

# SDS 385: Stat Models for Big Data

## Lecture 2: Linear Regression

---

Purnamrita Sarkar  
Department of Statistics and Data Science  
The University of Texas at Austin  
<https://psarkar.github.io/teaching>

# Linear regression: recap

Given  $n$  pairs  $(\mathbf{x}_i, y_i) \in \mathbb{R}^{p+1 \times 1}$ , consider the model:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \quad \epsilon_i \sim N(0, \sigma^2)$$

where:

$$\mathbf{y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}, \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}, \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}, \text{ and } \mathbf{x} = \begin{bmatrix} 1 & x_{12} & \dots & x_{1p} \\ 1 & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n2} & \dots & x_{np} \end{bmatrix}$$

- $\mathbf{X}, \mathbf{y}$  are given, you need to estimate  $\boldsymbol{\beta}$ .

## MLE - recap

$$f(\mathbf{y}|\mathbf{X}; \boldsymbol{\beta}) \propto \exp\left(\frac{-(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})}{2\sigma^2}\right)$$

- Take Log, we can get:

$$\frac{-(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})}{2\sigma^2} \quad (1)$$

- Same drill – differentiate and set it to zero.

$$-\mathbf{X}^T (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) = 0 \rightarrow \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\beta}} = \mathbf{X}^T \mathbf{y} \rightarrow \hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

- What happens when  $p \gg n$ ?  $\mathbf{X}^T \mathbf{X}$  is not invertible.

# Ridge regression

- Add a prior to  $\beta$ , i.e.  $\beta \sim N(0, \lambda I_p)$ , or think of adding a regularization that penalizes large values of  $\beta^T \beta$ .
- So now we have:

$$f(\mathbf{y}|\mathbf{X}, \beta) \propto \exp\left(\frac{-(\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta)}{2\sigma^2} - \lambda \beta^T \beta\right)$$

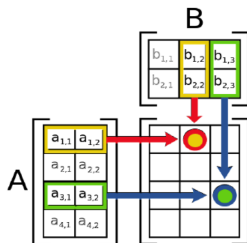
- Differentiating and setting to zero gives:

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X} + \lambda I_p)^{-1} \mathbf{X}^T \mathbf{y}$$

- Phew! – no issues with invertibility of  $\mathbf{X}^T \mathbf{X}$

# Exact computation

- If  $\mathbf{X}$  was dense, how much time would the computation of  $\mathbf{X}^T \mathbf{X}$  take?
- Wait, what is dense?
- Well dense means,  $\mathbf{X}$  has about  $\Theta(np)$  non-zero elements.



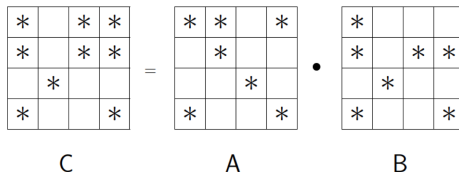
**Figure 1:** Dense matrix multiplication<sup>1</sup>

- So  $O(n)$  computation for each of  $p^2$  entries, and hence  $np^2$ .

<sup>1</sup>Borrowed from Cho-Jui Hsieh's classnotes at UC-Davis.

# Sparse matrix data structures

- How do you store a sparse vector?
- All you need is two vectors: one is of the indices of nonzero elements and one is the values.



$$C_{ij} = \sum_k A_{ik} \cdot B_{kj}$$

**Figure 2:** Sparse matrix multiplication<sup>2</sup>

- If  $A$  has  $nnz$  non-zeroes, then worst case, the complexity is  $O(n \times nnz)$  operations for multiplying a sparse matrix with another dense matrix.

<sup>2</sup>Borrowed from Grey Ballard and Alex Druinsky, SIAM conf. on Lin. Algebra

## Back to regression

- Inverting a  $p \times p$  matrix takes  $O(p^3)$  time.
- Alternatives: use linear solvers of the form  $A\mathbf{u} = \mathbf{v}$ .
- Here  $A = \mathbf{X}^T \mathbf{X} + \lambda I_p$ ,  $\mathbf{v} = \mathbf{X}^T \mathbf{y}$  and  $\mathbf{u} = \boldsymbol{\beta}$ .
- Unless your matrix  $A$  has some structure, linear solvers can also be expensive. However, if it does have structure, e.g. its diagonally dominant, etc, then there are nearly linear time solvers.
- Typically for regression, we don't expect to have such structure.
- So, what can be done?

