S&DS 365 / 565

# Data Mining and Machine Learning

Yale

# For today

- Finding solutions with generative models
- Recall linear regression solution
- iterative methods
- optimization error versus statistical error

# Recap

Exponential families: power probabilistic model

Can use to **roughly** approximate $\mathbb{P}_\beta(y|x)$

Classification: $\widehat{f}(x) = \mathbb{1}\left[\mathbb{P}_\beta(y=1|x) > \mathbb{P}_\beta(y=0|x)\right]$

How do we find parameters?

Start with different probabilistic model

# Generative modeling

Exponential families focus on $\mathbb{P}(y|x)$

Generative modeling: $\mathbb{P}(x|y)$

Want $\mathbb{P}(y|x)$

# Generative modeling

Bayes rule

$$\mathbb{P}(y|x) = \frac{\overset{\mathbb{P}}{\cancel{prob}}(x|y)\mathbb{P}(y)}{\mathbb{P}(x)}$$

Classification:

$$\mathbb{P}(y) = \begin{cases} \pi_0 & y = 0 \\ \pi_1 & y = 1 \end{cases}$$

$$\mathbb{P}(x) = \sum_j \mathbb{P}(x|y=j)\mathbb{P}(y=j)$$

# Generative modeling

Modeling choice: $\mathbb{P}(x|y)$

Gaussian: $x|y \sim N(\mu_y, \Sigma_y)$ (exploring in homework)

Naive Bayes: $x \in \mathbb{R}^d$ with

$$\mathbb{P}(x|y) = \prod_{j=1}^{d} \mathbb{P}(x_{(j)}|y)$$

Each coordinate is **independent** conditioned on $y$

# Naive Bayes

Can pick any univariate distribution for $x_{(j)}|y$

Binary classification:

$$\mathbb{P}(y = 1|x) > \mathbb{P}(y = 0|x)$$

# Naive Bayes

Can pick any univariate distribution for $x_i|y$

Binary classification:

$$\mathbb{P}(y=1|x) > \mathbb{P}(y=0|x)$$

$$\frac{\mathbb{P}(x|y=1)\mathbb{P}(y=1)}{\mathbb{P}(x)} > \frac{\mathbb{P}(x|y=0)\mathbb{P}(y=0)}{\mathbb{P}(x)}$$

# Naive Bayes

Can pick any univariate distribution for $x_i|y$

Binary classification:

$$\mathbb{P}(y=1|x) > \mathbb{P}(y=0|x)$$

$$\frac{\mathbb{P}(x|y=1)\mathbb{P}(y=1)}{\mathbb{P}(x)} > \frac{\mathbb{P}(x|y=0)\mathbb{P}(y=0)}{\mathbb{P}(x)}$$

$$\mathbb{P}(x|y=1)\mathbb{P}(y=1) > \mathbb{P}(x|y=0)\mathbb{P}(y=0)$$

# Naive Bayes

Can pick any univariate distribution for $x_i|y$

Binary classification:

$$\mathbb{P}(y = 1|x) > \mathbb{P}(y = 0|x)$$

$$\frac{\mathbb{P}(x|y = 1)\mathbb{P}(y = 1)}{\mathbb{P}(x)} > \frac{\mathbb{P}(x|y = 0)\mathbb{P}(y = 0)}{\mathbb{P}(x)}$$

$$\mathbb{P}(x|y = 1)\mathbb{P}(y = 1) > \mathbb{P}(x|y = 0)\mathbb{P}(y = 0)$$

$$\log(\mathbb{P}(x|y = 1)) + \log(\mathbb{P}(y = 1)) > \log(\mathbb{P}(x|y = 0)) + \log(\mathbb{P}(y = 0))$$

