Announcements

- Project proposal due next week: Tuesday 10/24
- Still looking for people to work on deep learning Phytolith project, join #phytolith slack channel

Gradient Descent

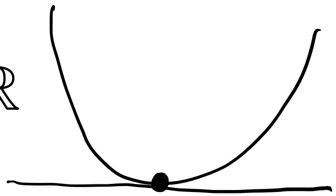
Machine Learning – CSE546
Kevin Jamieson
University of Washington

October ~~16~~ 19, 2016

Machine Learning Problems

- Have a bunch of iid data of the form:

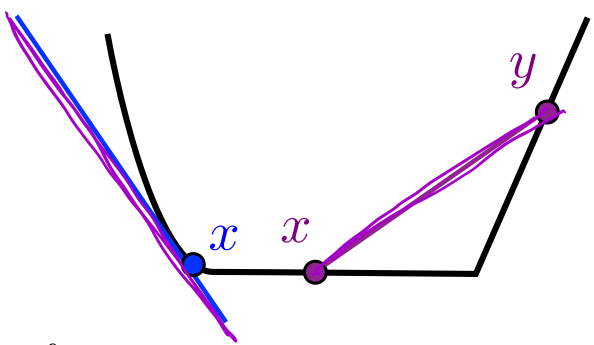
$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$



- Learning a model's parameters: $\sum_{i=1}^n \ell_i(w)$

f(x) is a minimum of f if g is a sub-gradient at x

Each $\ell_i(w)$ is convex.



g is a subgradient at x if
$$f(y) \geq f(x) + g^T(y-x)$$

f convex:

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) \qquad \forall x, y, \lambda \in [0,1]$$

$$f(y) \geq f(x) + \nabla f(x)^T (y-x) \qquad \forall x, y$$

Machine Learning Problems

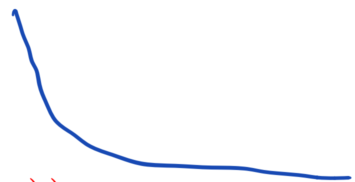
- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

- Learning a model's parameters:

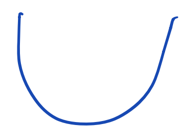
$$\sum_{i=1}^n \ell_i(w)$$

Each $\ell_i(w)$ is convex.



Logistic Loss: $\ell_i(w) = \log(1 + \exp(-y_i\, x_i^T w))$

Squared error Loss: $\ell_i(w) = (y_i - x_i^T w)^2$



Taylor Series Approximation

$\eta > 0$

- Taylor series in one dimension:

