Lecture 8 - scribbles

Friday, September 29, 2017
13:23

today: • brief recap on optimization
• Fischer Maden for classification + math tricks

Numerical optimization

want to minimize $f(w)$: $\min_{\text{s.t. } w \in \mathbb{R}^d} f(w)$

1) gradient descent (1st order method)

start at w_0

$$w_{t+1} = w_t - \gamma_t \nabla f(w_t)$$

γ_t: step-size

step-size rules:

a) constant step-size $\gamma_t = \frac{1}{L}$ ← Lipschitz constant of ∇f

b) decreasing step-size rule: $\gamma_t = \frac{C}{t}$ ← constant

usually want $\sum_t \gamma_t = \infty$ $\sum_t \gamma_t^2 < \infty$

c) choose γ_t by "line-search": $\min_{\gamma \in \mathbb{R}} f(w_t + \gamma d_t)$ ← direction for update (e.g. $-\nabla f(w_t)$)

↳ costly in general,

instead do approximate search

e.g. Armijo line search

(see Boyd's book) book link

2) Newton's method (2nd order method)

Hessian $[H(w_t)]_{ij} = \frac{\partial^2 f(w)}{\partial w_i \partial w_j}$

2) Newton's method (2nd order method)

Hessian $[H(w_t)]_{ij} = \frac{\partial^2 f(w)}{\partial w_i \partial w_j}$

motivation: minimizing a quadratic approximation:

Taylor expansion: $f(w) = f(w_t) + \nabla f(w_t)^T (w - w_t) + \frac{1}{2}(w - w_t)^T H(w_t)(w - w_t) + O(\|w - w_t\|^3)$

Taylor's remainder

$= Q_f(w) + O(\|w - w_t\|^3)$

quadratic model approximation

want $\nabla_w Q_f(w) = 0$

$= \nabla_w f(w_t) + H(w_t)(w - w_t)$

$\Rightarrow \quad (w - w_t) = -H^{-1} \nabla f(w_t)$

$\Rightarrow \quad \boxed{w_{t+1} = w_t - H^{-1}(w_t) \nabla f(w_t)}$ Newton's update

inverse Hessian → $O(d^3)$ time to compute in general

$O(d^2)$ space

damped Newton: you add a step-size to stabilize Newton's method

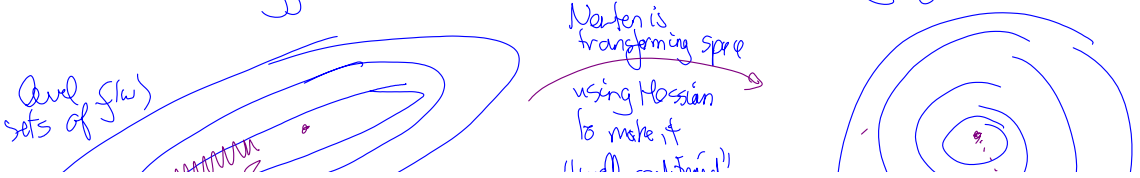
$w_{t+1} = w_t - \gamma H^{-1}(w_t) \nabla f(w_t)$

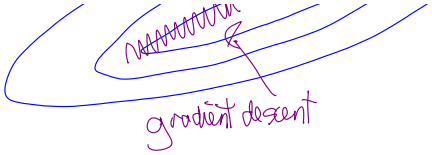
step-size

why Newton's method?

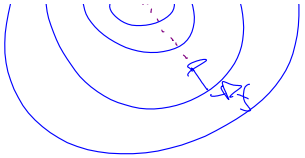
- much faster convergence in # of iterations vs gradient descent
- affine invariant → i.e. invariant to rescaling of variables

but - iterations are costly $O(d^3)$ time $O(d^2)$ memory





"well-conditioned"



