Lecture 8 - logistic regression & optimization

Friday, September 30, 2022 2:58 PM

today : - logistic regression
- numerical optimization

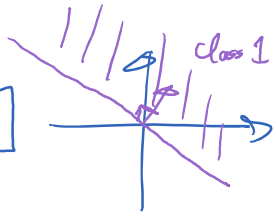
logistic regression model :

$p(y=1 \mid x, w) = \sigma(w^T x)$ $\quad y = \{0,1\}$ decision rule : $\mathbb{1}\{w^T x > 0\}$

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

[if $y = \{\pm 1\}$ "Rademacher R.V."]

encode $p(y \mid x) = \sigma(y\, w^T x)$



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

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

given $(x_i, y_i)_{i=1}^n$,

max. conditional log-likelihood to estimate $\hat{w}_{MCL}$

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

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

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

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

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

need to use numerical methods

Solve for $\nabla \ell(w) = 0$ $\Rightarrow$ need to solve a transcendental equation

because $\frac{1}{1+\exp(-w^T x_i)}(\dots) = 0$ ← solve for this

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

Numerical optimization

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

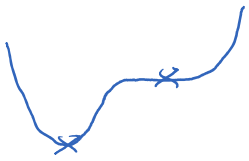
1) gradient descent (1st order method)

1) **gradient descent** (1st order method)

start w_0

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

stoping criterion : if $\|\nabla f(w_t)\|^2 < \delta$ then stop e.g. $\delta = 10^{-6}$



note: if f is μ-strongly convex ($\Leftrightarrow$ $f(w) - \frac{\mu}{2}\|w\|^2$ is convex in w) $\Rightarrow$ if $\|\nabla f(w_t)\|^2 < \delta$ $\Rightarrow f(w_t) - \min_w f(w) \leq \frac{1}{2\mu}\|\nabla f(w_t)\|^2$

step-size rules:

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

i.e. $\|\nabla f(w) - \nabla f(w')\| \leq L\|w - w'\|$ $\forall w, w' \in \mathbb{R}^d$

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

$\gamma_t = \frac{c}{t}$ ← constant

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

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

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

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

c) choose γ_t by "line search" : $\min_{\gamma \in \mathbb{R}} f(w_t + \gamma d_t)$ (d_t : direction of update, $-\nabla f(w_t)$)

costly in general

↳ instead do approximate search e.g. Armijo line search [see Boyd's book]

15h50

Newton's method (2nd order method)

motivate : minimizing a quadratic approximation

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

$\triangleq Q_t(w)$

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

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

$\mathcal{O}(\|w - w_t\|^3)$: Taylor's remainder

Q_t quadratic model approximation

w_{t+1} ⇝ obtain by minimizing $Q_t(w)$

$\nabla_w (Q_t(w)) = 0$ $\quad \nabla f(w_t) + H(w_t)(w - w_t) \overset{want}{=} 0$

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

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

↳ inverse Hessian

⇝ $O(d^3)$ time and $O(d^2)$ space

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

Note for hwk 2: solve for $H_t d_t = \nabla f(w_t)$ for d_t

$\min_d \| H_t d - \nabla f(w_t) \|^2$

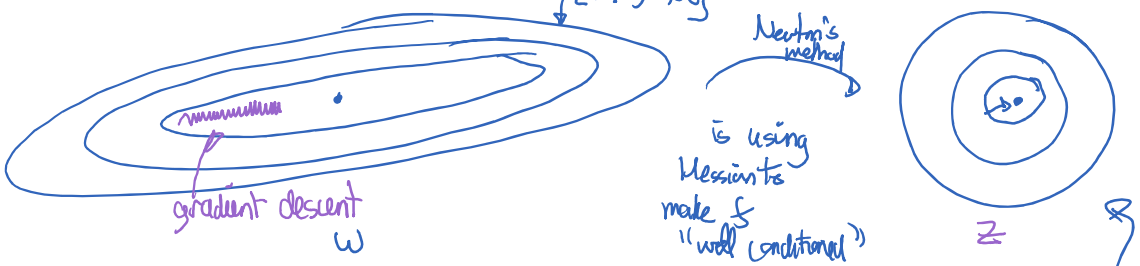
use numpy.linalg.lstsq to solve this ⇝ much more numerically stable