$$f(x+\delta) = f(x) + f'(x)\delta + \frac{1}{2} f''(x)\delta^2 + \dots$$

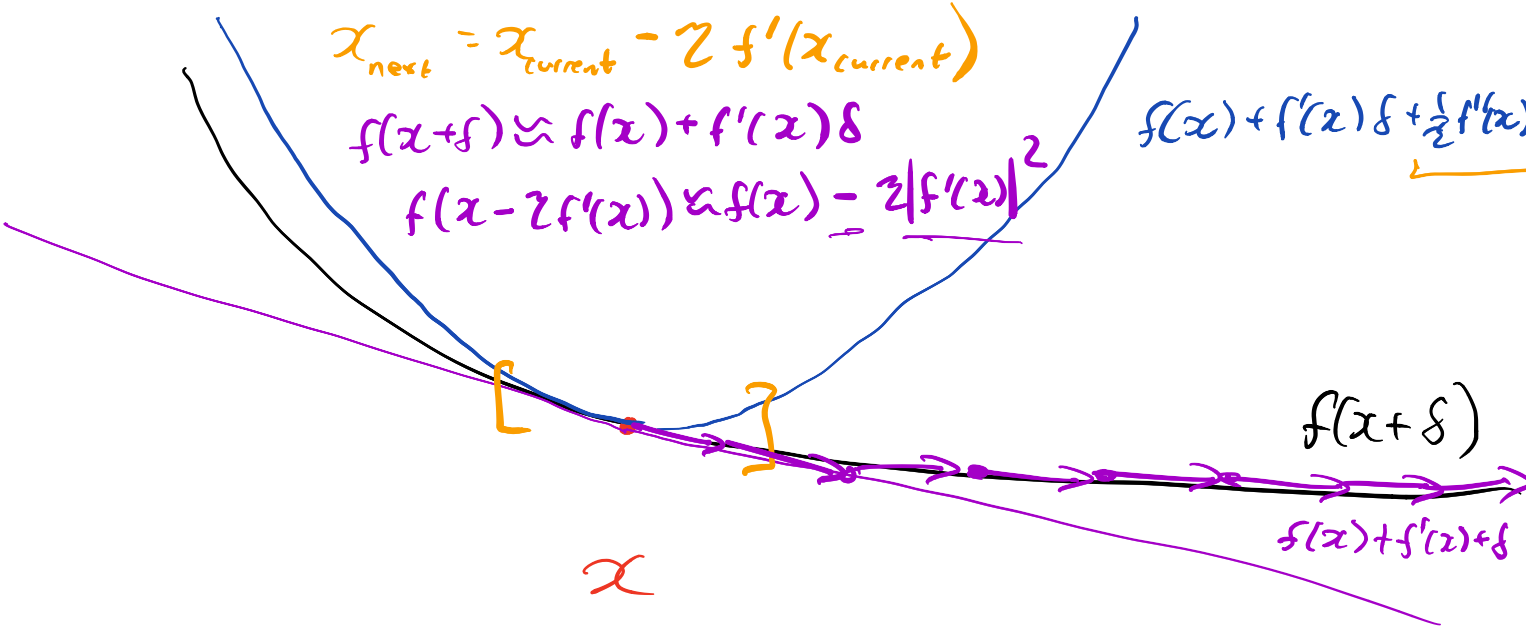
- Gradient descent:

$x_{next} = x_{current} - \eta f'(x_{current})$

$f(x+\delta) \simeq f(x) + f'(x)\delta$

$f(x - \eta f'(x)) \approx f(x) - \eta |f'(x)|^2$

$f(x) + f'(x)\delta + \frac{1}{2} f''(x)\delta^2$



Taylor Series Approximation

PSD

f is convex $\iff \nabla^2 f(x) \succeq 0 \quad \forall x$

- Taylor series in **d** dimensions:

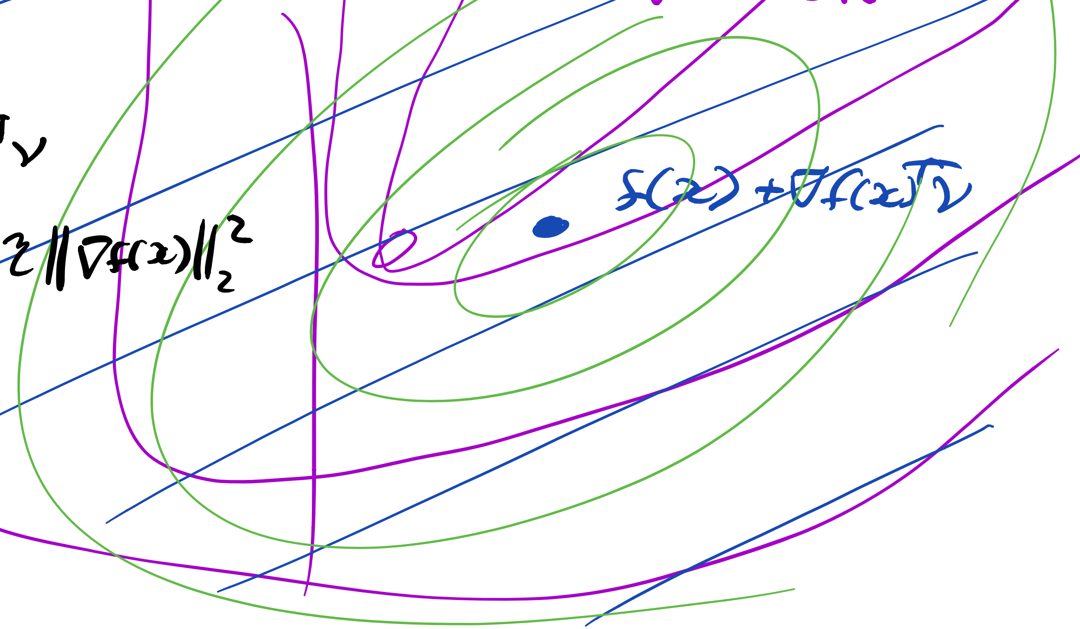
$$f(x+v) = \underbrace{f(x) + \nabla f(x)^T v}_{} + \frac{1}{2} v^T \underbrace{\nabla^2 f(x)}_{\text{Hessian of } f \text{ at } x} v + \ldots$$

