}{Z(\eta)}dx$$
$$= E[T(x)]$$

$$\frac{d^2A}{d\eta^2} = \int T^2(x)\frac{h(x)\exp\{\eta^T T(x)\}}{Z(\eta)}dx - \int T(x)\frac{h(x)\exp\{\eta^T T(x)\}}{Z(\eta)}dx\frac{1}{Z(\eta)}\frac{d}{d\eta}Z(\eta)$$
$$= E[T^2(x)] - E^2[T(x)]$$
$$= Var[T(x)]$$

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.

$$\frac{dA(\eta)}{d\eta} = \text{mean}$$

$$\frac{d^2A(\eta)}{d\eta^2} = \text{variance}$$

$$\ldots$$

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

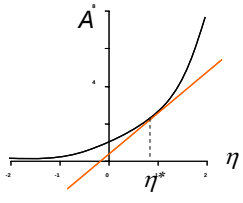
Moment vs canonical parameters

- The moment parameter μ 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

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



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

$$\eta \overset{\text{def}}{=} \psi(\mu)$$

  - A distribution in the exponential family can be parameterized not only by η – the canonical parameterization, but also by μ – the moment parameterization.

MLE for Exponential Family

- For *iid* data, the log-likelihood is

$$\ell(\eta; D) = \log \prod_n h(x_n) \exp\left\{\eta^T T(x_n) - A(\eta)\right\}$$

$$= \sum_n \log h(x_n) + \left(\eta^T \sum_n T(x_n)\right) - NA(\eta)$$

- Take derivatives and set to zero:

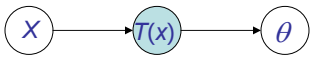
$$\frac{\partial \ell}{\partial \eta} = \sum_n T(x_n) - N \frac{\partial A(\eta)}{\partial \eta} = 0$$

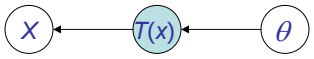
$$\Rightarrow \frac{\partial A(\eta)}{\partial \eta} = \frac{1}{N} \sum_n T(x_n)$$

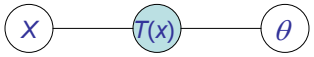
$$\hat{\mu}_{MLE} = \frac{1}{N} \sum_n T(x_n)$$

- 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 θ if there is no information in X regarding θ yeyond that in $T(x)$.
  - We can throw away X for the purpose pf inference w.r.t. θ.
  - Bayesian view 

$p(\theta \mid T(x), x) = p(\theta \mid T(x))$
  - Frequentist view 

$p(x \mid T(x), \theta) = p(x \mid T(x))$
  - The Neyman factorization theorem 

    - $T(x)$ is *sufficient* for θ if

$$p(x, T(x), \theta) = \psi_1(T(x), \theta)\psi_2(x, T(x))$$

$$\Rightarrow p(x \mid \theta) = g(T(x), \theta)h(x, T(x))$$

Examples

- Gaussian:

$$\eta = \left[\Sigma^{-1}\mu; -\tfrac{1}{2}\mathrm{vec}(\Sigma^{-1})\right]$$
$$T(x) = \left[x; \mathrm{vec}(xx^T)\right]$$
$$A(\eta) = \tfrac{1}{2}\mu^T\Sigma^{-1}\mu + \tfrac{1}{2}\log|\Sigma|$$
$$h(x) = (2\pi)^{-k/2}$$
$$\Rightarrow \mu_{MLE} = \frac{1}{N}\sum_n T_1(x_n) = \frac{1}{N}\sum_n x_n$$

- Multinomial:

$$\eta = \left[\ln\left(\pi_k / \pi_K\right); 0\right]$$
$$T(x) = [x]$$
$$A(\eta) = -\ln\left(1 - \sum_{k=1}^{K-1}\pi_k\right) = \ln\left(\sum_{k=1}^{K} e^{\eta_k}\right)$$
$$h(x) = 1$$
$$\Rightarrow \mu_{MLE} = \frac{1}{N}\sum_n x_n$$

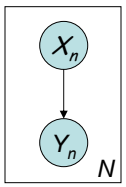
- Poisson:

$$\eta = \log\lambda$$
$$T(x) = x$$
$$A(\eta) = \lambda = e^{\eta}$$
$$h(x) = \frac{1}{x!}$$
$$\Rightarrow \mu_{MLE} = \frac{1}{N}\sum_n x_n$$

Generalized Linear Models (GLIMs)

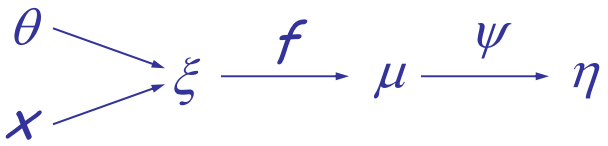
- The graphical model
  - Linear regression
  - Discriminative linear classification
  - Commonality:

    model $\mathrm{E}(Y) = \mu = f(\theta^T X)$
    - What is $p()$, 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 μ is represented as a function $f(\xi)$ of ξ, 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 μ.

