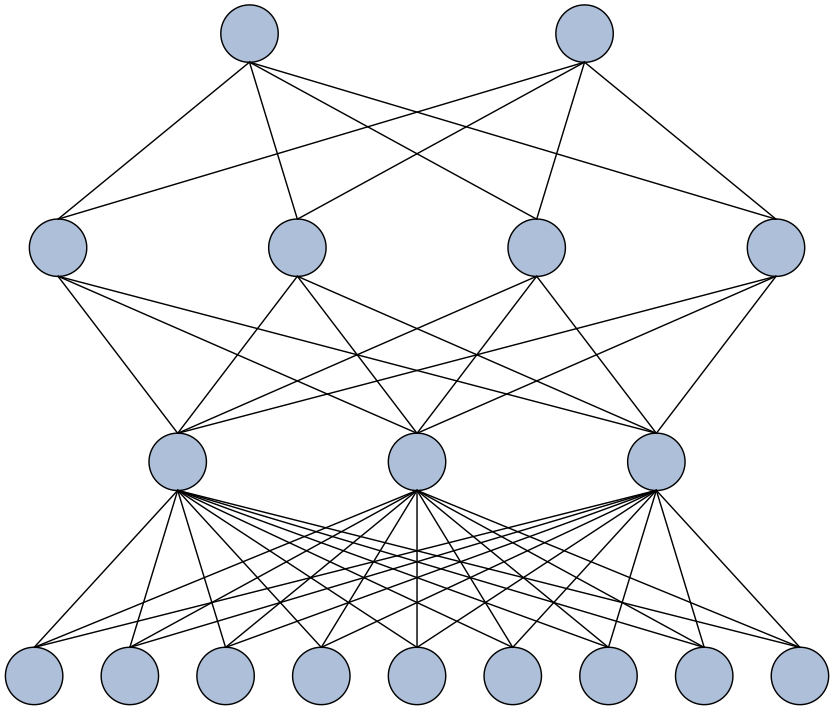
Lecture 4
Towards Deep Learning

(January 30, 2015)

Mu Zhu
University of Waterloo

Deep Network



Boltzmann Distribution

- probability distribution for a complex system

$$p(\boldsymbol{x}) = \frac{1}{Z} e^{f(\boldsymbol{x};\boldsymbol{\theta})} \quad \text{with} \quad Z \equiv \sum_{\boldsymbol{x}} e^{f(\boldsymbol{x};\boldsymbol{\theta})} \quad \left[\text{or} \quad \int e^{f(\boldsymbol{x};\boldsymbol{\theta})} d\boldsymbol{x} \right]$$

- often,

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \frac{-u(\boldsymbol{x};\boldsymbol{\vartheta})}{kT},$$

  where

$$\begin{aligned} u(\boldsymbol{x};\boldsymbol{\vartheta}) &= \text{energy function;} \\ k &= \text{Boltzmann constant;} \\ T &= \text{thermodynamic temperature} \end{aligned}$$

- e.g., lattice of particles, protein molecule

Boltzmann Distribution

$$\log[p(\boldsymbol{x};\boldsymbol{\theta})] = f(\boldsymbol{x};\boldsymbol{\theta}) - \log\left[\sum_{\boldsymbol{x}} e^{f(\boldsymbol{x};\boldsymbol{\theta})}\right]$$

