Distributed Logistic Regression

Main References

Ameet Talwalkar and Henry Chai, **ML with Large Datasets**, CMU

Outline

- Empirical Risk Minimization
- Binary Classification
- Logistic Regression
- Regularized Logistic Regression
- Hyperparameter Tuning
- Multiclass Classification
- Multi-class Logistic Regression
- Distributed Logistic Regression

Empirical Risk Minimization – ERM

- ERM is a common framework for supervised learning
- Given:
  - some labelled training dataset $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$
  - a loss function $l: Y \times Y \to R$
  - a hypothesis class or set of functions F
- The goal is to find

$$\hat{f} = \underset{f \in F}{argmin} \sum_{i=1}^{n} l\left(f\left(x^{(i)}\right), y^{(i)}\right)$$

with the hope that

$$E_{p(x,y)}[l(f(x), y)] \approx \frac{1}{n} \sum_{i=1}^{n} l\left(f\left(x^{(i)}\right), y^{(i)}\right)$$

Empirical Risk Minimization – ERM

- ERM is a common framework for supervised learning
- Given:
  - a labelled training dataset $D = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}, x^{(i)} \in \mathbb{R}^{d+1}$
  - a loss function $l: Y \times Y \to R$
  - a hypothesis class or set of functions F
- The goal is to find

$$\hat{f} = \underset{f \in F}{argmin} \sum_{i=1}^{n} l\left(f\left(x^{(i)}\right), y^{(i)}\right)$$

- Depending on the choice of F and l, this objective function may be convex (easy to optimize) or non-convex (hard)

Binary Classification

- Classification is a type of supervised learning
- Given:
  - a labelled training dataset $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$
  - a loss function $l: Y \times Y \rightarrow R$, where $Y = \{0,1\}$
  - a hypothesis class or set of functions F
- The goal is to find

$$\hat{f} = \underset{f \in F}{argmin} \sum_{i=1}^{n} l\left(f\left(x^{(i)}\right), y^{(i)}\right)$$

- Depending on the choice of F and l, this objective function may be convex (easy to optimize) or non-convex (hard)

Binary Classification with 0/1 Loss

- Classification is a type of supervised learning
- Given:
  - a labelled training dataset $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$
  - a loss function $l(y, y') = \delta(y \neq y')$, for $y, y' \in \{0,1\}$ $\quad \delta(p) = \begin{cases} 1, if\ p\ is\ true \\ 0, if\ p\ is\ false \end{cases}$
  - a hypothesis class or set of functions F
- The goal is to find

$$\hat{f} = \underset{f \in F}{argmin} \sum_{i=1}^{n} l\left(f\left(x^{(i)}\right), y^{(i)}\right)$$

- This loss function is difficult to optimize (non-convex)

A Probabilistic Approach to Binary Classification

- Suppose we have binary labels $y \in \{0,1\}$ and $(d+1)$-dimensional inputs $x = (1, x_1, x_2, \cdots, x_d)^T \in \mathbb{R}^{d+1}$
- Assume

$$P(Y=1|x) = \sigma(\theta^T x) = \frac{1}{1+e^{-\theta^T x}} = \frac{e^{\theta^T x}}{1+e^{\theta^T x}}$$

- This implies two useful facts

$$P(Y=0|x) = 1 - P(Y=1|x) = \frac{1}{1+e^{\theta^T x}}$$

$$\frac{P(Y=1|x)}{P(Y=0|x)} = e^{\theta^T x} \Rightarrow log\frac{P(Y=1|x)}{P(Y=0|x)} = \theta^T x$$

Logistic Function

- Why use logistic function?
  - Differentiable everywhere
  - $\sigma: \mathbb{R} \rightarrow [0,1]$
  - The decision boundary is linear in x