GLIM, cont.



$$p(y \mid \eta) = h(x) \exp\left\{\eta^T(x) y - A(\eta)\right\}$$

$$\Rightarrow p(y \mid \eta) = h(x) \exp\left\{\tfrac{1}{\phi}\left(\eta^T(x) y - A(\eta)\right)\right\}$$

- The choice of exp family is constrained by the nature of the data y
  - Example: y is a continuous vector → multivariate Gaussian
    y is a class label → Bernoulli or multinomial
- The choice of the response function
  - Following some mild constrains, e.g., [0,1]. Positivity …
  - Canonical response function: $f = \psi^{-1}(\cdot)$
    - In this case $\theta^T x$ directly corresponds to canonical parameter η.

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

$$\frac{d\ell}{d\theta} = \sum_n \left(x_n y_n - \frac{dA(\eta_n)}{d\eta_n}\frac{d\eta_n}{d\theta}\right)$$
$$= \sum_n (y_n - \mu_n) x_n$$
$$= X^T(y - \mu)$$

This is a fixed point function because μ is a function of θ

- Online learning for canonical GLIMs
  - Stochastic gradient ascent = least mean squares (LMS) algorithm:

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

where $\mu_n^t = \left(\theta^t\right)^T x_n$ and ρ is a step size

Batch learning for canonical GLIMs

- The Hessian matrix

$$H = \frac{d^2\ell}{d\theta d\theta^T} = \frac{d}{d\theta^T}\sum_n (y_n - \mu_n)x_n = \sum_n x_n \frac{d\mu_n}{d\theta^T}$$
$$= -\sum_n x_n \frac{d\mu_n}{d\eta_n}\frac{d\eta_n}{d\theta^T}$$
$$= -\sum_n x_n \frac{d\mu_n}{d\eta_n} x_n^T \quad \text{since } \eta_n = \theta^T x_n$$
$$= -X^T W X$$

where $X = \left[x_n^T\right]$ 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 $A(\eta_n)$

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:

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

where the adjusted response is $z^t = X\theta^t + \left(W^t\right)^{-1}(y - \mu^t)$

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

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

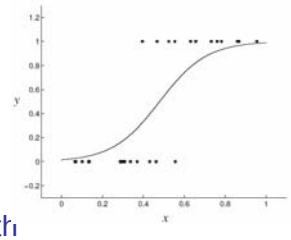
Example 1: logistic regression (sigmoid classifier)

- The condition distribution: a Bernoulli

$$p(y \mid x) = \mu(x)^y (1-\mu(x))^{1-y}$$

where μ is a logistic function

$$\mu(x) = \frac{1}{1+e^{-\eta(x)}}$$



- $p(y|x)$ 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 $\frac{d\mu}{d\eta} = \mu(1-\mu)$

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

Logistic regression: practical issues

- It is very common to use *regularized* maximum likelihood.

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

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

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

  - IRLS takes $O(Nd^3)$ 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 $O(Nd)$ per iteration, and usually works best in practice.
  - Stochastic gradient descent can also be used if N is large c.f. perceptron rule:

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

Example 2: linear regression

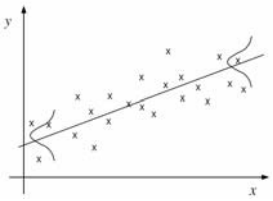
- The condition distribution: a Gaussian

$$p(y|x,\theta,\Sigma)=\frac{1}{(2\pi)^{k/2}|\Sigma|^{1/2}}\exp\left\{-\frac{1}{2}(y-\mu(x))^T\Sigma^{-1}(y-\mu(x))\right\}$$

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

where μ is a linear function

$$\mu(x)=\theta^T x=\eta(x)$$



- $p(y|x)$ is an exponential family function, with
  - mean: $E[y\mid x]=\mu=\theta^T x$
  - and canonical response function $\eta_1=\xi=\theta^T x$
- IRLS $\frac{d\mu}{d\eta}=1$, $W=I$

$$\Rightarrow\quad \theta^{t+1}=\left(X^TW^tX\right)^{-1}X^TW^tz^t$$
$$=\left(X^TX\right)^{-1}X^T\left(X\theta^t+(y-\mu^t)\right)$$
$$=\theta^t+\left(X^TX\right)^{-1}X^T(y-\mu^t)$$

Steepest descent

$$\overset{t\to\infty}{\Rightarrow}\quad \theta=(X^TX)^{-1}X^Ty$$

Normal equation