# Iterative solvers

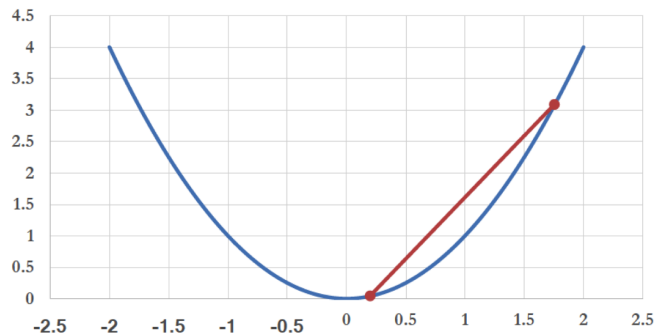
- Lets talk about gradient descent type methods.

- Model:  $\sum_{i=1}^n f(\underbrace{x_i}_{\text{data}}; \underbrace{\beta}_{\text{parameter}})$

- Example of  $f$ : negative log-likelihood over iid data-points, e.g. linear regression, logistic regression, etc.
- Goal:  $\hat{\beta} = \arg \min_{\beta} f(x_i; \beta)$
- Lets deal with convex loss functions.



# Convex functions



$$f(x) = x^2$$

**Figure 3:** A convex function

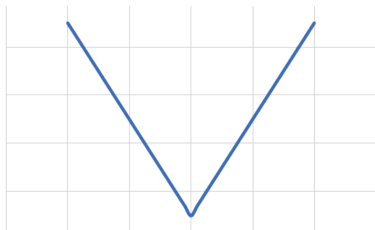
$$\forall \alpha \in [0, 1], f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y)$$

## Quadratic function $f(y) = y^2$

$$\begin{aligned} f(\alpha x + (1 - \alpha)y) &= (\alpha x + (1 - \alpha)y)^2 \\ &= \alpha^2 x^2 + (1 - \alpha)^2 y^2 + 2\alpha(1 - \alpha)xy \\ &\leq \alpha^2 x^2 + (1 - \alpha)^2 y^2 + \alpha(1 - \alpha)(x^2 + y^2) \\ &= \alpha x^2 + (1 - \alpha)y^2 \end{aligned}$$

- Where did I use  $\alpha \in [0, 1]$ ?

## Convex functions $f(y) = |y|$



**Figure 4:**  $f(y) = |y|$

$$\begin{aligned} f(\alpha x + (1 - \alpha)y) &= |\alpha x + (1 - \alpha)y| \\ &\leq |\alpha x| + |(1 - \alpha)y| \\ &\leq \alpha|x| + (1 - \alpha)|y| \end{aligned}$$

# Local optima is also global optima

## Theorem

Consider an optimization problem  $\min_x f(x)$  where  $f$  is convex. Let  $x^*$  be a local minima. Prove that it is also a global minima.

## Proof.

- By definition,  $\exists p > 0$ , such that  $\forall x \in B(x^*, p)$ ,  $f(x) \geq f(x^*)$ .
- If  $x^*$  is not the global optima,  $z \notin B(x^*, p)$  such that  $f(z) < f(x^*)$ .
- Take  $t \in [0, 1]$  and the point  $y = tx^* + (1 - t)z$ .  
$$f(y) \leq tf(x^*) + (1 - t)f(z) < f(x^*)$$
- Now  $|y - x^*| = (1 - t)|z - x^*|$ . If we take  $t$  large enough such that  $(1 - t)|z - x^*| \leq p$ , then  $y \in B(x^*, p)$  but  $f(y) < f(x^*)$ , which is a contradiction.



# Properties of convex functions

- Non-negative linear combinations of convex functions is also convex.
  - For example  $af(x) + bg(x)$  where  $f, g$  are both convex and  $a, b \geq 0$  is also convex.
- A convex function composed with an affine function is also convex.
- Point-wise maxima of convex functions is convex.

# Properties of convex functions

- Compositions of convex functions not necessarily convex
- $f, g$  convex.
  - Is  $f - g$  convex?
  - Is  $fg$  convex?

## Convex functions: other definitions

- First order:

$$\langle x - y, \nabla f(x) - \nabla f(y) \rangle \geq 0$$

- Second order:

$$\nabla^2 f(x) \succeq 0$$

- Example:  $f(x) = x^2$ .  $(x - y)^2 > 0$  and  $f''(x) = 2 \geq 0$ .
- Example:  $f(x) = \log(1 + e^x)$ .
  - $f'(x) = \frac{1}{1 + e^{-x}}$  is monotonically increasing with  $x$  and so the first order condition is satisfied.
  - Second order:  $f''(x) = f(x)(1 - f(x)) \geq 0$