$\nabla^2 f(x) \in \mathbb{R}^{d \times d}$

- Gradient descent:

$f(x+v) \simeq f(x) + \nabla f(x)^T v$

$f(x - \eta \nabla f(x)) \simeq f(x) - \eta \|\nabla f(x)\|_2^2$



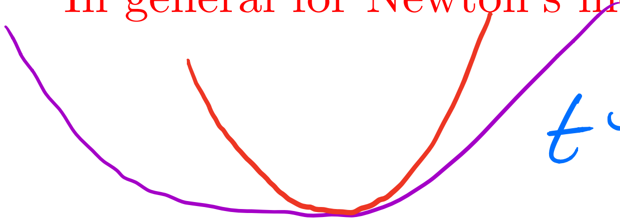
General case

$w_0 = 0$

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

$w_{t+1} = w_t + \eta p_t$

In general for Newton's method to achieve $f(w_t) - f(w_*) \leq \epsilon$:



$t \simeq O(\log(\log(1/\epsilon)))$

So why are ML problems overwhelmingly solved by gradient methods?

Hint: v_t is solution to : $\nabla^2 f(w_t) v_t = -\nabla f(w_t)$

General Convex case $f(w_t) - f(w_*) \leq \epsilon$

Newton's method:

$$t \approx \log(\log(1/\epsilon))$$

$A \preceq B$

$B - A \succeq 0$

$B-A$ is PSD

Gradient descent:

- f is *smooth* and *strongly convex:* $aI \preceq \nabla^2 f(w) \preceq bI$

$t \simeq \frac{b}{a} \log(1/\varepsilon)$

- f is *smooth*: $\nabla^2 f(w) \preceq bI$

$t \simeq \frac{b}{\varepsilon}$

Nesterov's method $\sqrt{\frac{b}{\varepsilon}}$

- f is potentially non-differentiable: $||\nabla f(w)||_2 \leq c$

$1/\varepsilon^2$

Clean convergence proofs: Bubeck

Nocedal +Wright, Bubeck

Other: BFGS, Heavy-ball, BCD, SVRG, ADAM, Adagrad,…

Revisiting… Logistic Regression

Machine Learning – CSE546
Kevin Jamieson
University of Washington

October ~~16~~ 14, 2016

Loss function: Conditional Likelihood

- Have a bunch of iid data of the form: $\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^n P(y_i|x_i, w) \qquad P(Y = y|x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

$$\frac{e^a}{1+e^a} = 1 - \frac{1}{1+e^a}$$

$$f(w) = \arg\min_w \sum_{i=1}^n \log(1 + \exp(-y_i\, x_i^T w))$$

$$\nabla f(w) = \sum_{i=1}^n \frac{1}{1+\exp(-y_i x_i^T w)} -y_i x_i \exp(-y_i x_i^T w)$$