$$\begin{aligned}
\frac{d}{d\boldsymbol{\theta}}\log[p(\boldsymbol{x};\boldsymbol{\theta})] &= \frac{d}{d\boldsymbol{\theta}}f(\boldsymbol{x};\boldsymbol{\theta}) - \frac{1}{\sum_{\boldsymbol{x}} e^{f(\boldsymbol{x};\boldsymbol{\theta})}}\left[\sum_{\boldsymbol{x}} e^{f(\boldsymbol{x};\boldsymbol{\theta})}\cdot\frac{d}{d\boldsymbol{\theta}}f(\boldsymbol{x};\boldsymbol{\theta})\right] \\
&= \frac{d}{d\boldsymbol{\theta}}f(\boldsymbol{x};\boldsymbol{\theta}) - \sum_{\boldsymbol{x}}\frac{e^{f(\boldsymbol{x};\boldsymbol{\theta})}}{Z}\cdot\frac{d}{d\boldsymbol{\theta}}f(\boldsymbol{x};\boldsymbol{\theta}) \\
&= \frac{d}{d\boldsymbol{\theta}}f(\boldsymbol{x};\boldsymbol{\theta}) - \sum_{\boldsymbol{x}} p(\boldsymbol{x})\cdot\frac{d}{d\boldsymbol{\theta}}f(\boldsymbol{x};\boldsymbol{\theta}) \\
&= \frac{d}{d\boldsymbol{\theta}}f(\boldsymbol{x};\boldsymbol{\theta}) - \mathbb{E}\left[\frac{d}{d\boldsymbol{\theta}}f(\boldsymbol{x};\boldsymbol{\theta})\right]
\end{aligned}$$

Boltzmann Distribution

- given

$$\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_i \overset{iid}{\sim} p(\boldsymbol{x}; \boldsymbol{\theta})$$

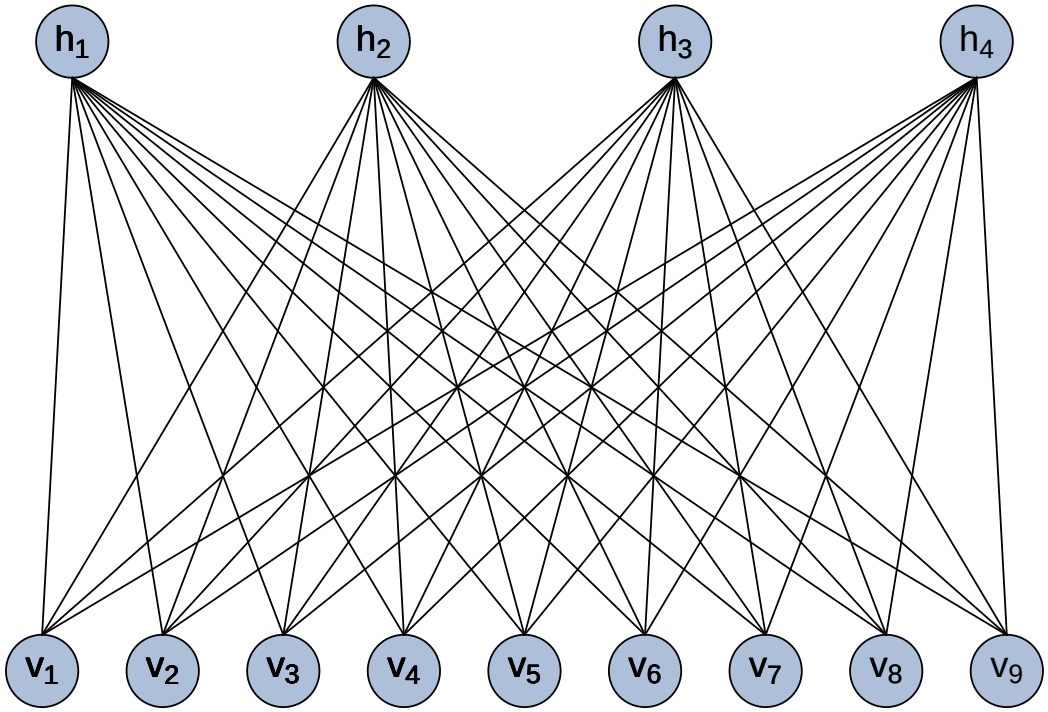
- log-likelihood is

$$\ell(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \log[p(\boldsymbol{x}_i; \boldsymbol{\theta})]$$