$$y = \begin{cases} 1, & P(Y=1|x) \geq 0.5 \\ 0, & otherwise \end{cases}$$

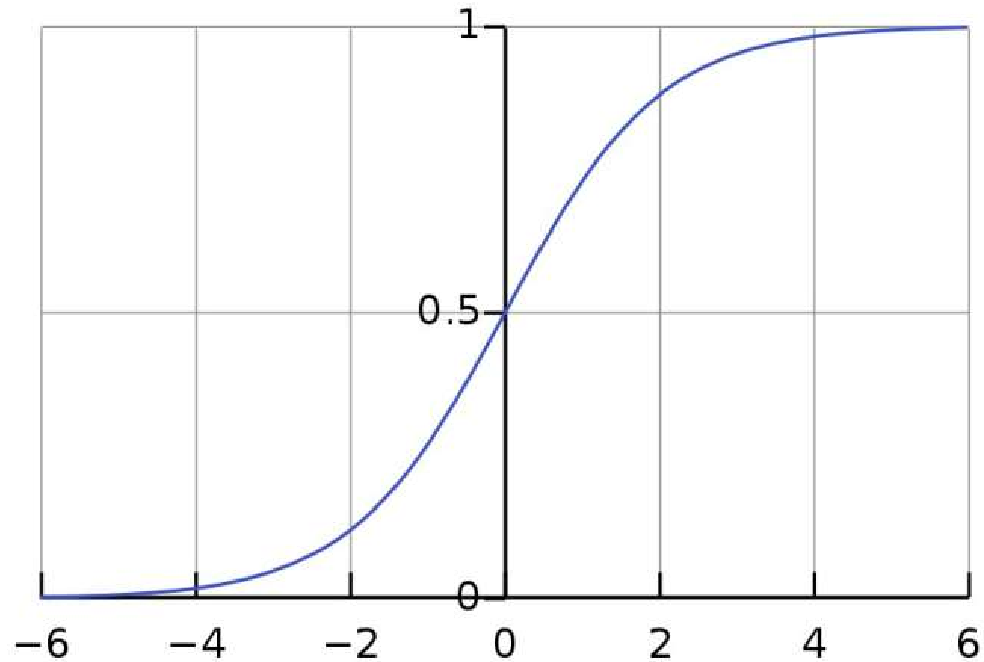
$$P(Y=1|x) = \sigma(\theta^T x) = \frac{1}{1+e^{-\theta^T x}} \geq \frac{1}{2}$$

