Lecture 9 - logistic regression & optimization

Tuesday, September 29, 2020 14:21

today: • logistic regression
• numerical optimization

logistic regression model:

$p(y=1|x) = \sigma(w^T x) \quad Y = \{0,1\}$

$p(y=0|x) = 1 - \sigma(w^T x) = \sigma(-w^T x)$

"Rademacher" RV

[if $Y = \{\pm 1\}$ encode $p(y|x) = \sigma(y w^T x)$]

$Y|X=x$ is Bernoulli $(\sigma(w^T x))$

$p(y|x) = \sigma(w^T x)^y \sigma(-w^T x)^{1-y}$

given $(x_i, y_i)_{i=1}^n$, max. conditional log-likelihood to estimate $\hat{w}_{MCL}$

$$\ell(w) = \sum_{i=1}^n \log p(y_i|x_i, w) = \sum_{i=1}^n y_i \log \sigma(w^T x_i) + (1-y_i) \log \sigma(-w^T x_i)$$

$\nabla_w \sigma(w^T x) = x[\sigma(w^T x)\sigma(-w^T x)]$ let $v_i \triangleq w^T x_i$

$$\nabla \ell(w) = \sum_{i=1}^n x_i \left[\frac{y_i}{\sigma(v_i)} \sigma(v_i)\sigma(-v_i) + \frac{(1-y_i)(-1)}{\sigma(-v_i)} \sigma(v_i)\sigma(-v_i) \right]$$

$$= \sum_{i=1}^n x_i \left[y_i [\underbrace{\sigma(v_i) + \sigma(-v_i)}_{1}] - \sigma(v_i) \right]$$

$$\boxed{\nabla \ell(w) = \sum_{i=1}^n x_i [y_i - \sigma(w^T x_i)]}$$

need to use numerical methods

solve for $\nabla \ell(w) = 0$ $\Rightarrow$ need to solve a transcendental equation because $\frac{1}{1+\exp(-w^T x_i)}(\ldots) = 0$ — solve for this

contrast to linear regression $\nabla \ell(w) = \sum_{i=1}^n x_i [y_i - w^T x_i]$ — linear in w

Numerical optimization:

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

1) gradient descent (1st order method)

start w_0

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

init: w_0

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

stopping criterion: $\|\nabla f(w_t)\|^2 \leq \delta$ e.g. $\delta = 10^{-6}$

note: f μ-strongly convex $\Rightarrow \|\nabla f(w_t)\|^2 \geq \delta$

$\Leftrightarrow$ $\mu > 0$

$f(w) - \frac{\mu}{2}\|w\|^2$ is convex

$\Rightarrow f(w_t) - f(w^*) \leq \frac{1}{2\mu}\|\nabla f(w_t)\|^2$

step-size rules:

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

i.e. $\|\nabla f(w) - \nabla f(w')\| \leq L\|w - w'\|$

b) decreasing step-size rule [this is more common for stochastic optimization]

$\gamma_t = \frac{c}{t}$ ← constant

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

e.g. $f(w) \triangleq \mathbb{E}_\xi g(w, \xi)$

ξ "x_i"

$w_{t+1} = w_t - \gamma_t \nabla_w g(w_t, \xi_t)$

e.g. $g(w, \xi) = \ell(y_i, h_w(x_i))$

$\xi = (x_i, y_i)$

for ERM

c) choose γ_t by "line search": $\min_{\gamma \in \mathbb{R}} f(w_t + \gamma d_t)$

costly in general

direction for update e.g. $-\nabla f(w_t)$

→ instead do approximate search e.g. Armijo line search (see Boyd's book)

15h26

Newton's method (2nd order method)

motivation: minimizing a quadratic approximation

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

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

Taylor's remainder

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

↑ quadratic model approx.

$w_{t+1} \rightsquigarrow$ minimizing $Q_t(w)$

$\nabla_w Q_t(w) = 0 \qquad \nabla f(w_t) + H(w_t)(w - w_t) = 0$

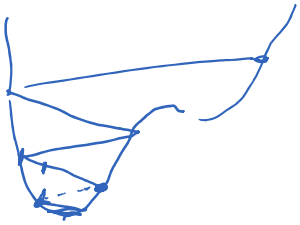
$\Rightarrow w - w_t = -H^{-1}(w_t)\nabla f(w_t)$