Newton's method for logistic regression: IRLS

recall for log. reg., $\nabla \ell(w) = \sum_{i=1}^{n} x_i [y_i - \sigma(w^T x_i)]$

$$H(\ell(w)) = -\sum_{i=1}^{n} x_i x_i^T \left(\sigma(w^T x_i)\,\sigma(-w^T x_i)\right)$$

$$v^T H v = -\sum_{i=1}^{n} \underbrace{(v^T x_i)(x_i^T v)}_{(v^T x_i)^2 \geq 0} \underbrace{\left(\sigma(-)\,\sigma(--)\right)}_{\geq 0}$$

ie. $v^T H v \leq 0 \quad \forall v \in \mathbb{R}^q$

$\Rightarrow H \leq 0$

notation: let $\mu_i \triangleq \sigma(w^T x_i) \quad (\in]0,1[)$

recall $X = \begin{pmatrix} -x_1^T- \\ \vdots \end{pmatrix}$

Let's look at minimizing negative log-likelihood

$$\nabla \ell(w) : \sum_{i=1}^{n} x_i [\mu_i - y_i] = X^T(\vec{\mu} - \vec{y})$$

$$\text{Hessian} = \sum_{i=1}^{n} x_i x_i^T \mu_i(1-\mu_i) = X^T D X \quad \text{where } D_{ii} = \mu_i(1-\mu_i)$$

[note: D depends on w]

Newton's update: $w_{t+1} = w_t - (X^T D_t X)^{-1} X^T(\vec{\mu}_t - \vec{y})$

$$= (X^T D_t X)^{-1}\left[(X^T D_t X) w_t + X^T(\vec{y} - \vec{\mu}_t)\right]$$

$$= (X^T D_t X)^{-1}[X^T D_t z_t]$$

where $z_t \triangleq X w_t + D_t^{-1}(\vec{y} - \vec{\mu}_t)$

this is solution to "weighted least square problem"

$$\min_w \|D^{1/2}(z_t - Xw)\|^2 \quad \text{new target}$$

$$(z_t)_i = x_i^T w_t + \frac{y_i - \mu_i(w_t)}{\mu_i(1-\mu_i)}$$

$$\sum_{i=1}^n \frac{(z_i - x_i^T w)^2}{d_{ii}^{-1}}$$

compare with Gaussian noise model for least-square

$$\sum_i \frac{(y_i - x_i^T w)^2}{2\sigma^2}$$

data-dependent noise

Newton's method for logistic regression = "iterative reweighted least square (IRLS)

Big data logistic regression

- cannot do $O(d^2)$ or $O(d^3)$ operations $\Rightarrow$ first-order methods
- if n is huge, you cannot use batch method $\nabla f = \frac{1}{n}\sum_{i=1}^n \nabla f_i(w)$ every data point $O(nd)$ time

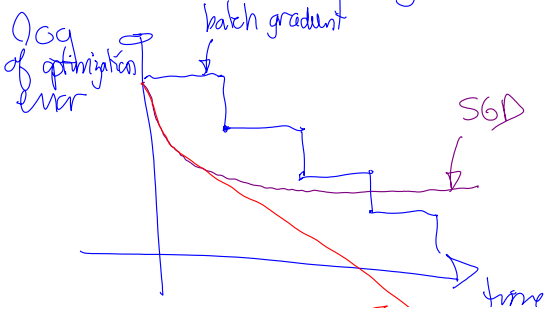
(suppose $f(w) = \frac{1}{n}\sum_{i=1}^n f_i(w)$) function depends only on i^{th} example

instead you use "incremental gradient methods"

e.g. stochastic gradient descent (SGD): $w_{t+1} = w_t - \alpha_t \nabla f_{i_t}(w_t)$ $O(d)$ where i_t is picked randomly

SGD $\rightarrow$ cheap updates, but slower convergence per iteration

batch-gradient $\rightarrow$ expensive " , but fast " " "



$$w_{t+1} = w_t - \alpha_t \frac{1}{n}\sum_{i=1}^n \nabla f_i(w_t)$$

SAG: stochastic average gradient [2012]

memory

$$w_{t+1} = w_t - \alpha \frac{1}{n}\sum_{i=1}^n v_i$$

SAGA

where $v_i^0 = \nabla f_i(w_0)$

at each t, update only one $v_{i_t}^t = \nabla f_{i_t}(w_t)$

SAGA: $w_{t+1} = w_t - \gamma \left(\nabla f_{i_t}(w_t) + \frac{1}{n} \sum_{j=1}^{n} v_j^t - v_{i_t}^t \right)$