$$\Rightarrow 2 \geq 1 + e^{-\theta^T x} \Rightarrow 1 \geq e^{-\theta^T x}$$

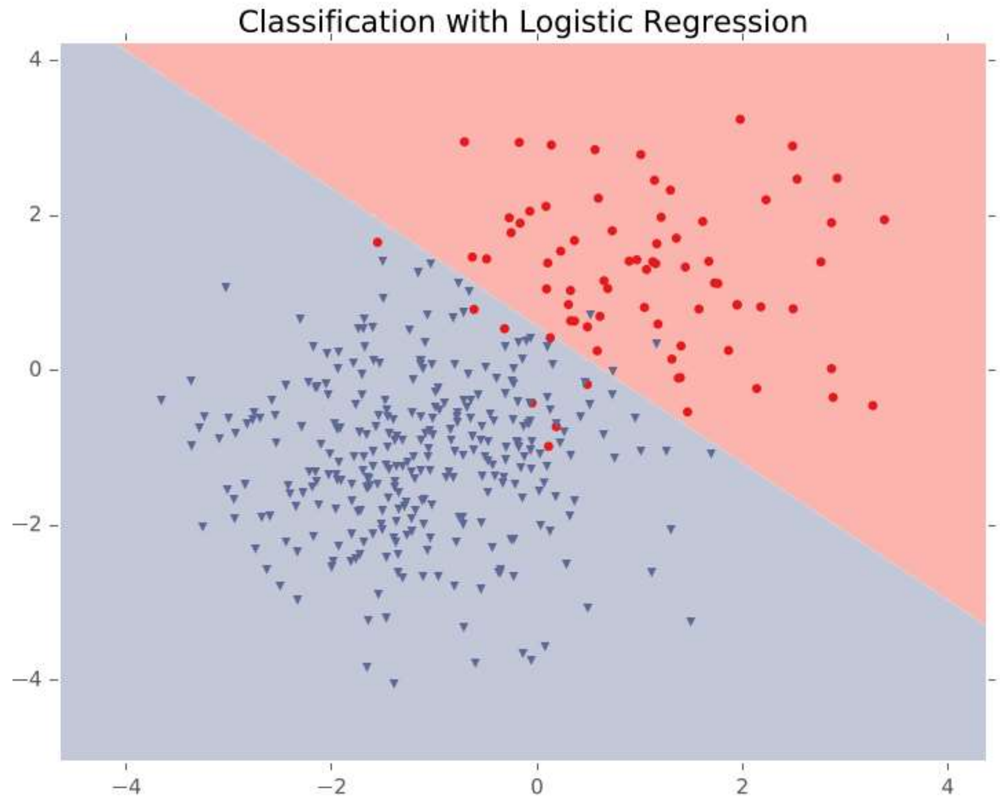
$$\Rightarrow 0 \geq -\theta^T x$$

$$\Rightarrow \theta^T x \geq 0$$

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x} \in \{0,1\}$$



Logistic Regression Decision Boundary



Logistic Regression – Maximum Likelihood (1)

- Goal: find the θ that maximizes the (conditional) probability of the training dataset

$$\prod_{i=1}^{n} P(y^{(i)}|x^{(i)},\theta)$$

- This is equivalent to finding the θ that minimizes the negative log of this probability

$$L_D(\theta) = -log\left(\prod_{i=1}^{n} P\left(y^{(i)}\middle|x^{(i)},\theta\right)\right) = -\sum_{i=1}^{n} log\left(P\left(y^{(i)}\middle|x^{(i)},\theta\right)\right)$$

$$= -\sum_{i=1}^{n} log\left(P\left(Y=1\middle|x^{(i)},\theta\right)^{y^{(i)}} P\left(Y=0\middle|x^{(i)},\theta\right)^{1-y^{(i)}}\right)$$

$$= -\sum_{i=1}^{n} \left(y^{(i)} log\left(\sigma(\theta^T x^{(i)})\right) + (1-y^{(i)}) log\left(1-\sigma(\theta^T x^{(i)})\right)\right)$$

Logistic Regression – Maximum Likelihood (2)

- Classification is a type of supervised learning
- Given:
  - a labelled training dataset $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$
  - a loss function $l(y, y') = \delta(y \neq y')$, for $y, y' \in \{0,1\}$
  - a hypothesis class or set of functions F
- The goal is to find

$$\underset{\theta}{\operatorname{argmin}} - \sum_{i=1}^{n} \left(y^{(i)} log\left(\sigma(\theta^T x^{(i)})\right) + (1 - y^{(i)}) log\left(1 - \sigma(\theta^T x^{(i)})\right) \right)$$

$$= \underset{\theta}{\operatorname{argmin}} - \sum_{i=1}^{n} \left(y^{(i)} log\left(\frac{\sigma(\theta^T x^{(i)})}{1 - \sigma(\theta^T x^{(i)})} \right) + log\left(1 - \sigma(\theta^T x^{(i)})\right) \right)$$

$$= \underset{\theta}{\operatorname{argmin}} - \sum_{i=1}^{n} \left(y^{(i)} \theta^T x^{(i)} - log\left(1 + e^{\theta^T x^{(i)}}\right) \right)$$

Logistic Regression – Maximum Likelihood (3)

$$L_D(\theta) = -\sum_{i=1}^{n}\left(y^{(i)}\theta^T x^{(i)} - log\left(1 + e^{\theta^T x^{(i)}}\right)\right)$$

$$\nabla_\theta L_D(\theta) = -\sum_{i=1}^{n}\left(y^{(i)}\nabla_\theta\left(\theta^T x^{(i)}\right) - \nabla_\theta log\left(1 + e^{\theta^T x^{(i)}}\right)\right)$$

$$\vdots$$

Note: $\frac{\partial\sigma(z)}{\partial z} = \sigma(z)(1 - \sigma(z))$

$$= \sum_{i=1}^{n}\left(\sigma\left(\theta^T x^{(i)}\right) - y^{(i)}\right) x^{(i)}$$

Gradient Descent for Logistic Regression

- Dataset $D = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$

1. Initialize $\theta^{(0)} = 0$ (zero vector) and set $t = 0$
2. While not converged
   - Compute the gradient

$$\nabla_\theta L_D\left(\theta^{(t)}\right) = 2\sum_{i=1}^{n}\left(\sigma\left({\theta^{(t)}}^T x^{(i)}\right) - y^{(i)}\right) x^{(i)}$$