# Naive Bayes

Can pick any univariate distribution for $x_i|y$

Binary classification:

$$\mathbb{P}(y=1|x) > \mathbb{P}(y=0|x)$$

$$\frac{\mathbb{P}(x|y=1)\mathbb{P}(y=1)}{\mathbb{P}(x)} > \frac{\mathbb{P}(x|y=0)\mathbb{P}(y=0)}{\mathbb{P}(x)}$$

$$\mathbb{P}(x|y=1)\mathbb{P}(y=1) > \mathbb{P}(x|y=0)\mathbb{P}(y=0)$$

$$\log(\mathbb{P}(x|y=1)) + \log(\mathbb{P}(y=1)) > \log(\mathbb{P}(x|y=0)) + \log(\mathbb{P}(y=0))$$

$$\sum_{j=1}^{d} \log\left(\frac{\mathbb{P}(x_{(j)}|y=1)}{\mathbb{P}(x_{(j)}|y=0)}\right) > \log\left(\frac{\pi_0}{\pi_1}\right)$$

# Example: spam

Each feature is count of a word

More easily each feature $x_{(j)} \in \{0, 1\}$ is if word $j$ appears or not

$$\mathbb{P}(x_{(j)}|y) = p_{j,y}^{x_{(j)}}(1 - p_{j,y})^{1-x_{(j)}}$$

# Example: spam

Each feature is count of a word

More easily each feature $x_{(j)} \in \{0, 1\}$ is if word $j$ appears or not

$$\mathbb{P}(x_{(j)}|y) = p_{j,y}^{x_{(j)}}(1 - p_{j,y})^{1-x_{(j)}}$$

$$\sum_{j=1}^{d} \log\left(\frac{p_{j,1}^{x_{(j)}}(1 - p_{j,1})^{1-x_{(j)}}}{p_{j,0}^{x_{(j)}}(1 - p_{j,0})^{1-x_{(j)}}}\right) > \log\left(\frac{\pi_0}{\pi_1}\right)$$

# Example: spam

Each feature is count of a word

More easily each feature $x_{(j)} \in \{0, 1\}$ is if word $j$ appears or not

$$\mathbb{P}(x_{(j)}|y) = p_{j,y}^{x_{(j)}}(1 - p_{j,y})^{1-x_{(j)}}$$

$$\sum_{j=1}^{d} \log\left(\frac{p_{j,1}^{x_{(j)}}(1 - p_{j,1})^{1-x_{(j)}}}{p_{j,0}^{x_{(j)}}(1 - p_{j,0})^{1-x_{(j)}}}\right) > \log\left(\frac{\pi_0}{\pi_1}\right)$$

$$\sum_{j=1}^{d} x_{(j)} \log\left(\frac{p_{j,1}}{1 - p_{j,1}}\frac{1 - p_{j,0}}{p_{j,0}}\right) + \log\left(\frac{1 - p_{j,1}}{1 - p_{j,0}}\right) > \log\left(\frac{\pi_0}{\pi_1}\right)$$

Linear classifier!

# Naive Bayes

Can't actually use in practice. Don't have $\mathbb{P}(x_{(j)}|y)$

Spam example: Estimate $p_{j,y}$ from training data!

Very simple procedure: called plug-in methods

Move back to general optimization

# Finding solutions

- For linear regression we have normal equations

$$\min_{\beta} \frac{1}{2}\|X\beta - y\|_2^2 \quad \min_{\beta} \frac{1}{2} \sum_{i=1}^{n} (x_i^T \beta - y_i)^2$$

$$\nabla \frac{1}{2}\|X\beta - y\|_2^2 = X^T(X\beta - y)$$

set to zero

$$\widehat{\beta} = (X^T X)^{-1} X^T y$$

(can use psuedo-inverse if not invertible)