- its first derivative is

$$\begin{aligned} \frac{d}{d\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) &= \frac{1}{n} \sum_{i=1}^{n} \left\{ \frac{d}{d\boldsymbol{\theta}} f(\boldsymbol{x}_i; \boldsymbol{\theta}) - \mathbb{E}\left[\frac{d}{d\boldsymbol{\theta}} f(\boldsymbol{x}_i; \boldsymbol{\theta}) \right] \right\} \\ &\equiv \widehat{\mathbb{E}}\left[\frac{d}{d\boldsymbol{\theta}} f(\boldsymbol{x}; \boldsymbol{\theta}) \right] - \mathbb{E}\left[\frac{d}{d\boldsymbol{\theta}} f(\boldsymbol{x}; \boldsymbol{\theta}) \right] \end{aligned}$$

Restricted Boltzmann Machine



bottom nodes $\boldsymbol{v} = (v_1, v_2, ...)^{\mathrm{T}}$; top nodes $\boldsymbol{h} = (h_1, h_2, ...)^{\mathrm{T}}$

Restricted Boltzmann Machine

- bottom nodes $\boldsymbol{v} = (v_1, v_2, ...)^{\mathrm{T}}$
- top nodes $\boldsymbol{h} = (h_1, h_2, ...)^{\mathrm{T}}$
- Boltzmann distribution

$$p(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = \frac{1}{Z} e^{f(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})}$$

with

$$\begin{aligned} f(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) &= \boldsymbol{h}^{\mathrm{T}} \boldsymbol{W} \boldsymbol{v} + \boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{h} + \boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{v} \\ &= \sum_t h_t \boldsymbol{w}_{t\cdot}^{\mathrm{T}} \boldsymbol{v} + \sum_t \alpha_t h_t + \boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{v} \end{aligned}$$

i.e., $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$

Restricted Boltzmann Machine

- if just one binary top node $h \in \{0, 1\}$
- get

$$p(\boldsymbol{v}, h = 1) = \frac{1}{Z} e^{\boldsymbol{w}^{\mathrm{T}} \boldsymbol{v} + \alpha + \boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{v}}, \quad p(\boldsymbol{v}, h = 0) = \frac{1}{Z} e^{\boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{v}}$$

- so

$$\frac{p(\boldsymbol{v}, h = 1)}{p(\boldsymbol{v}, h = 0)} = \frac{\mathbb{P}(h = 1|\boldsymbol{v}) f(\boldsymbol{v})}{\mathbb{P}(h = 0|\boldsymbol{v}) f(\boldsymbol{v})}$$

$$\Rightarrow \log \frac{\mathbb{P}(h = 1|\boldsymbol{v})}{\mathbb{P}(h = 0|\boldsymbol{v})} = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{v} + \alpha$$

- hence, model for $h|\boldsymbol{v}$ is usual logistic regression

Restricted Boltzmann Machine

- if more than one binary top nodes $h_1, h_2, ... \in \{0, 1\}$
- get

$$f(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = h_t \boldsymbol{w}_{t\cdot}^{\mathrm{T}} \boldsymbol{v} + \alpha_t h_t + \sum_{s \neq t} h_s \boldsymbol{w}_{s\cdot}^{\mathrm{T}} \boldsymbol{v} + \sum_{s \neq t} \alpha_s h_s + \boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{v}$$

- so

$$\log \frac{\mathbb{P}(h_t = 1 | \boldsymbol{v}, \boldsymbol{h}_{-t})}{\mathbb{P}(h_t = 0 | \boldsymbol{v}, \boldsymbol{h}_{-t})} = \boldsymbol{w}_{t\cdot}^{\mathrm{T}} \boldsymbol{v} + \alpha_t$$

- in general, model for $h_t | \boldsymbol{v}, \boldsymbol{h}_{-t}$ is usual logistic regression
- notice conditional independence between h_t and $\boldsymbol{h}_{-t}$ given $\boldsymbol{v}$