   - Update the weights

$$\theta^{(t+1)} = \theta^{(\mathrm{t})} - \eta\sum_{i=1}^{n}\left(\sigma\left({\theta^{(t)}}^T x^{(i)}\right) - y^{(i)}\right) x^{(i)}$$

   - Increment t: $t = t + 1$

- Output $\theta^{(\mathrm{t})}$

Regularized Logisitic Regression

- Logistic regression

$$L_D(\theta) = \sum_{i=1}^{n} \left(\sigma\left(\theta^T x^{(i)}\right) - y^{(i)}\right) x^{(i)}$$

- Ridge logistic regression (L_2 regularization)

$$\underset{\theta}{\text{minimize}} \left(L_D(\theta) + \frac{\lambda}{2} \|\theta\|_2^2 \right)$$

λ is hyperparameter

- Lasso logistic regression (L_1 regularization)

$$\underset{\theta}{\text{minimize}} \left(L_D(\theta) + \frac{\lambda}{2} \|\theta\|_1 \right)$$

λ is hyperparameter

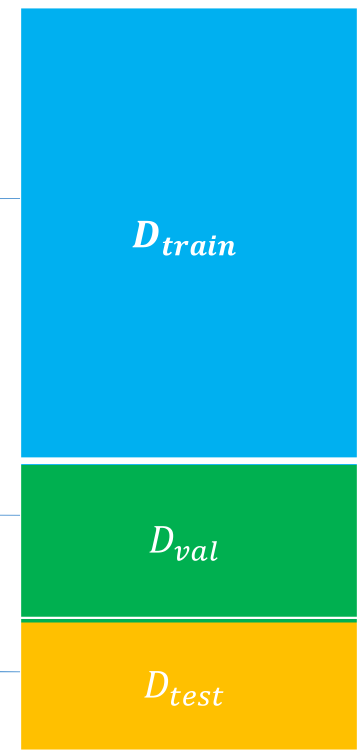
- ElasticNet logistic regression ($L_1 + L_2$regularization)

$$\underset{\theta}{\text{minimize}} \left(L_D(\theta) + \lambda \left(\alpha \|\theta\|_1 + \frac{1-\alpha}{2} \|\theta\|_2^2 \right) \right)$$

λ and α are hyperparameters

Hyperparameter Tuning

- Suppose we want to compare multiple hyperparameter settings $\lambda_1, \lambda_2, \cdots, \lambda_k$
- For $i = 1,2, \ldots, k$
  - Train a model on D_{train} using λ_i
- Evaluate each model on D_{val} and find the best hyperparameter setting, λ_{i^*}
- Compute the error of a model trained with λ_{i^*} on D_{test}



Multi-class Classification

- Classification is a type of supervised learning
- Given:
  - a labelled training dataset $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$
  - a loss function $l: Y \times Y \to R$, where $Y = \{1,2,\cdots,k\}$
  - a hypothesis class or set of functions F
- The goal is to find

$$\hat{f} = \underset{f \in F}{argmin} \sum_{i=1}^{n} l\left(f\left(x^{(i)}\right), y^{(i)}\right)$$

- Depending on the choice of F and l, this objective function may be convex (easy to optimize) or non-convex (hard)

Multi-class Logistic Regression

- Suppose we have k classes $y \in \{1,2,\cdots,k\}$ and $(d+1)$-dimensional inputs $x = (1, x_1, x_2, \cdots, x_d)^T \in \mathbb{R}^{d+1}$
- For each class i, we have a binary classifier with parameter $\theta^{(i)}$
- Then

$$P(Y = i|x) = \frac{e^{\theta^{(i)^T} x}}{\sum_{j=1}^{k} e^{\theta^{(j)^T} x}} = \frac{e^{\theta_0^{(i)} + \theta_1^{(i)} x_1 + \cdots \theta_d^{(i)} x_d}}{\sum_{j=1}^{k} e^{\theta_0^{(j)} + \theta_1^{(j)} x_1 + \cdots \theta_d^{(j)} x_d}}$$