# Finding solutions with optimization

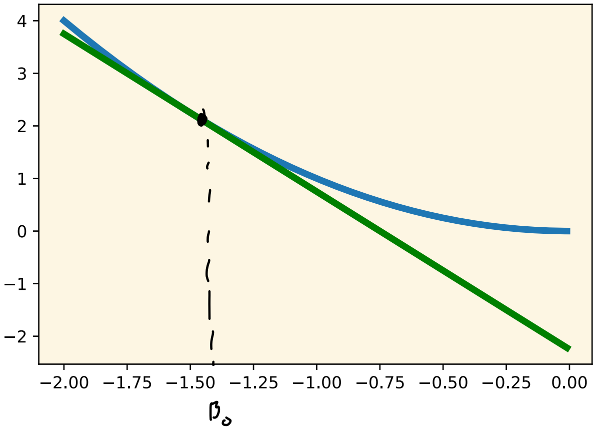
- Let $f(\beta)$ be some convex function
- Goal: solve

$$\arg\min_{\beta} f(\beta)$$

- Eg $f(\beta) = \frac{1}{2n}\|X\beta - y\|_2^2$
- Logistic: $f(\beta) = \frac{1}{n}\sum_i -y_i x_i^T\beta + \log(1 + \exp(x_i^T\beta))$

# Gradients, descents, and recall Taylor

Taylor: $f(\beta) \approx f(\beta_0) + (\beta - \beta_0)^T \nabla f(\beta_0)$ for $\beta$ close to $\beta_0$

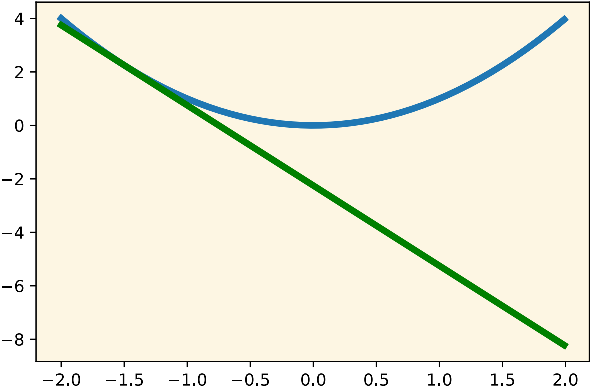


# Gradients, descents, and recall Taylor

Taylor: $f(\beta) \approx f(\beta_0) + (\beta - \beta_0)^T \nabla f(\beta_0)$ for $\beta$ close to $\beta_0$

$\beta = \beta_0 - \eta \nabla f(\beta_0)$

$f(\beta) \approx f(\beta) - \eta \|\nabla f(\beta_0)\|^2$



# Gradient Descent

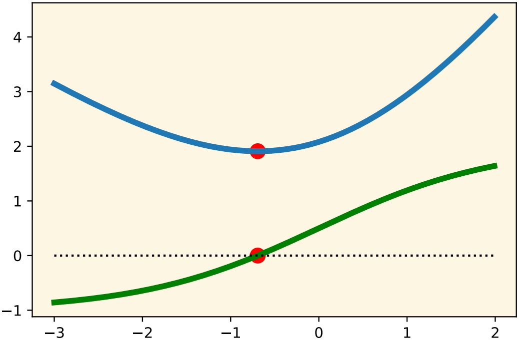
$$\beta_k = \beta_{k-1} - \eta_{k-1} \nabla f(\beta_{k-1})$$

# Newton and Taylor

Remember trying to set $\nabla f(\beta) = 0$

Consider: $f(\beta) = 3\log(1 + \exp(\beta)) - \beta$ $= \sum_{i=1}^{3} \log(1 + \exp(\beta)) - y_i \beta$