Fitting RBM

- given $(\boldsymbol{v}_i, \boldsymbol{h}_i)$, $i = 1, 2, ..., n$, MLE by gradient ascent,

$$\boldsymbol{W}_{new} = \boldsymbol{W}_{old} + \varepsilon \left[\frac{d\ell}{d\boldsymbol{W}} \right]_{\boldsymbol{\theta}_{old}},$$

  and likewise for $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$

- gradients:

$$f(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = \boldsymbol{h}^{\mathrm{T}} \boldsymbol{W} \boldsymbol{v} + \boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{h} + \boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{v}$$

$$\Downarrow$$

$$\frac{d\ell}{d\boldsymbol{W}} = \widehat{\mathbb{E}} \left[\boldsymbol{h}\boldsymbol{v}^{\mathrm{T}} \right] - \mathbb{E} \left[\boldsymbol{h}\boldsymbol{v}^{\mathrm{T}} \right]$$

$$\frac{d\ell}{d\boldsymbol{\alpha}} = \widehat{\mathbb{E}} \left[\boldsymbol{h} \right] - \mathbb{E} \left[\boldsymbol{h} \right], \quad \frac{d\ell}{d\boldsymbol{\beta}} = \widehat{\mathbb{E}} \left[\boldsymbol{v} \right] - \mathbb{E} \left[\boldsymbol{v} \right]$$

Fitting RBM

- given $\{(\boldsymbol{v}_i, \boldsymbol{h}_i)\}_{i=1}^n$, $\widehat{\mathbb{E}}\left[\boldsymbol{h}\boldsymbol{v}^{\mathrm{T}}\right], \widehat{\mathbb{E}}\left[\boldsymbol{h}\right], \widehat{\mathbb{E}}\left[\boldsymbol{v}\right]$ are easy to compute
  - just take empirical averages [definition of $\widehat{\mathbb{E}}(\cdot)$]
- but $\mathbb{E}\left[\boldsymbol{h}\boldsymbol{v}^{\mathrm{T}}\right], \mathbb{E}\left[\boldsymbol{h}\right], \mathbb{E}\left[\boldsymbol{v}\right]$ are hard to compute
  - estimate by drawing an MCMC sample from $p(\boldsymbol{v}, \boldsymbol{h})$
  - in particular, do Gibbs Sampling for just a few iterations

Gibbs Sampling Repeat

- given $\boldsymbol{v}$, sample $\boldsymbol{h}$ from the conditional distribution of $\boldsymbol{h}|\boldsymbol{v}$;
- given $\boldsymbol{h}$, sample $\boldsymbol{v}$ from the conditional distribution of $\boldsymbol{v}|\boldsymbol{h}$;

until convergence ("burn-in").

Gibbs Sampling

- recall: model for $h_t|\boldsymbol{v}, \boldsymbol{h}_{-t}$ is usual logistic regression, so

$$h_t|\boldsymbol{v}, \boldsymbol{h}_{-t} \sim \text{Bernoulli}\left[\sigma\left(\alpha_t + \boldsymbol{w}_{t\cdot}^{\text{T}}\boldsymbol{v}\right)\right]$$

- if $v_1, v_2, ...$ also binary, then symmetry gives

$$v_b|\boldsymbol{h}, \boldsymbol{v}_{-b} \sim \text{Bernoulli}\left[\sigma\left(\beta_b + \boldsymbol{h}^{\text{T}}\boldsymbol{w}_{\cdot b}\right)\right]$$

