ADMM based Scalable Machine Learning on Apache Spark

Sauptik Dhar, Mohak Shah

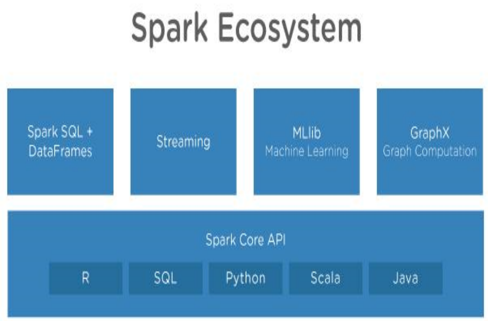
Bosch AI Research

Data is transformational

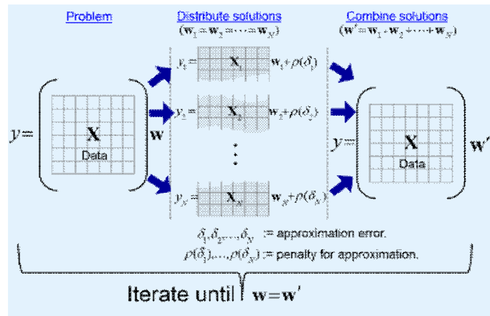
Image: https://www.slideshare.net/mongodb/internet-of-things-and-big-data-vision-and-concrete-use-cases

Big data, Spark and Status-quo

- Challenges
  - Learning → (Convex) Optimization
  - Current solutions (MLLib/ML packages) adopt
    - **SGD:** convergence dependent on step-size, conditionality
    - **LBFGS:** Adapting to non-differentiable functions non-trivial



https://www.simplilearn.com/apache-spark-guide-for-newbies-article



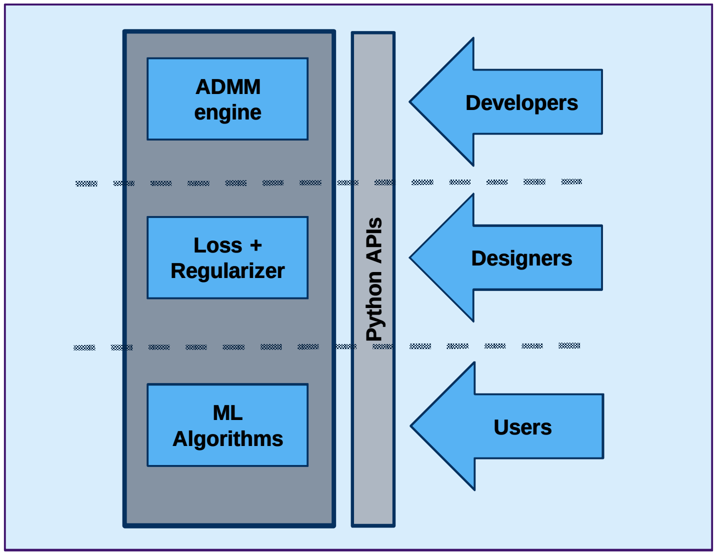
- **ADMM** (**A**lternating **D**irection **M**ethod of **M**ultipliers)
  - Large problem → (simpler) sub-problems
  - Guaranteed convergence and robustness to step-size selection
  - Robust to ill-conditioned problems

ADMM advantages

- ADMM for Spark
- Coverage (ML algorithms)
- Go beyond MLLib/ML for Python community
- Address sub-optimality leading from internal normalization (MLLib/ML)

ADMML Package

- Generic ADMM based formulation: **Coverage** (ML algorithms)
- Robust **Guarantees on Convergence and Accuracy**
- Python API's give **accessibility** to users, developers and designer



Generic ML formulation

- **Given**: training data $(\mathbf{x}_i, y_i)_{i=1}^N$ where $\mathbf{x} \in \mathbb{R}^D$ $y \in \begin{cases} \mathbb{R} \ , & (regression) \\ \{-1,+1\} \ , & (classification) \end{cases}$
- **Solve**: $\boxed{\min_{\mathbf{w},b} \quad L(f_{\mathbf{w},b}(\mathbf{x}), y) \ + \ \lambda R(\mathbf{w})}$ where, $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$

Methods	Loss Function $L(f_{\mathbf{w},b}(\mathbf{x}), y)$	Regularizer $R(\mathbf{w})$
Classification $y \in \{-1,+1\}$		Elastic-net
Logistic Regression	$(1/N)\sum_{i=1}^N \log(1+e^{-y_i(\mathbf{w}^T\mathbf{x}_i+b)})$	$\sum_{j=1}^{D} \delta_j \left\{ \alpha \lvert \mathbf{w}_j \rvert + (1-\alpha)\frac{\mathbf{w}_j^2}{2} \right\}$
LS-SVM	$(1/2N)\sum_{i=1}^N (1-y_i(\mathbf{w}^T\mathbf{x}_i+b))^2$	
Squared-Hinge SVM	$(1/2N)\sum_{i=1}^N (\max(1-y_i(\mathbf{w}^T\mathbf{x}_i+b)))^2$	$\alpha = 0$ (L2-reg)
Regression $y \in \mathbb{R}$		$\alpha = 1$ (L1-reg)
Linear Regression	$(1/2N)\sum_{i=1}^N (y_i-(\mathbf{w}^T\mathbf{x}_i+b))^2$	
Huber	$(1/N)\sum_{i=1}^N \begin{cases} 1/2\,(y_i-(\mathbf{w}^T\mathbf{x}_i+b))^2 \ , \text{if } \lvert y_i-(\mathbf{w}^T\mathbf{x}_i+b)\rvert \le \mu \\ \mu \lvert y_i-(\mathbf{w}^T\mathbf{x}_i+b)\rvert - (1/2)\mu^2 \ , \quad \text{else} \end{cases}$	Group $\sum_{g \in G} \delta_g \lVert \mathbf{w}_g \rVert_2$
Pseudo-Huber	$(1/N)\sum_{i=1}^N \sqrt{\mu^2+(y_i-(\mathbf{w}^T\mathbf{x}_i+b))^2} - \mu$	