$y_1 = 1 \quad y_2 = 0 \quad y_3 = 0$

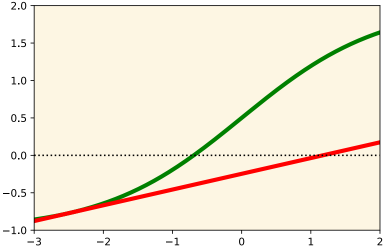


$\nabla f(\beta) \approx \nabla f(\beta_0) + H f(\beta_0)(\beta - \beta_0)$

$H f(\beta)$ is the Hessian

Consider: $f(\beta) = 3 \log(1 + \exp(\beta)) - \beta$

$$f'(\beta) = 3 \frac{\exp(\beta)}{1 + \exp(\beta)} - 1$$

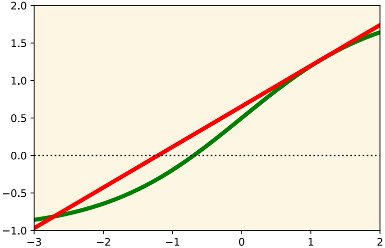


$$\nabla f(\beta) \approx \nabla f(\beta_0) + H\,f(\beta_0)(\beta - \beta_0)$$

$H\,f(\beta)$ is the Hessian

Consider: $f(\beta) = 3\log(1 + \exp(\beta)) - \beta$

$$f'(\beta) = 3\frac{\exp(\beta)}{1 + \exp(\beta)} - 1$$

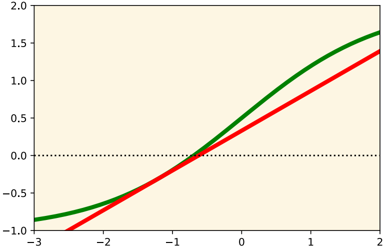


$$\nabla f(\beta) \approx \nabla f(\beta_0) + H f(\beta_0)(\beta - \beta_0)$$

$H f(\beta)$ is the Hessian

Consider: $f(\beta) = 3\log(1 + \exp(\beta)) - \beta$

$$f'(\beta) = 3\frac{\exp(\beta)}{1 + \exp(\beta)} - 1$$

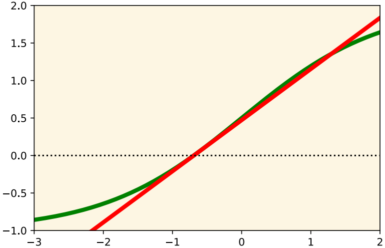


$\nabla f(\beta) \approx \nabla f(\beta_0) + H f(\beta_0)(\beta - \beta_0)$

$H f(\beta)$ is the Hessian

Consider: $f(\beta) = 3 \log(1 + \exp(\beta)) - \beta$

$$f'(\beta) = 3 \frac{\exp(\beta)}{1 + \exp(\beta)} - 1$$



# Newton and Taylor

- $\nabla f(\beta) \approx \nabla f(\beta_0) + H\,f(\beta_0)(\beta - \beta_0)$ $= 0 \Rightarrow \beta = \beta_0 - [H f(\beta_0)]^{-1} \nabla f(\beta_0)$
- $H\,f(\beta)$ is the Hessian

$$[H\,f(\beta_k)]_{ij} = \frac{\partial^2 f(\beta_k)}{\partial \beta_i \partial \beta_j}$$

- Newton's method $\beta_k = \beta_{k-1} - (H\,f(\beta_{k-1}))^{-1} \nabla f(\beta_{k-1})$
- Computational complexity is $O(p^3)$ in general. For large problems impractical.

# Newton's method and Linear Regression

What is it?

$$f(B) = \frac{1}{2} \|XB - y\|_2^2$$

$$\nabla f(B) = X^T(XB - y)$$

$$H(f(B)) = X^T X$$

$$B_K = B_{K-1} - (X^T X)^{-1}(X^T(XB_{K-1} - y))$$

$$= B_{K-1} - (B_{K-1} - (X^T X)^{-1} X^T y)$$

$$= (X^T X)^{-1} X^T y$$

$$f(B) = \frac{1}{2} \sum_{i=1}^{n} (x_i^T B - y_i)^2 \xrightarrow{\nabla} \frac{1}{2} \sum_{i=1}^{n} \nabla (x_i^T B - y_i)^2 \rightarrow \sum_{i=1}^{n} (x_i^T B - y_i) x_i$$

$$\nabla f(B) = \sum_{i=1}^{n} x_i (x_i^T B - y_i) \xrightarrow{\text{Hessian}} \sum_{i=1}^{n} x_i (x_i^T) = X^T X$$

$$[H f(B)]_{(ab)} = \frac{\partial([\nabla f(B)]_a)}{\partial B_{(b)}} = \frac{\partial\left(\sum_{i=1}^{n} x_{i(a)} (x_i^T B - y_i)\right)}{\partial B_{(b)}}$$

$$= \sum_{i=1}^{n} x_{i(a)} \frac{\partial (x_i^T B - y_i)}{\partial B_{(b)}} = \sum_{i=1}^{n} x_{i(a)} x_{i(b)}$$