- Gibbs sampling amounts to back-and-forth coin flips

- note: $\sigma(\cdot)$ above denotes the sigmoid function

Towards Deep Learning

- stack many RBMs on top of each other
- top nodes for layer ℓ becomes bottom nodes for layer $\ell+1$, i.e.,

$$\boldsymbol{v}^{(\ell+1)} = \boldsymbol{h}^{(\ell)}$$

- fit the whole thing layer by layer
- note intermediate layers are latent (hidden, not visible)
- so for things like $\widehat{\mathbb{E}}(\boldsymbol{h})$, can use $\widehat{\mathbb{E}}(\widetilde{\boldsymbol{h}})$ where $\widetilde{\boldsymbol{h}}_i \equiv \mathbb{E}(\boldsymbol{h}|\boldsymbol{v}_i)$

$$\Uparrow$$

kind of an EM procedure

Towards Deep Learning

- **repeat**
  - with current parameters $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{W}\}$
  - (a) estimate $\widetilde{\boldsymbol{h}}_i \equiv \mathbb{E}(\boldsymbol{h}|\boldsymbol{v}_i) \Rightarrow \widehat{\mathbb{E}}(\cdot)$
  - (b) draw $\boldsymbol{h}_i \sim p(\boldsymbol{h}|\boldsymbol{v}_i)$ and $\boldsymbol{v}_i \sim p(\boldsymbol{v}|\boldsymbol{h}_i)$ [or repeat] $\Rightarrow \mathbb{E}(\cdot)$
  - move along the gradient (a crude estimate of it)

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \varepsilon \left[\widehat{\mathbb{E}}(\cdot) - \mathbb{E}(\cdot)\right]$$

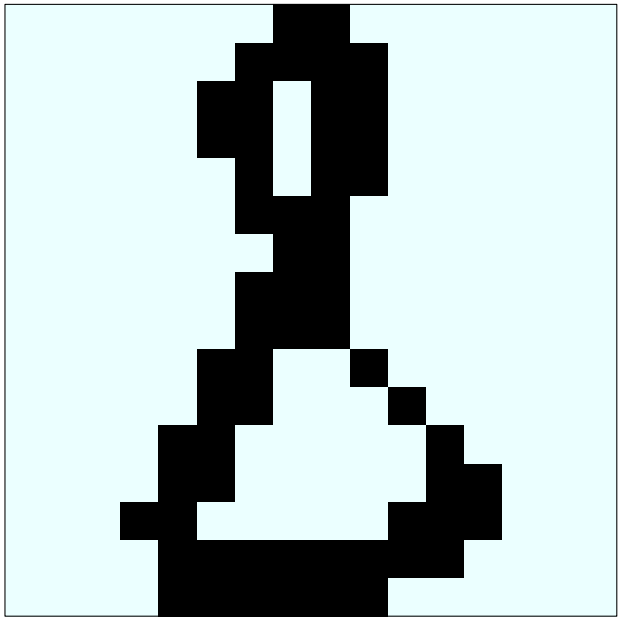
  until some stopping criterion

- proceed to next layer

On $v_1, v_2, \ldots$ Being Binary

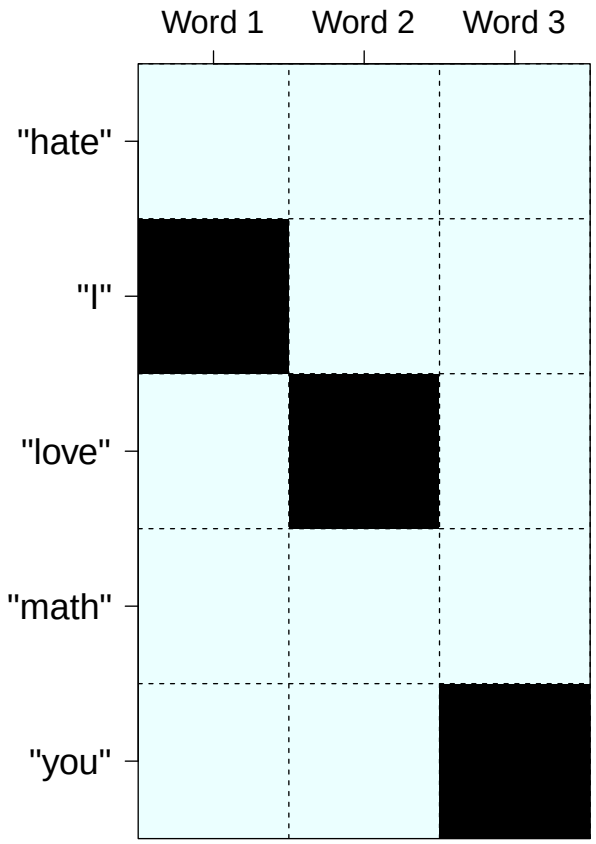
- not as restrictive as it appears
- can already handle image data and text data
  - to some extent [examples next two slides]
- can generalize to other inputs
  - make some changes to the model $f(\boldsymbol{v}, \boldsymbol{h})$
  - $p(\boldsymbol{v}|\boldsymbol{h})$ no longer logistic model [obviously]
  - ideally, want $p(\boldsymbol{v}|\boldsymbol{h})$ "nice" to sample from

