10/19 <u>Numerical optimization</u>

How to minimize (maximize) a function $f(\vec{\theta})$ over $\vec{\theta} = (\theta_1, \cdots, \theta_d) \in \mathbb{R}^d$?

1) Gradient descent (or ascent)

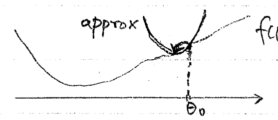
$$\vec{\theta} \leftarrow \vec{\theta} - \eta \left(\frac{\partial f}{\partial \vec{\theta}}\right)$$

scalar learning rate $\eta > 0$.

2) Newton's method

* Approximate $f(\theta)$ near $\theta = \theta_0$.

$$f(\theta) = f(\theta_0) + f'(\theta_0)(\theta - \theta_0) + \frac{1}{2} f''(\theta)(\theta - \theta_0)^2 + \cdots$$



* Minimize quadratic approximation.

$$\frac{\partial}{\partial \theta}\left[f(\theta_0) + f'(\theta_0)(\theta - \theta_0) + \frac{1}{2} f''(\theta_0)(\theta - \theta_0)^2\right] = 0$$

$$f'(\theta_0) + f''(\theta_0)(\theta - \theta_0) = 0$$

$$\theta^* = \theta_0 - \frac{f'(\theta_0)}{f''(\theta_0)}$$

* Update rule

$$\theta \longleftarrow \theta - \frac{f'(\theta)}{f''(\theta)}$$

* In d dimensions:

Define H = $d \times d$ matrix of 2nd partial derivatives

$H_{\alpha\beta} = \frac{\partial^2 f}{\partial \theta_\alpha \partial \theta_\beta}$ (symmetric) Hessian matrix

Update rule: $\vec{\theta} \longleftarrow \vec{\theta} - \underbrace{H^{-1}\left(\frac{\partial f}{\partial \vec{\theta}}\right)}_{\text{matrix-vector multiplication}}$ evaluated at current value of $\vec{\theta}$

matrix inverse of Hessian

* Pros

– no learning rate that needs to be tuned

– converges very fast (when it converges)

* Cons

– unstable if far from optimum

– often expensive to compute or invert Hessian
$O(d^2)$ $O(d^3)$

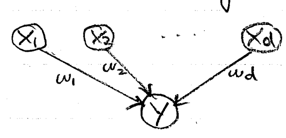
– converges only to local minimum (if it converges).

<u>Case IIb.</u> Learning in BNs with sigmoid CPTs, complete data (logistic regression)

* belief network

parents $\vec{X} \in \mathbb{R}^d$

child $Y \in \{0, 1\}$



* Sigmoid CPT

$$P(Y=1 \mid \vec{X}=\vec{x}) = \sigma(\vec{w}\cdot\vec{x}) \quad \text{with} \quad \sigma(z) = \frac{1}{1+\exp(-z)}$$

* properties of sigmoid function

$$\sigma(-z) = 1 - \sigma(z)$$

$$\sigma'(z) = \sigma(z)\,\sigma(-z)$$

* log-likelihood of training examples

$\{(\vec{x}_t, y_t)\}_{t=1}^{T}$ with $y_t \in \{0,1\}$

$$\begin{aligned}
\mathcal{L}(\vec{w}) &= \log P(\text{data}) \qquad \text{assume data is i.i.d.} \\
&= \log \prod_{t=1}^{T} P(Y=y_t \mid \vec{X}=\vec{x}_t) \\
&= \sum_{t=1}^{T} \log P(Y=y_t \mid \vec{X}=\vec{x}_t) \\
&= \sum_{t=1} \log \left[\sigma(\vec{w}\cdot\vec{x}_t)^{y_t} \cdot \sigma(-\vec{w}\cdot\vec{x}_t)^{1-y_t} \right] \\
&= \sum_t \left[y_t \log \sigma(\vec{w}\cdot\vec{x}_t) + (1-y_t)\log \sigma(-\vec{w}\cdot\vec{x}_t) \right]
\end{aligned}$$

* To maximize $\mathcal{L}(\vec{w})$

$$\begin{aligned}
0 = \frac{\partial \mathcal{L}}{\partial w_\alpha} &= \sum_t \Big[y_t \frac{1}{\sigma(\vec{w}\cdot\vec{x}_t)} \sigma(\vec{w}\cdot\vec{x}_t)\,\sigma(-\vec{w}\cdot\vec{x}_t)\, x_{\alpha t} \\
&\qquad + (1-y_t)\frac{1}{\sigma(-\vec{w}\cdot\vec{x}_t)} \sigma(-\vec{w}\cdot\vec{x}_t)\,\sigma(\vec{w}\cdot\vec{x}_t)(-x_{\alpha t}) \Big] \\
&= \sum_t \left[y_t(1-\sigma(\vec{w}\cdot\vec{x}_t))\,x_{\alpha t} - (1-y_t)\,\sigma(\vec{w}\cdot\vec{x}_t)\,x_{\alpha t} \right] \\
&= \sum_t \left[y_t - \sigma(\vec{w}\cdot\vec{x}_t) \right] x_{\alpha t} \qquad \text{for } \alpha = 1,2,\ldots,d
\end{aligned}$$