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 # iterations vs gradient descent
- affine invariant ⇒ method is invariant to rescaling of variables



$\frac{1}{2} w^T H w = c$

$H = P^T \Sigma P$ (H is symmetric & psd)

$\frac{1}{2} w^T P^T \Sigma P w = c \Leftrightarrow \frac{1}{2} z^T z = c$

change of variables

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

exercise to reader: $z_{t+1} = z_t - \gamma \nabla \tilde{f}(z_t)$ $\Leftrightarrow$

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

Newton's method for logistic regression: IRLS

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

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

$v^T H v = -\sum_{i=1}^{n} \underbrace{(v^T x_i)(x_i^T v)}_{(x_i^T v)^2 \geq 0} \underbrace{\sigma(-)\sigma(-)}_{\geq 0}$ $\quad v^T H v \leq 0 \ \forall v \in \mathbb{R}^d$

i.e. $H \succeq 0$

i.e. concave fct.

$\Updownarrow$

Newton is maximizing instead of minimizing

notation : recall

$X = \underset{n \times d}{\begin{pmatrix} -x_1^T- \\ \vdots \end{pmatrix}}$

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

$\mu_t \triangleq \begin{pmatrix} \vdots \\ \sigma(w^T x_i) \\ \vdots \end{pmatrix}$

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

Hessian $= -\sum_i x_i x_i^T \mu_i (1-\mu_i) = -X^T D(w) X$

$D(w)$ diagonal matrix, $D_{ii} \triangleq \mu_i(1-\mu_i)$

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

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

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

this is a solution to a "weighted least square problem"

$\min_w \| D^{1/2}(z_t - Xw) \|_2^2$

$D^{1/2}$: weights, z_t : new target

$\sum_i (z_i - w^T x_i)^2 d_{ii}^{-1}$ compare with Gaussian noise model for least square $\sum_i \frac{(y_i - w^T x_i)^2}{2\sigma_i^2}$

Newton's method for logistic regression

= iterated reweighted least squares (IRLS)

Big data logistic regression

- big d $\Rightarrow$ cannot do $O(d^2)$ or $O(d^3)$ operations $\Rightarrow$ first order methods gradient d...

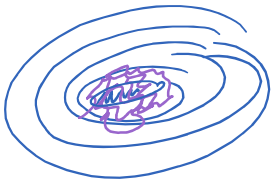
- big d $\Rightarrow$ cannot do $O(d^2)$ or $O(d^3)$ operations $\Rightarrow$ first order methods
- if n is large, you cannot use batch methods $\nabla f(w) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w)$ ($O(nd)$)

(gradient of each data point; batch gradient)

instead you use "incremental gradient methods"

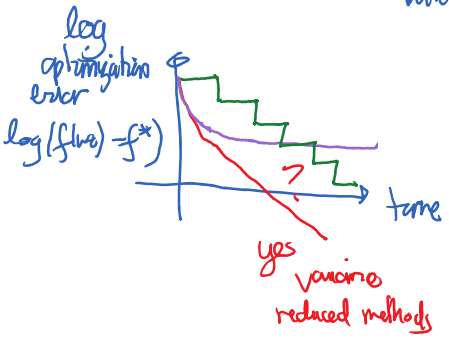
e.g. stochastic gradient descent (SGD): $w_{t+1} = w_t - \gamma_t \nabla f_{i_t}(w_t)$ ($O(d)$)

where i_t is picked uniformly at random



SGD → cheap updates, but slower convergence per iteration

batch gradient → expensive 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]: $w_{t+1} = w_t - \gamma_t \frac{1}{n}\sum_{i} v_i$ ← memory

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

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

SAGA [2014]: $w_{t+1} = w_t - \gamma_t \left(\nabla f_{i_t}(w_t) + \frac{1}{n}\sum_{j} v_j - v_{i_t} \right)$

($\nabla f_{i_t}(w_t)$; variance reduction correction)

(default method for large scale logistic regression in Scikit-learn)

SVRG