# Gradient descent

- Used extensively for large scale problems
- $\beta_k = \beta_{k-1} - \eta_{k-1} \nabla f(\beta_{k-1})$
- but "slower" than Newton
- Before looking at figures, what is the update for linear regression?

# Gradient descent and Linear Regression

Recall that $f(\beta) = \frac{1}{2n}\|X\beta - y\|_2^2$

$$\nabla f(B) = \frac{1}{n} X^T (XB - y)$$

$$B_K = B_{K-1} - u_{K-1} \cdot \frac{1}{n}(X^T (XB_{K-1} - y))$$

# Gradient descent and Linear Regression (2)

Recall that $f(\beta) = \frac{1}{2n}\sum_{i=1}^{n}(x_i^T\beta - y_i)^2$

$$B_k = B_{k-1} - \frac{\eta_{k-1}}{n}\sum_{i=1}^{n} x_i \underbrace{(x_i^T B_{k-1} - y_i)}_{\text{error}_i}$$

Exponential families:

$$f(B) = \frac{1}{n}\sum_{i=1}^{n} A(x_i^T B) - y_i x_i^T B$$

recall

$A: \mathbb{R} \to \mathbb{R}$

$$\nabla f(B) = \frac{1}{n}\sum_{i=1}^{n} A'(x_i^T B) x_i - y_i x_i$$

$$= \frac{1}{n}\sum_{i=1}^{n} \underbrace{(A'(x_i^T B) - y_i)}_{\text{error}_i} x_i$$

$$Hf(B) = \frac{1}{n}\sum_{i=1}^{n} A''(x_i^T B) x_i x_i^T$$

$\text{error} \in \mathbb{R}^n$

$\text{error}_i = A'(x_i^T B) - y_i$

$$= \frac{1}{n} X^T(\text{error})$$

Linear: $A(\theta) = \frac{1}{2}\theta^2$

Logistic : $A(\theta) = \log(1 + \exp(\theta))$

$A'(\theta) = \frac{\exp(\theta)}{1+\exp(\theta)}$

# Stochastic Gradient Descent

- gradient descent cycles through all of the data
- large $n$, problematic
- stochastic gradient descent (sgd)

# Stochastic Gradient Descent

- Recall that $f(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (x_i^T \beta - y_i)^2$
- gradient descent is

$$\beta_k = \beta_{k-1} - \eta_{k-1} \nabla f(\beta_{k-1})$$

- For linear regression that becomes

$$\beta_k = \beta_{k-1} - \eta_{k-1} \frac{1}{n} \sum_{i=1}^{n} x_i (x_i^T \beta - y_i)$$

- Let's be lazy and not go through all of the data

# Stochastic Gradient Descent

- Recall that $f(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (x_i^T \beta - y_i)^2$
- gradient descent is

$$\beta_k = \beta_{k-1} - \eta_{k-1} \nabla f(\beta_{k-1})$$

- For linear regression that becomes

$$\beta_k = \beta_{k-1} - \eta_{k-1} \frac{1}{n} \sum_{i=1}^{n} x_i (x_i^T \beta - y_i)$$

- Let's be lazy and not go through all of the data

$$\beta_k = \beta_{k-1} - \eta_{k-1} x_J (x_J^T \beta_{k-1} - y_J)$$

- $J \in [n]$ is uniform random variable
- iterative learning algo applying stochastic gradient descent (sgd)