$$= \sum_{i=1}^n (1 - \sigma_i(w))(-y_i x_i)$$

$$w_0 = 0$$

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

$$\sigma_i(w) = \frac{1}{1+\exp(-y_i x_i^T w)}$$

Online Learning

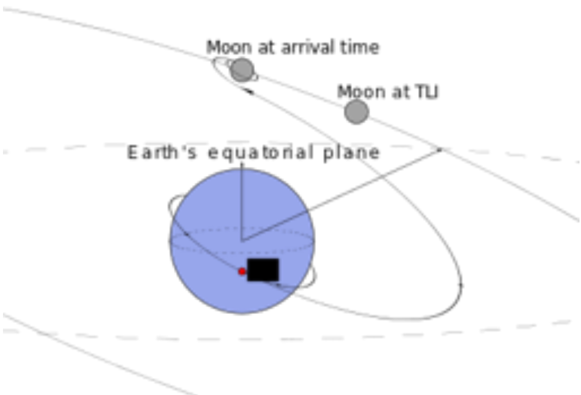
Machine Learning – CSE546

Kevin Jamieson

University of Washington

October ~~18~~ 24, 2016

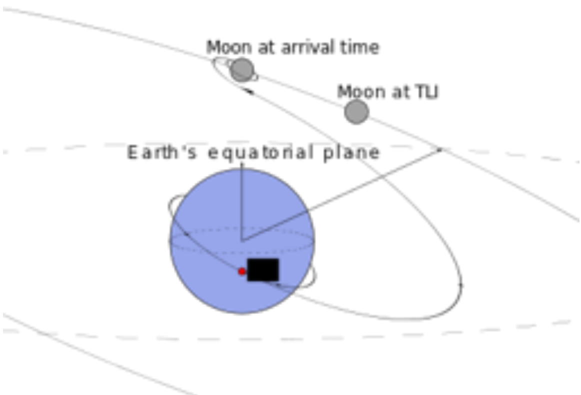
Going to the moon



Guidance computer predicts trajectories around moon and back with

- Noisy sensors
- Imperfect models
- Little computational power
- Big risk of failure

Going to the moon



Guidance computer predicts trajectories around moon and back with
- Noisy sensors
- Imperfect models
- Little computational power
- Big risk of failure

Why is Tom Hanks flying erratically?

Because they didn't have the power to turn on the Kalman FIlter!

State Estimation

- Predict current state given past state and current control input

$$\widetilde{w}_n = f(w_{n-1}) + g(u_n)$$

- Given current context, x_n compare your prediction to noisy measurement y_n

$$\ell_n(\widetilde{w}_n) = (y_n - h(x_n, \widetilde{w}_n))^2$$

- Update current state to include measurement

$$w_n = \widetilde{w}_n - K_n \nabla_w \ell_n(w)\Big|_{w=\widetilde{w}_n}$$

Kalman filter does optimal least squares state estimation if f, g, h are linear!

Recursive Least Squares (RLS)

Least squares = special case of Kalman Filter: no dynamics, no control

$$\widetilde{w}_n = f(w_{n-1}) + g(u_n)$$

$= w_{n-1}$

$$\ell_n(\widetilde{w}_n) = (y_n - h(x_n, \widetilde{w}_n))^2$$

$h(x,y) := x^T y$

$= (y_n - x_n^T \widetilde{w}_n)^2$

$= (y_n - x_n^T w_{n-1})^2$

$$w_n = \widetilde{w}_n - K_n \nabla_w \ell_n(w)\Big|_{w=\widetilde{w}_n}$$

$= w_{n-1} + 2(y_n - x_n^T w_{n-1}) K_n x_n$

Recursive Least Squares (RLS)

Least squares = special case of Kalman Filter: no dynamics, no control

$$
\begin{aligned}
\widetilde{w}_n &= f(w_{n-1}) + g(u_n) \\
&= w_{n-1}
\end{aligned}
$$

$$
\begin{aligned}
\ell_n(\widetilde{w}_n) &= (y_n - h(x_n, \widetilde{w}_n))^2 \\
&= (y_n - x_n^T \widetilde{w}_n)^2 \\
&= (y_n - x_n^T w_{n-1})^2
\end{aligned}
$$

Ideally:

$$w_n = \arg\min_w \sum_{i=1}^{n} (y_i - x_i^T w)^2$$

$$
\begin{aligned}
w_n &= \widetilde{w}_n - K_n \nabla_w \ell_n(w)\Big|_{w=\widetilde{w}_n} \\
&= w_{n-1} + 2(y_n - x_n^T w_{n-1}) K_n x_n
\end{aligned}
$$

Recursive Least Squares (RLS)

Sherman–Morrison: $(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u}.$

$$\boxed{w_n = \underbrace{\left(\sum_{i=1}^n x_i x_i^T\right)^{-1}}_{S_n} \underbrace{\sum_{i=1}^n x_i y_i}_{b_n}}$$

Ideally:
$$w_n = \arg\min_w \sum_{i=1}^n (y_i - x_i^T w)^2$$

$$= \Big(\underbrace{\sum_{i=1}^{n-1} x_i x_i^T}_{XX^T} + x_n x_n^T\Big)^{-1} \Big(\underbrace{\sum_{i=1}^{n-1} x_i y_i}_{Xy} + x_n y_n\Big)$$