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

↳ Inverse Hessian

$\rightsquigarrow O(d^3)$ time to compute in general and $O(d^2)$ space

$d_t = H^{-1}\nabla f \iff H d_t = \nabla f$

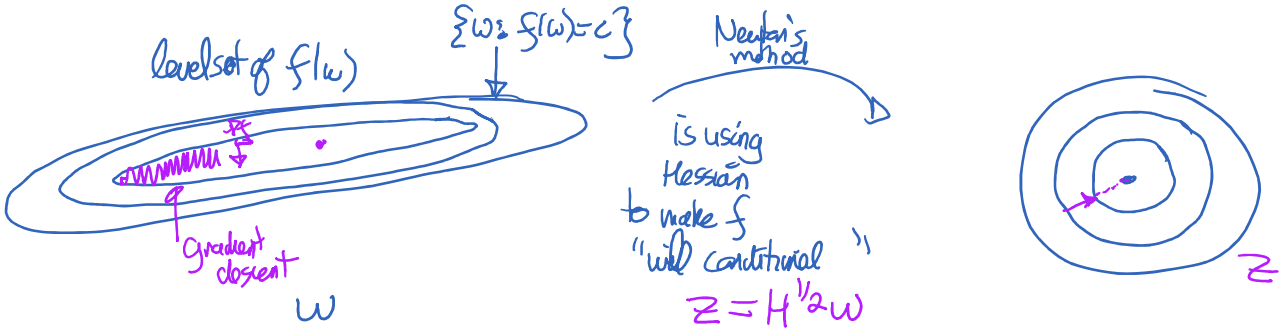


Damped Newton: you add a step-size to stabilize Newton's method

$$w_{t+1} = w_t - \underbrace{\gamma_t}_{\text{step-size}} H^{-1}(w_t)\nabla f(w_t)$$

why Newton's method?

- much faster convergence in # iterates vs. gradient descent
- affine covariant $\Rightarrow$ method is invariant to rescaling of variables



$\frac{1}{2} w^T H w = c$ ← "quadratic form"

$H = P^T \Sigma P$ H is symmetric & PSD

$\frac{1}{2} w^T P^T \Sigma P w = c$ (Σ diagonal) $\rightsquigarrow \frac{1}{2} z^T z = c$

$z = H^{1/2} w = \Sigma^{1/2} P w$

exercise to reader:

$z_{t+1} = z_t - \gamma \nabla f(z_t)$

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

Newton's method for logistic regression: IRLS

Newton's method for logistic regression : IRLS

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

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

$$v^T H v = -\sum \underbrace{(v^T x_i)(x_i^T v)}_{(x_i^T v)^2 \geq 0} \underbrace{\sigma(-)\sigma(-)}_{>0}$$

$v^T H v \leq 0 \quad \forall v \in \mathbb{R}^d$

i.e. $H \leq 0$

i.e. concave fct.

Newton is maximizing instead of min.

notation: recall $X_{n \times d} = \begin{pmatrix} -x_1^T- \\ \vdots \\ -x_n^T- \end{pmatrix}$

let $\mu_i = \sigma(w^T x_i) \in]0,1[$ $\quad \mu_t \triangleq \begin{pmatrix} \vdots \\ \sigma(w_t^T x_i) \\ \vdots \end{pmatrix}$

$$\nabla \ell(w) = \sum_i x_i [y_i - \mu_i] = X^T (y - \mu)$$

$$\text{Hessian} = -\sum_i x_i x_i^T \mu_i (1 - \mu_i) = -X^T D(w) X \quad \text{where } D_{ii} \triangleq \mu_i (1 - \mu_i)$$

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

$$= (X^T D_t X)^{-1} [(X^T D_t X) w_t + X^T (y - \mu_t)]$$

$$\boxed{w_{t+1} = (X^T D_t X)^{-1} [X^T D_t z_t]} \quad \text{where} \quad \boxed{z_t \triangleq X w_t + D_t^{-1} (y - \mu_t)}$$

this is a solution to "weighted least square problem"

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

D: weights; z_t: new target

$$\to \sum_i \frac{(z_i - w^T x_i)^2}{d_{ii}^{-1}}$$

compare with Gaussian noise model for least square: $\sum_i \frac{(y_i - x_i^T w)^2}{\sigma_i^2}$

Newton's method for logistic regression

= iterated reweighted least squares (IRLS)

note: $x = A^{-1} b \quad Ax = b \quad \min_x \|Ax - b\|^2$

Big data logistic regression:

- big d $\Rightarrow$ cannot do $O(d^2)$ or $O(d^3)$ operations $\Rightarrow$ first order methods
- if n is large, you cannot do batch methods $\nabla f(w) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w)$ $O(nd)$ (gradient of one fct.; batch gradient)

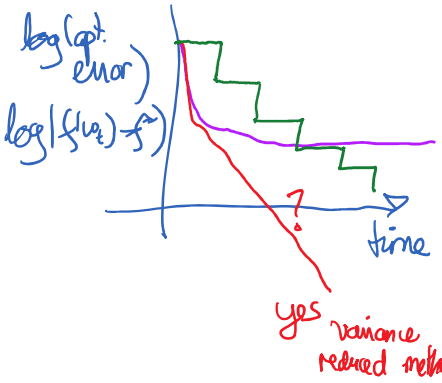
  instead you "incremental gradient methods"

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

  where i_t is picked uniformly at random

SGD → cheap updates, but slower convergence per iteration

batch gradient → expense updates, but faster convergence



→ SAG: stochastic average gradient

G.D.: $w_{t+1} = w_t - \gamma_t \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w_t)$

SAG [2012]: $w_{t+1} = w_t - \gamma_t \frac{1}{n}\sum_{i=1}^{n} v_i$ (memory)

where $v_i = \nabla f_i(w_{old})$

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

SAGA [2014]: $w_{t+1} = w_t - \gamma_t \left(\nabla f_{i_t}(w_t) + \frac{1}{n}\sum_j v_j - v_{i_t} \right)$, with $\nabla f_{i_t}(w_t)$ and v_{i_t} (variance reduction correction)

(default method for log. regression in Scikit-learn)