# Stochastic Gradient Descent

- Recall that $f(\beta) = \frac{1}{2n}\sum_{i=1}^{n}(x_i^T\beta - y)^2$
- We can write $f(\beta) = \frac{1}{n}\sum_{i=i}^{n} f_i(\beta)$

$$f_i(\beta) = \frac{1}{2}(x_i^T\beta - y)^2$$

- So <u>batch</u> gradient descent

$$\beta_k = \beta_{k-1} - \eta_{k-1}\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(\beta_{k-1})$$

- stochastic gradient descent (sgd) is

$$\beta_k = \beta_{k-1} - \eta_{k-1}\nabla f_i(\beta_{k-1})$$

# Stochastic Gradient Descent

- Most general version of sgd
- Goal: $\arg\min_\beta f(\beta)$
- Have access to random vector $\mathbb{E}g_\beta = \nabla f(\beta)$

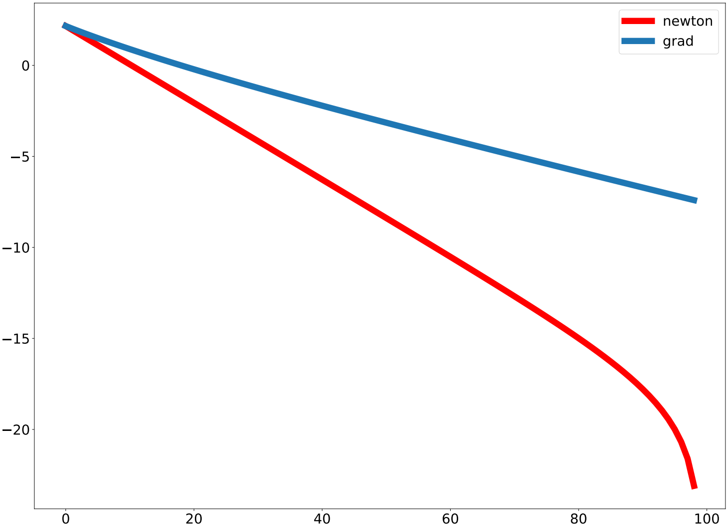
random!

$$\beta_k = \beta_{k-1} - \eta_{k-1}\boxed{g_{\beta_{k-1}}}$$

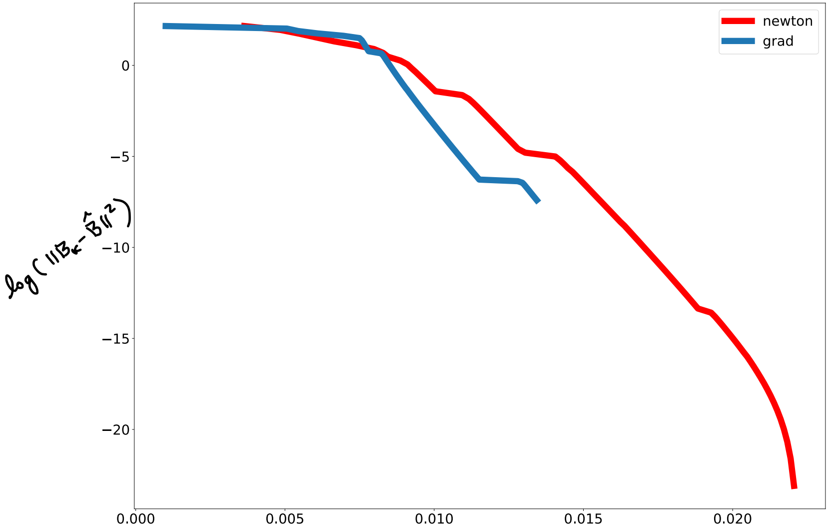
- In the previous slides randomness over $J$ the random example

$$\mathbb{E}\nabla f_J(\beta) = \frac{1}{n}\sum_{j=1}^{n} \nabla(x_j^T\beta - y_j)^2 = \frac{1}{n}\sum_{j=1}^{n} \nabla f_j(\beta)$$
$$= \nabla f(\beta)$$