$X \in \mathbb{R}^{d \times n-1}$

$$= \left((XX^T)^{-1} - \frac{(XX^T)^{-1} x_n x_n^T (XX^T)^{-1}}{1 + x_n^T (XX^T)^{-1} x_n} \right) \left(X^T y + x_n y_n \right)$$

$$= \underbrace{\left(S_{n-1} - \frac{S_{n-1} x_n x_n^T S_{n-1}}{1 + x_n^T S_{n-1} x_n} \right)}_{S_n} \left(b_{n-1} + x_n y_n \right)$$

$$\boxed{= S_n b_n}$$

$$S_n = \left(\sum_{i=1}^n x_i x_i^T\right)^{-1} = S_{n-1} - \frac{S_{n-1} x_n x_n^T S_{n-1}}{1 + x_n^T S_{n-1} x_n}$$

$$b_n = \sum_{i=1}^n x_i y_i = b_{n-1} + x_n y_n$$

Recursive Least Squares (RLS)

$$w_n = \left(\sum_{i=1}^n x_i x_i^T \right)^{-1} \sum_{i=1}^n x_i y_i$$

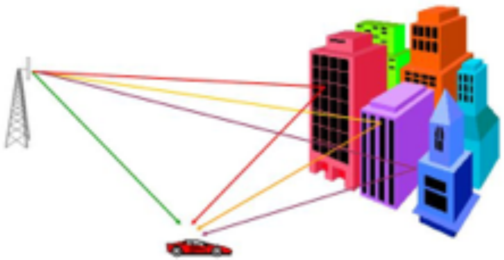
Great, what's the time-complexity of this?

$d^{1.5}$ (matrix-vector multiply

It is 2017. Not the 60's… is limited computation still really a problem?

Digital Signal Processing

The original "Big Data"

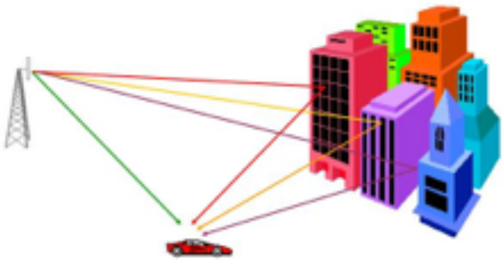


Wifi/cell-phones are *constantly* solving least squares to invert out multipath

Low power devices, high data rates

Digital Signal Processing

The original "Big Data"



Wifi/cell-phones are *constantly* solving least squares to invert out multipath

Low power devices, high data rates

Gigabytes of data per second

Incremental Gradient Descent

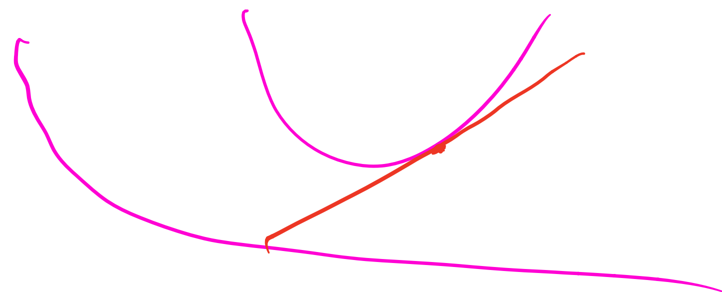
Note: no matrix multiply

(x_t, y_t) arrive:

$$w_{t+1} = w_t - \eta \left[\nabla_w (y_t - x_t^T w)^2 \Big|_{w=w_t} \right]$$

We know RLS is exact. How much worse is this?

In general convex $\ell_t(w)$ arrives:



$$\ell(\cdot) \text{ is convex } \iff \ell(y) \geq \ell(x) + \nabla \ell(x)^T (y - x) \ \forall x, y$$