Example: Image Data



each $v_1, v_2, ...$ a binary pixel

Example: Text Data



each $v_1, v_2, ...$ an indicator for a particular word

The Gaussian-Bernoulli RBM

- $v_1, v_2, ... \in \{0, 1\}$:

$$\begin{aligned} f(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) &= \boldsymbol{h}^{\mathrm{T}} \boldsymbol{W} \boldsymbol{v} + \boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{h} + \boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{v} \\ &= \sum_b \left[\boldsymbol{h}^{\mathrm{T}} \boldsymbol{w}_{\cdot b}\right] v_b + \boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{h} + \sum_b \beta_b v_b \end{aligned}$$

- $v_1, v_2, ... \in \mathbb{R}$:

$$f(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = \sum_b \left[\boldsymbol{h}^{\mathrm{T}} \boldsymbol{w}_{\cdot b}\right] \left[\frac{v_b}{\tau_b}\right] + \boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{h} - \sum_b \frac{(v_b - \beta_b)^2}{2\tau_b^2}$$

Exercise Let $\widetilde{\boldsymbol{v}} = (v_1/\tau_1, v_2/\tau_2, ...)^{\mathrm{T}}$. Show that

(a) $p(h_t | \boldsymbol{v}, \boldsymbol{h}_{-t}) \sim \text{Bernoulli}[\sigma(\alpha_t + \boldsymbol{w}_{t\cdot}^{\mathrm{T}} \widetilde{\boldsymbol{v}})]$; (easy)

(b) $p(v_b | \boldsymbol{h}, \boldsymbol{v}_{-b}) \sim \text{N}(\beta_b + \tau_b(\boldsymbol{h}^{\mathrm{T}} \boldsymbol{w}_{\cdot b}), \tau_b^2)$. (slightly harder)

Iterative Optimization

- consider an iterative rule, $x_{t+1} = m(x_t)$, for minimizing $f(x)$
- Newton's method:

$$m(x_t) = x_t - \frac{f'(x_t)}{f''(x_t)}$$

- gradient descent:

$$m(x_t) = x_t - f'(x_t)$$

- multivariate case ... same idea, with gradient and Hessian, i.e.,

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \varepsilon \boldsymbol{H}_t^{-1} \boldsymbol{g}_t, \quad \boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \varepsilon \boldsymbol{g}_t;$$

will explain extra ε later

Iterative Optimization

- suppose iteration converges to a local minimum x^*
- must have $m(x^*) = x^*$ [i.e., a fixed point]
- then, in neighborhood near x^*,

$$\begin{aligned} e_{t+1} &\equiv x_{t+1} - x^* = m(x_t) - m(x^*) \\ &= \left[m(x^*) + m'(x^*)(x_t - x^*) + \frac{m''(\xi)}{2}(x_t - x^*)^2 \right] - m(x^*) \\ &= m'(x^*)e_t + \frac{m''(\xi)}{2}{e_t}^2 \end{aligned}$$

- thus, a "basic requirement" is $|m'(x^*)| < 1$ (and nearby x^*)