# Convergence grad v newton



# Convergence grad v newton against time

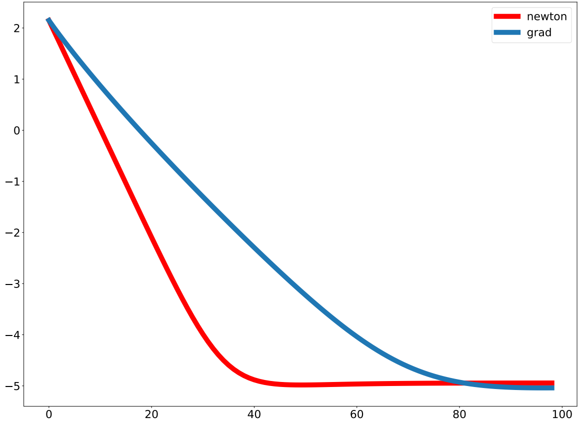


# Convergence grad v newton error to $\beta^*$

Notion of statistical error versus optimization error

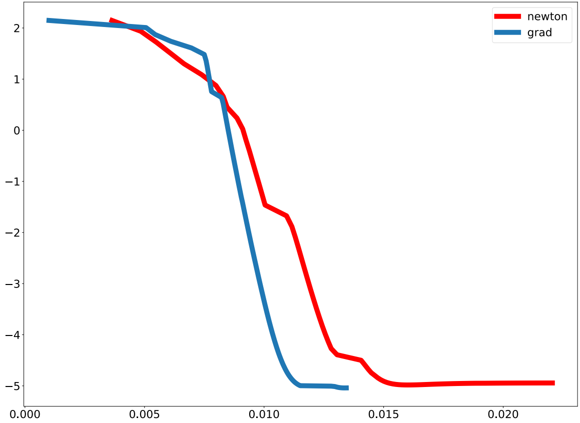
$y = X\beta^* + w$

$\log(\|\beta_k - \beta^*\|_2^2)$

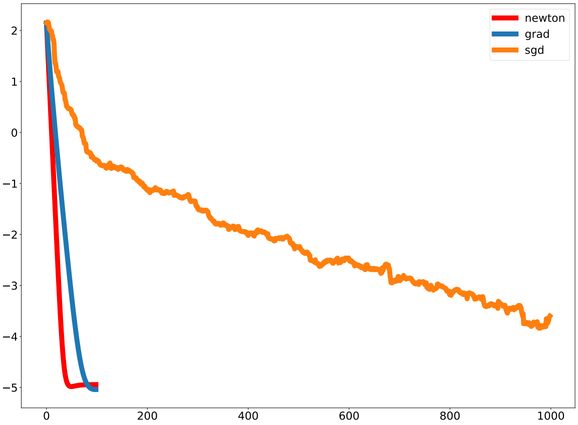


# Convergence grad v newton error to $\beta^*$

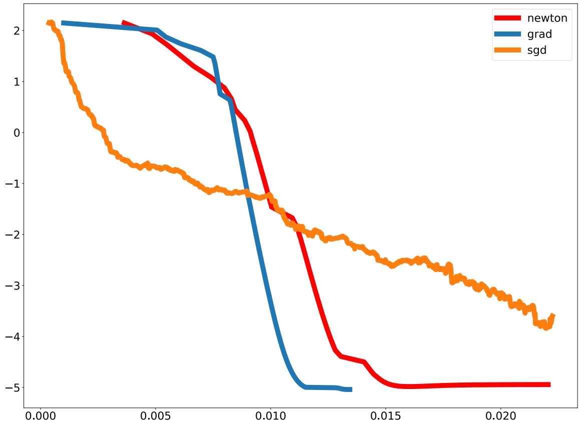
Notion of statistical error versus optimization error vs time



# Convergence grad v newton v sgd error to $\beta^*$



# Convergence grad v newton v sgd error to $\beta^*$



Move to notebook