[2014] paper

variance reducing correction

(default method for logistic regression in Scikit-learn)

---

generative model for classification: (Fisher) linear discriminant analysis

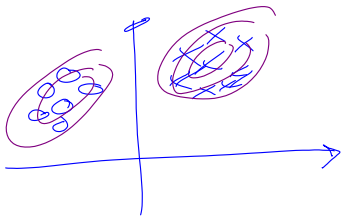
for classification $Y \in \{0,1\}$, $X \in \mathbb{R}^d$

generative approach $p(x,y;\theta) = \underbrace{p(x|y;\theta)}_{\text{class conditionals}} p(y;\theta)$

vs. conditional approach $p(y|x;\theta)$

for Fisher model, we assume $p(x|y;\theta) = N(x|\mu_y, \Sigma)$ (same for both classes)

μ_0, μ_1



$\theta = (\mu_0, \mu_1, \Sigma, \pi)$

Σ: $d \times d$ shared for 2 classes

π: prior $p(y=1)$

can then show that $p(y|x;\theta) = \sigma(w^T x)$

where w is function of $(\mu_0, \mu_1, \Sigma, \pi)$

[if you use Σ_0 & Σ_1, get "quadratic discriminant analysis" (QDA) → see Prob. 2]

[if you use $\Sigma_0 \neq \Sigma_1$, get "quadratic discriminant analysis" (QDA) → see Rmk. 2]

i.e. $\sigma(w^T \varphi(x))$ where $\varphi(x)$ is quadratic in x

④ generative approach: do joint ML i.e. $\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmax}} \sum_i \log p(x_i, y_i; \theta)$

sidenote: MLE for multivariate Gaussian

$X_i \sim N(\mu, \Sigma)$ $\quad \mu \in \mathbb{R}^d$

$\Sigma \in \mathbb{R}^{d \times d}$, Σ is symmetric $\Sigma \succ 0$

$$p(x) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu) \right)$$

$\operatorname{tr}(AB) = \operatorname{tr}(BA)$

$(x-\mu)^T \Sigma^{-1} (x-\mu)$ is a scalar, so

$$= \operatorname{tr}\left((x-\mu)^T \Sigma^{-1} (x-\mu) \right)$$

$$= \operatorname{tr}\left(\Sigma^{-1} (x-\mu)(x-\mu)^T \right) = \left\langle \Sigma^{-1}, (x-\mu)(x-\mu)^T \right\rangle$$

$\langle A, B \rangle \triangleq \sum_{ij} A_{ij} B_{ij} = \operatorname{tr}(A^T B)$

$\theta = (\mu, \Sigma)$

log-likelihood: $\sum_{i=1}^n \log p(x_i; \theta) = \text{cst.} - \frac{n}{2} \log|\Sigma| - \frac{n}{2} \left\langle \Sigma^{-1}, \frac{1}{n} \sum_{i=1}^n (x_i-\mu)(x_i-\mu)^T \right\rangle$

$\sum_{i=1}^n (x_i-\mu)(x_i-\mu)^T = \tilde{\Sigma}$

vector derivative review:

suppose $f: \mathbb{R}^m \to \mathbb{R}^n$

f is differentiable at x_0 iff $\exists$ a linear operator $df_{x_0}: \mathbb{R}^m \to \mathbb{R}^n$ s.t. $\forall \Delta \in \mathbb{R}^m$, $f(x_0 + \Delta) - f(x_0) = df_{x_0}(\Delta) + o(\|\Delta\|)$

"little oh"

$o(\|\Delta\|)$ means some function $h(\|\Delta\|)$ s.t. $\lim_{\|\Delta\| \to 0} \frac{h(\|\Delta\|)}{\|\Delta\|} \to 0$

$f(x) \in \mathbb{R}^n$

$\Delta \in \mathbb{R}^m$ i.e. $df_{x_0} : \mathbb{R}^m \to \mathbb{R}^n$

is linear i.e. $df_{x_0}(\Delta_1 + b\Delta_2) = df_{x_0}(\Delta_1) + b\, df_{x_0}(\Delta_2)$

can represent as a $n \times m$ matrix

called the Jacobian matrix

standard representation $(df_{x_0})_{ij} = \frac{\partial f_i}{\partial x_j}$ ← i^{th} component of f; ← j^{th} component of x

1) this gives you a way to get df_{x_0} for anything (matrix, tensor, ...)