- Assign x to the class that maximizes the (conditional) probability

$$\underset{i}{argmax}\, P(Y = i|x) = \underset{i}{argmax}\, \theta^{(i)^T} x$$

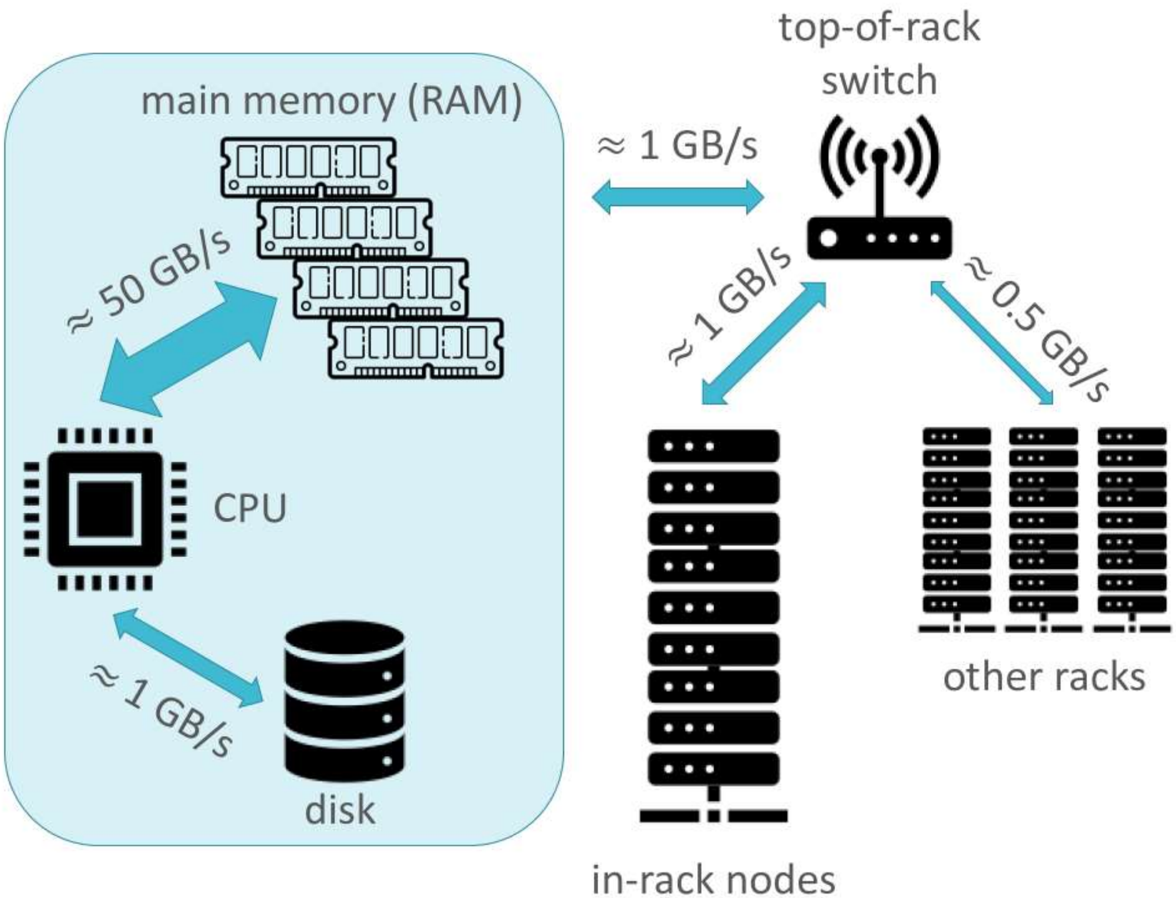
- Multi-class logistic regression is also called multinomial logistic regression or softmax regression

Distributed Gradient Descent for Logistic Regression

Worker	$\begin{bmatrix} \leftarrow & {x^{(1)}}^T & \rightarrow \\ \leftarrow & {x^{(4)}}^T & \rightarrow \\ \vdots & \vdots & \vdots \end{bmatrix}$	$\begin{bmatrix} \leftarrow & {x^{(2)}}^T & \rightarrow \\ \leftarrow & {x^{(3)}}^T & \rightarrow \\ \vdots & \vdots & \vdots \end{bmatrix}$	$\begin{bmatrix} \leftarrow & {x^{(5)}}^T & \rightarrow \\ \leftarrow & {x^{(7)}}^T & \rightarrow \\ \vdots & \vdots & \vdots \end{bmatrix}$	$O(nd)$ distributed storage (total)
Map	$\left(\sigma\left({\theta^{(t)}}^T x^{(i)}\right) - y^{(i)}\right) x^{(i)}$	$\left(\sigma\left({\theta^{(t)}}^T x^{(i)}\right) - y^{(i)}\right) x^{(i)}$	$\left(\sigma\left({\theta^{(t)}}^T x^{(i)}\right) - y^{(i)}\right) x^{(i)}$	$O(nd)$ distributed work (total); $O(d)$ local storage
Reduce	$\theta^{(t+1)} = \theta^{(t)} - \eta \sum_{i=1}^{n} \left(\sigma\left({\theta^{(t)}}^T x^{(i)}\right) - y^{(i)}\right) x^{(i)}$			$O(d)$ local work; $O(d)$ local storage