## Strongly convex functions – add curvature

- First order:

$$\langle x - y, \nabla f(x) - \nabla f(y) \rangle \geq \mu \|x - y\|^2$$

- Second order:

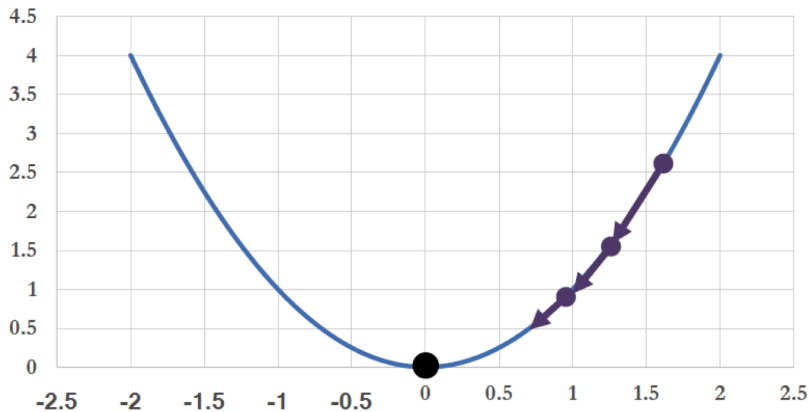
$$\nabla^2 f(x) \succeq \mu I$$

- So you add a margin to each inequality.



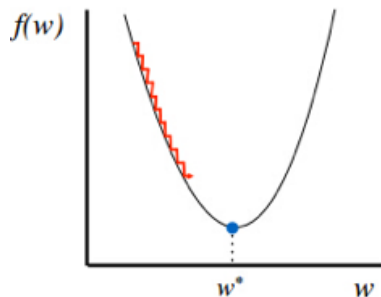
# Gradient descent

$$\beta \leftarrow \beta - \alpha \nabla f(\beta)$$

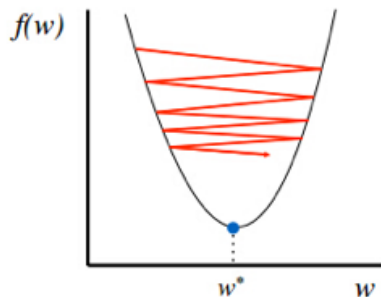


**Figure 5:** Convex function minimization with gradient descent

## Step size



Too small: converge  
very slowly



Too big: overshoot and  
even diverge

**Figure 6:** Choice of step size is crucial

# Newton Raphson

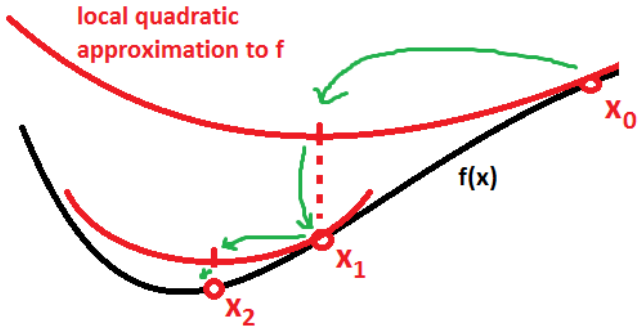


Figure 7: Newton Raphson<sup>4</sup>

<sup>4</sup>Borrowed from Nick Alger, [math.stackexchange.com](http://math.stackexchange.com)

## Newton Raphson cont.

- GD takes into account only first order information.
- NR also takes second order information.
- In particular it uses the Hessian,

$$H(i,j) = \frac{\partial^2 f}{\partial \theta_i \partial \theta_j}, \text{ where } i,j \in \{1, \dots, k\},$$

- Lets try to minimize a quadratic function:

$$f = \mathbf{a} + \mathbf{b}^T \boldsymbol{\theta} + \frac{1}{2} \boldsymbol{\theta}^T C \boldsymbol{\theta}.$$

- $C$  is positive semidefinite and so this is a convex function.
- We can minimize the function by differentiating it and by setting the result equal to 0:

$$\begin{aligned}\nabla f(\boldsymbol{\theta}^*) &= \mathbf{b} + C\boldsymbol{\theta}^* = 0 \\ \boldsymbol{\theta}^* &= -C^{-1}\mathbf{b}\end{aligned}$$

## Newton Raphson cont.

- In the neighborhood of  $\theta_t$ , we can use the approximation:

$$f(\theta^{(t)} + \mathbf{h}) \approx f(\theta^{(t)}) + \nabla f(\theta^{(t)})^T \mathbf{h} + \frac{1}{2} \mathbf{h}^T H(\theta^{(t)}) \mathbf{h}. \quad (2)$$

- Therefore the general updating rule is

$$\theta^{(t+1)} = \theta^{(t)} - H^{-1}(\theta^{(t)}) \cdot \nabla f(\theta^{(t)})$$

- You can use a stepsize here as well.