2) careful with dimension: differential of $f : \mathbb{R}^m \to \mathbb{R}$ is a row vector $(1 \times m)$ i.e. $df_{x_0} = (\nabla f(x_0))^T$

Chain rule:

suppose $f : \mathbb{R}^m \to \mathbb{R}^n$

$g : \mathbb{R}^n \to \mathbb{R}^q$

$$\boxed{d(g \circ f)_{x_0} = dg_{f(x_0)} \circ df_{x_0}}$$

e.g.

$f(x) = x - \mu \qquad df_{x_0} = I$

$g(x) = x^T A x \qquad dg_{x_0} = x^T(A + A^T) \qquad [\nabla g^T]$

$$d(g \circ f)_{x_0} = (x_0 - \mu)^T (A + A^T) I$$

---

for Gaussian

$$-\frac{1}{2} \sum_{i=1}^{n} (x_i - \mu)^T \Sigma^{-1} (x_i - \mu)$$

∇_μ

$$-\frac{1}{2} \sum_{i=1}^{n} 2\Sigma^{-1}(x_i - \mu) \overset{\text{want}}{=} 0$$

$$\Rightarrow \boxed{\hat{\mu}_{MLE} = \frac{1}{n} \sum_{i=1}^{n} x_i}$$

$\Rightarrow \boxed{\hat{\mu}_{MLE} = \frac{1}{n}\sum_{i=1}^{n} x_i}$

Example 2: derivative of $f(A) \triangleq \log\det(A)$

can represent derivative of a fct. from matrix to scalar, as a matrix

$$f(A+\Delta) - f(A) = \mathrm{tr}(f'(A)^T \Delta) + O(\|\Delta\|)$$
$$= \langle f'(A), \Delta \rangle + \dots$$

$\log\det(A+\Delta) - \log\det A$ [assume A is symmetric]

$= \log\det\left(A^{1/2}(I + A^{-1/2}\Delta A^{-1/2})A^{1/2}\right) - \log\det A$

$= \log |A|^{1/2} |I + A^{-1/2}\Delta A^{-1/2}| |A|^{1/2} - \log|A|$

$= \log\det(I + A^{-1/2}\Delta A^{-1/2})$

$= \sum_i \log(1 + \lambda_i(A^{-1/2}\Delta A^{-1/2}))$ ← $\det(A) = \prod_i \lambda_i(A)$ (e-values)

$= \sum_i \lambda_i(A^{-1/2}\Delta A^{-1/2}) + O(\|\Delta\|)$ ← $\log(1+x) = x + O(x^2)$ for $|x| < 1$

$= \mathrm{tr}(A^{-1/2}\Delta A^{-1/2}) + O(\|\Delta\|)$ ← $\mathrm{tr}(A) = \sum_i \lambda_i(A)$

$= \mathrm{tr}(A^{-1}\Delta) + O(\|\Delta\|)$

$\Rightarrow \boxed{\frac{d}{dA}\log\det(A) = A^{-1}}$

back to our log-likelihood:

$$\frac{n}{2}\log|\Sigma^{-1}| - \frac{n}{2}\langle \Sigma^{-1}, \tilde{\Sigma}\rangle$$

derivative w.r.t. Σ^{-1}

$$\frac{n}{2}\underbrace{(\Sigma^{-1})^{-1}}_{\Sigma} - \frac{n}{2}\tilde{\Sigma} \overset{\text{cont}}{=} 0 \Rightarrow \boxed{\hat{\Sigma}_{MLE} = \tilde{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_{MLE})(x_i - \mu_{MLE})^T}$$

some references:

- for convex optimization: Boyd & Vandenberghe's book
- DL book -- chapter 4.3 on gradient-based optimization
- for matrix calculus:
  Matrix Differential Calculus with Applications in Statistics and Econometrics, Heinz Neudecker and Jan R. Magnus -- free online