Reducer send the latest weight vector to worker

Communication Hierarchy



Perform parallel and in-memory computation whenever possible

Minimize network communication

- Need to tradeoff between parallelism and network communication
- Three types of objects that may need to be communicated
  - Data
  - Model
  - Intermediate objects
- Strategies
  - Keep large objects local
  - Reduce the number of iterations

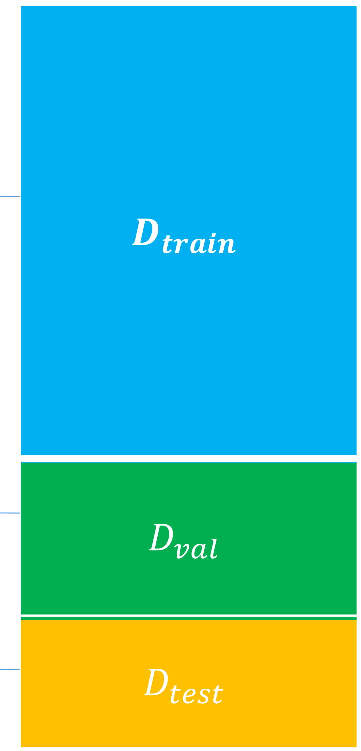
Data Parallel: Compute pointwise gradients locally

Worker	$\begin{bmatrix} \leftarrow & {x^{(1)}}^T & \rightarrow \\ \leftarrow & {x^{(4)}}^T & \rightarrow \\ \vdots & \vdots & \vdots \end{bmatrix}$	$\begin{bmatrix} \leftarrow & {x^{(2)}}^T & \rightarrow \\ \leftarrow & {x^{(3)}}^T & \rightarrow \\ \vdots & \vdots & \vdots \end{bmatrix}$	$\begin{bmatrix} \leftarrow & {x^{(5)}}^T & \rightarrow \\ \leftarrow & {x^{(7)}}^T & \rightarrow \\ \vdots & \vdots & \vdots \end{bmatrix}$	$O(nd)$ distributed storage (total)
Map	$\left(\sigma\left({\theta^{(t)}}^T x^{(i)}\right) - y^{(i)}\right) x^{(i)}$	$\left(\sigma\left({\theta^{(t)}}^T x^{(i)}\right) - y^{(i)}\right) x^{(i)}$	$\left(\sigma\left({\theta^{(t)}}^T x^{(i)}\right) - y^{(i)}\right) x^{(i)}$	$O(nd)$ distributed work (total); $O(d)$ local storage
Reduce	$\theta^{(t+1)} = \theta^{(\mathrm{t})} - \eta \sum_{i=1}^{n} \left(\sigma\left({\theta^{(t)}}^T x^{(i)}\right) - y^{(i)}\right) x^{(i)}$			$O(d)$ local work; $O(d)$ local storage

Reducer send the latest weight vector to worker

Model Parallel: Train each hyperparameter setting on different machine(s)

- Suppose we want to compare multiple hyperparameter settings $\lambda_1, \lambda_2, \cdots, \lambda_k$
- For $i = 1,2, \dots, k$
  - Train a model on D_{train} using λ_i
- Evaluate each model on D_{val} and find the best hyperparameter setting, λ_{i^*}
- Compute the error of a model trained with λ_{i^*} on D_{test}



Summay

1. Computation and storage should be linear in n and *d*
    - For linear regression
        - When *d* is small, distribute matrix computation using outer products
        - When *d* is large, minimize squared error via distributed gradient descent
2. Perform parallel and in-memory computation whenever possible
3. Minimize network communication
    - Data vs model parallelism