$w_{t+1} = w_t - \eta \nabla \ell(w_t)$

Incremental Gradient Descent

$$||w_{t+1} - w_*||_2^2 = ||w_t - \eta \nabla \ell_t(w_t) - w_*||_2^2$$

$$= ||w_t - w_*||_2^2 - 2\eta \nabla \ell_t(w_t)^T (w_t - w_*) + \eta^2 ||\nabla \ell_t(w_t)||_2^2$$

$$\ell_t(w_t) - \ell_t(w_*) \le \nabla \ell_t(w_t)^T (w_t - w_*) = \frac{||w_t - w_*||_2^2 - ||w_{t+1} - w_*||_2^2 + \eta^2 ||\nabla \ell_t(w_t)||_2^2}{2\eta}$$

$$\frac{1}{t} \sum_{s=0}^{t} \ell_t(w_t) - \ell_t(w_*) \le \frac{\sum_{s=0}^{t} ||w_s - w_*||_2^2 - ||w_{s+1} - w_*||_2^2 + \eta^2 \sum_{s=0}^{t} ||\nabla \ell_s(w_s)||_2^2}{2\eta}$$

$$\le \frac{||w_0 - w_*||_2^2 - \cancel{||w_{t+1} - w_*||_2^2} + t\eta^2 \max_{1 \le s \le t} ||\nabla \ell_s(w_s)||_2^2}{2\eta t}$$

$\eta = \frac{1}{\sqrt{t}}$

$$\le \frac{||w_0 - w_*||_2^2 + \max_s ||\nabla \ell_s(w_s)||_2^2}{2\sqrt{t}}$$

Incremental Gradient Descent

Stochastic Gradient Descent

- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

- Learning a model's parameters:

Each $\ell_i(w)$ is convex.

$$\frac{1}{n}\sum_{i=1}^n \ell_i(w)$$

Stochastic Gradient Descent

- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

- Learning a model's parameters:

$$\frac{1}{n}\sum_{i=1}^n \ell_i(w)$$

Each $\ell_i(w)$ is convex.

Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla_w \left(\frac{1}{n}\sum_{i=1}^n \ell_i(w) \right)\Bigg|_{w=w_t} = w_t - \frac{\eta}{n}\sum_{i=1}^n \nabla_w \ell_i(w)$$

Stochastic Gradient Descent

- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

- Learning a model's parameters:

Each $\ell_i(w)$ is convex.

$$\frac{1}{n}\sum_{i=1}^n \ell_i(w)$$

Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla_w \left(\frac{1}{n}\sum_{i=1}^n \ell_i(w) \right) \Big|_{w=w_t}$$

Stochastic Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla_w \ell_{I_t}(w) \Big|_{w=w_t}$$

I_t drawn uniform at random from $\{1, \dots, n\}$

$$\mathbb{E}[\nabla \ell_{I_t}(w)] = \frac{1}{n}\sum_{i=1}^n \nabla \ell_i(w)$$

Stochastic Gradient Descent

Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla_w \left(\underbrace{\frac{1}{n}\sum_{i=1}^{n} \ell_i(w)}_{\ell(w)} \right) \Bigg|_{w=w_t}$$

Stochastic Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla_w \ell_{I_t}(w) \Big|_{w=w_t}$$

I_t drawn uniform at random from $\{1, \ldots, n\}$

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} \ell_t(w_t) - \ell_t(w_*)\right] \leq \mathbb{E}\left[\frac{\|w_0 - w_*\|_2^2 + \max_s \|\nabla \ell_s(w_s)\|_2^2}{2\sqrt{T}}\right]$$

$$\bar{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$$

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\ell(w_t)] - \ell(w_*) =$$

$$\leq$$

$$\ell(\bar{w}_T) - \ell(w_*) \leq \frac{\|w_0 - w_*\|_2^2 + \max_s \mathbb{E}\|\nabla \ell_s(w_s)\|_2^2}{2\sqrt{T}}$$

Stochastic Gradient Ascent for Logistic Regression

- Logistic loss as a stochastic function:

$$E_{\mathbf{x}}\left[\ell(\mathbf{w}, \mathbf{x})\right] = E_{\mathbf{x}}\left[\ln P(y|\mathbf{x}, \mathbf{w}) - \lambda ||\mathbf{w}||_2^2\right]$$

- Batch gradient ascent updates:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \frac{1}{N} \sum_{j=1}^{N} x_i^{(j)} [y^{(j)} - P(Y=1|\mathbf{x}^{(j)}, \mathbf{w}^{(t)})] \right\}$$

- Stochastic gradient ascent updates:
  - Online setting:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta_t \left\{ -\lambda w_i^{(t)} + x_i^{(t)} [y^{(t)} - P(Y=1|\mathbf{x}^{(t)}, \mathbf{w}^{(t)})] \right\}$$