Gradient Descent

$$m(x_t) = x_t - f'(x_t) \quad \Rightarrow \quad m'(x^*) = 1 - f''(x^*)$$

$$x^* \quad \text{local mimimum} \quad \Rightarrow \quad f''(x^*) > 0$$

$$\text{so, need} \quad f''(x^*) < 1 \quad (\text{and nearby } x^*)$$

$$\text{ensure by letting} \quad m(x_t) = x_t - \varepsilon f'(x_t)$$

Newton's Method

$$m(x_t) = x_t - \frac{f'(x_t)}{f''(x_t)}$$

$$m'(x^*) = 1 - \frac{f''(x^*)f''(x^*) - f'''(x^*)f'(x^*)}{[f''(x^*)]^2} = \frac{f'''(x^*)f'(x^*)}{[f''(x^*)]^2} = 0$$

get $\quad e_{t+1} \sim O(e_t^2), \quad$ much faster local convergence

but second derivative $\boldsymbol{H}_{d\times d}$ is not easy for large d

Quasi-Newton

- use local curvature information $\Rightarrow$ fast local convergence
- want to "do some of this" without computing $\boldsymbol{H}_t$
- key idea: construct a sequence $\boldsymbol{B}_t$ to "mimic" $\boldsymbol{H}_t$
- example [symmetric rank-1 (SR1)]: construct $\boldsymbol{B}_{t+1}$ so that
  - (a) $\boldsymbol{g}_{t+1} = \boldsymbol{g}_t + \boldsymbol{B}_{t+1}(\boldsymbol{x}_{t+1} - \boldsymbol{x}_t)$ [$\boldsymbol{B}_{t+1}$ "like a Hessian"]
  - (b) $\boldsymbol{B}_{t+1} = \boldsymbol{B}_t + \boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ [update "just a little"]
  - (c) $\boldsymbol{u} \propto \boldsymbol{v}$ [ensures symmetry]
- $\exists$ many variations ...

Some Details of SR1

$$\left(\boldsymbol{B}_t + \boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}\right) \underbrace{\left(\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\right)}_{\Delta \boldsymbol{x}_t} = \underbrace{\boldsymbol{g}_{t+1} - \boldsymbol{g}_t}_{\Delta \boldsymbol{g}_t}$$

$$\boldsymbol{u} \underbrace{\left[\boldsymbol{v}^{\mathrm{T}}(\Delta \boldsymbol{x}_t)\right]}_{scalar} = (\Delta \boldsymbol{g}_t) - \boldsymbol{B}_t(\Delta \boldsymbol{x}_t) \quad \Rightarrow \quad \boldsymbol{u} = \frac{(\Delta \boldsymbol{g}_t) - \boldsymbol{B}_t(\Delta \boldsymbol{x}_t)}{\boldsymbol{v}^{\mathrm{T}}(\Delta \boldsymbol{x}_t)}$$

taking $\boldsymbol{u} = \boldsymbol{v} = \gamma\left[(\Delta \boldsymbol{g}_t) - \boldsymbol{B}_t(\Delta \boldsymbol{x}_t)\right]$ leads to

$$\boldsymbol{B}_{t+1} = \boldsymbol{B}_t + \frac{\left[(\Delta \boldsymbol{g}_t) - \boldsymbol{B}_t(\Delta \boldsymbol{x}_t)\right]\left[(\Delta \boldsymbol{g}_t) - \boldsymbol{B}_t(\Delta \boldsymbol{x}_t)\right]^{\mathrm{T}}}{\left[(\Delta \boldsymbol{g}_t) - \boldsymbol{B}_t(\Delta \boldsymbol{x}_t)\right]^{\mathrm{T}}(\Delta \boldsymbol{x}_t)}$$

Summary

- key ideas:
  - RBMs (building block for deep learning)
  - gradient descent; Gibbs sampling
  - local convergence behavior (gradient *vs* Newton)
- specific techniques:
  - quasi-Newton (SR1)
  - Gaussian-Bernoulli RBM

Next ...

- a short, 10-minute break
- lecture by Dr. R. Grosse on some current work about RBMs