* Gradient

$$\frac{\partial \mathcal{L}}{\partial \vec{w}} = \sum_t \underbrace{(y_t - \sigma(\vec{w}\cdot\vec{x}_t))}\,\vec{x}_t$$

difference between target value $y_t \in \{0,1\}$ and model's prediction $P(Y=1 \mid \vec{x}_t)$

* Hessian

$$H_{\alpha\beta} = \frac{\partial^2 \mathcal{L}}{\partial w_\alpha \partial w_\beta} = -\sum_t \sigma(\vec{w}\cdot\vec{x}_t)\,\sigma(-\vec{w}\cdot\vec{x}_t)\,x_{\beta t}\,x_{\alpha t}.$$

$$H = \frac{\partial^2 \mathcal{L}}{\partial \vec{w}\,\partial \vec{w}^T} = -\sum_t \sigma(\vec{w}\cdot\vec{x}_t)\,\sigma(-\vec{w}\cdot\vec{x}_t)\,\vec{x}_t\,\vec{x}_t^{\,T}$$

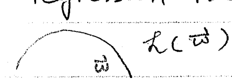
* Algorithms for ML estimation.

1) Gradient ascent : update $\vec{w} \leftarrow \vec{w} + \eta\left(\frac{\partial \mathcal{L}}{\partial \vec{w}}\right)$ suggest: $\eta = \frac{0.02}{T}$.

2) Newton's method : update $\vec{w} \leftarrow \vec{w} - H^{-1}\left(\frac{\partial \mathcal{L}}{\partial \vec{w}}\right)$

* global optimality

It can be shown that $\mathcal{L}(\vec{w})$ for logistic regression has no spurious local maxima. i.e., $\mathcal{L}(\vec{w})$ is convex.



<u>Case III</u> fixed DAG, discrete nodes, "lookup" CPTs, <u>incomplete</u> data

* Variables in BN : H = hidden variables $\}$ may vary from one example to next.

V = visible variables

$X = H \cup V$

* Data set: Assume T incomplete / partial examples drawn i.i.d from $P(X)$.

#	X_1	X_2	...	X_n
1	5	9		?
2	?	4		2
3	?	?		1
⋮				
T	1	2		2

* log-likelihood

$$\mathcal{L} = \log P(\text{DATA}) \qquad \text{assume i.i.d.}$$

$$= \log \prod_{t=1}^{T} P(V^{(t)}) \quad \text{observed nodes in BN for } t^{th} \text{ example.}$$

$$= \sum_t \log P(V^{(t)})$$

$$= \sum_t \log \sum_{h^{(t)}} P(V^{(t)}, h^{(t)}) \quad \text{joint distribution}$$

$$= \sum_t \log \sum_{h^{(t)}} \prod_{i=1}^{n} P(X_i \mid pa(X_i)) \Big|_{\substack{V=V^{(t)} \\ H=h^{(t)}}}$$

Before for complete data:

CPTs decoupled $\Rightarrow$ many independent optimizations

Now: all CPTs are coupled.

How to optimize?

* $\boxed{\text{Expectation} - \text{Maximization (EM)}}$

- alternative to gradient ascent or Newton's method.

* Intuition – by analogy

· ML estimates for complete data

$$I(a,b) = \begin{cases} 1 & \text{if } a=b \\ 0 & \text{if } a\neq b \end{cases} \quad \text{indicator function.}$$

$$P_{ML}(X_i = x \mid pa(X_i) = \pi) = \frac{\text{count}(X_i = x, pa_i = \pi)}{\text{count}(pa_i = \pi)} = \frac{\sum_t I(X_i^{(t)}, x)\, I(pa_i^{(t)}, \pi)}{\sum_t I(pa_i^{(t)}, \pi)}$$

· For incomplete data, we must "fill in" missing values:

$$P(X_i = x \mid pa_i = \pi) \longleftarrow \frac{\sum_t P(X_i = x, pa_i = \pi \mid V^{(t)})}{\sum_t P(pa_i = \pi \mid V^{(t)})}$$

RHS reduces to previous formula for complete data.

Intuition: expected statistics under $P(H|V)$ substitute for observed statistics in complete data case.