# Gradient Descent

- for  $t = 1 : T$  (or until convergence)
- Do  $\beta_{t+1} \leftarrow \beta_t - \alpha \nabla f(\beta)$

## Theorem

*Let  $\beta^*$  is the global minima, and the second derivative is bounded as  $\mu I \preceq H(\beta) \preceq LI$ . Then with  $\alpha = 2/(L + \mu)$ , gradient descent converges geometrically, i.e.*

$$\|\beta_{t+1} - \beta^*\| \leq \frac{L - \mu}{L + \mu} \|\beta_t - \beta^*\|$$

- Lets look at the distance from the optima:

$$\begin{aligned}\beta_{t+1} - \beta^* &= \beta_t - \beta^* - \alpha(\nabla f(\beta_t) - \nabla f(\beta^*)) \\ &= \beta_t - \beta^* - \alpha H(z_t)(\beta_t - \beta^*) \\ &= (I - \alpha \nabla^2 f(z_t))(\beta_t - \beta^*)\end{aligned}$$

- Now take norm of both sides and use Triangle.

$$\begin{aligned}\|\beta_{t+1} - \beta^*\| &\leq \|I - \alpha H(z_t)\| \|\beta_t - \beta^*\| \\ &\leq \max(|1 - \alpha\mu|, |1 - \alpha L|) \|\beta_t - \beta^*\|\end{aligned}$$

# Linear convergence

- Set  $\alpha = 2/(L + \mu)$ . You get

$$\|\beta_{t+1} - \beta^*\| \leq \frac{L - \mu}{L + \mu} \|\beta_t - \beta^*\|$$

- Finally after  $T$  iterations, we have:

$$\|\beta_{T+1} - \beta^*\| \leq \left( \frac{L - \mu}{L + \mu} \right)^T \|\beta_1 - \beta^*\|$$

- This is a typical “linear” contraction result.



## Theorem

*Let  $\beta^*$  is the global minima, and the second derivative is  $L$  Lipschitz, i.e.  $\|H(x) - H(x')\| \leq \kappa\|x - x'\|$  and  $\|H^{-1}\| \leq 1/\mu$ . Then with  $\alpha = 1$ , Newton Raphson converges quadratically, i.e.*

$$\|\beta_{t+1} - \beta^*\| \leq \kappa/\mu \|\beta_t - \beta^*\|^2$$

- Note that this is useful only when  $\|\beta_{t+1} - \beta^*\| \ll 1$

$$\begin{aligned}
 \beta_{t+1} - \beta^* &= \beta_t - \beta^* - H^{-1}(\beta_t)(\nabla f(\beta_t) - \nabla f(\beta^*)) \\
 &= \beta_t - \beta^* - H^{-1}(\beta_t)H(z_t)(\beta_t - \beta^*) \\
 &= (I - H^{-1}(\beta_t)H(z_t))(\beta_t - \beta^*) \\
 &= H^{-1}(\beta_t)(H(\beta_t) - H(z_t))(\beta_t - \beta^*) \\
 \|\beta_{t+1} - \beta^*\| &\leq \|H^{-1}(\beta_t)\| \|H(\beta_t) - H(z_t)\| (\beta_t - \beta^*) \\
 &\leq \kappa/\mu \|\beta_t - z_t\| (\beta_t - \beta^*) \\
 &\leq \kappa/\mu \|\beta_t - \beta^*\|^2
 \end{aligned}$$

# Scalability concerns

- You have to calculate the gradient every iteration.
- Take ridge regression.
- You want to minimize  $1/n \left( (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) - \lambda \boldsymbol{\beta}^T \boldsymbol{\beta} \right)$
- Take a derivative:  $(-2\mathbf{X}^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) - 2\lambda \boldsymbol{\beta})/n$
- Grad descent update takes  $\boldsymbol{\beta}_{t+1} \leftarrow \boldsymbol{\beta}_t + \alpha (\mathbf{X}^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}_t) + \lambda \boldsymbol{\beta}_t)$
- What is the complexity?
  - Trick: first compute  $\mathbf{y} - \mathbf{X}\boldsymbol{\beta}$ .
  - $np$  for matrix vector multiplication,  $\text{nnz}(\mathbf{X})$  for sparse matrix vector multiplication.
  - Remember the examples with humongous  $n$  and  $p$ ?

# Acknowledgment

Cho-Jui Hsieh and Christopher De Sa's large scale ML classes.