ADMM algorithm

$$\min_{\mathbf{w}} \; L(f_{\mathbf{w}}(\mathbf{x}), y) + \lambda R(\mathbf{w}) \equiv \min_{\mathbf{w},\mathbf{z}} \; \boxed{L(f_{\mathbf{w}}(\mathbf{x}), y)} + \boxed{\lambda R(\mathbf{z})} \; \text{s.t.} \; \mathbf{w} = \mathbf{z}$$

Augmented Lagrangian,

$$\ell_{\rho} := L(f_{\mathbf{w}}(\mathbf{x}), y) + \lambda R(\mathbf{z}) + (\rho/2) \left\| \mathbf{w} - \mathbf{z} + \mathbf{u} \right\|^2 + (const)$$

ADMM Steps at each iteration $k+1$

- **w** - update: $\boxed{\mathbf{w}^{k+1} = \arg\min_{\mathbf{w}} \; \mathrm{L}(f_{\mathbf{w}}(\mathbf{x}), y) + (\rho/2) \left\| \mathbf{w} - \mathbf{z}^k + \mathbf{u}^k \right\|^2}$
- **z** - update: $\boxed{\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \; R(\mathbf{z}) + (\rho/2) \left\| \mathbf{w}^{k+1} - \mathbf{z} + \mathbf{u}^k \right\|^2}$
- **u** – update: $\mathbf{u}^{k+1} = \mathbf{u}^k + (\mathbf{w}^{k+1} - \mathbf{z}^k)$

Solve smaller sub-problems which can be easily distributed.

Example (Linear Regression with Elastic-Net)

- Given $(\mathbf{x}_i, y_i)_{i=1}^{N}$ with $\mathbf{x} \in \mathbb{R}^D$ and $y \in \mathbb{R}$
- Solve $\min_{\mathbf{w}} \ \lambda \sum_{j=1}^{D} \delta_j \left\{ \alpha \left| \mathbf{w}_j \right| + (1-\alpha) \frac{\mathbf{w}_j^2}{2} \right\} + \frac{1}{2N} \sum_{i=1}^{N} (y_i - \mathbf{x}_i^T \mathbf{w})^2$
- ADMM updates,

$$\mathbf{w}^{k+1} = \arg\min_{\mathbf{w}} \ \frac{1}{2N} \sum_{i=1}^{N} (y_i - \mathbf{x}_i^T \mathbf{w})^2 + \frac{\rho}{2} \left\| \mathbf{w} - \mathbf{z}^k + \mathbf{u}^k \right\|_2^2$$

$$\boxed{= P^{-1}\mathbf{q}; \quad P = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T + \rho I_D \ \text{and} \ q = \frac{1}{N} \sum_{i=1}^{N} y_i \mathbf{x}_i + \rho(\mathbf{z}^k - \mathbf{u}^k)}$$

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \ \lambda \sum_{j=1}^{D} \delta_j \left\{ \alpha \left| \mathbf{z}_j \right| + (1-\alpha) \frac{\mathbf{z}_j^2}{2} \right\} + \frac{\rho}{2} \left\| \mathbf{w}^{k+1} - \mathbf{z} + \mathbf{u}^k \right\|_2^2$$

$$z_i^{k+1} = \frac{S_{\kappa_j}(\mathbf{w}_j^{k+1} + \mathbf{u}_j^k)}{1 + \lambda \delta_j (1-\alpha) / \rho} \ ; \ \kappa_j = \lambda \delta_j \alpha / \rho \ \text{and} \ S_\kappa = (1 - \frac{\kappa}{|t|})_+ t$$

Example (Logistic Regression with Elastic-Net)

- Given $(\mathbf{x}_i, y_i)_{i=1}^N$ with $\mathbf{x} \in \mathbb{R}^D$ and $y \in \{-1, +1\}$
- Solve $\boxed{\min_{\mathbf{w}} \ \lambda \sum_{j=1}^{D} \delta_j \left\{ \alpha \left| \mathbf{w}_j \right| + (1-\alpha) \frac{\mathbf{w}_j^2}{2} \right\} + \frac{1}{N} \sum_{i=1}^{N} \log(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i})}$
- ADMM updates,

$$\mathbf{w}^{k+1} = \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} \log(1 + e^{-y_i \mathbf{x}_i^T \mathbf{w}}) + \frac{\rho}{2} \left\| \mathbf{w} - \mathbf{z}^k + \mathbf{u}^k \right\|_2^2$$

- Gradient

$$-\frac{1}{N} \sum_i y_i (1 - p_i) \mathbf{x}_i + \rho(\mathbf{w} - \mathbf{z}^k + \mathbf{u}^k)$$

- Hessian

$$\frac{1}{N} \sum_i p_i (1 - p_i) \mathbf{x}_i \mathbf{x}_i^T + \rho I$$

Algorithm 1: Iterative Algorithm for $\mathbf{w}^{k+1}$

Input: $\mathbf{w}^k, \mathbf{z}^k, \mathbf{u}^k$

Output : $\mathbf{w}^{k+1}$

initialize $\mathbf{v}^{(0)} \leftarrow \mathbf{w}^k,\ j \leftarrow 0$

while *not converged* **do**

$$p_i^{(j)} \leftarrow 1 \big/ (1 + e^{-\mathbf{x}_i^T \mathbf{v}^{(j)}})$$

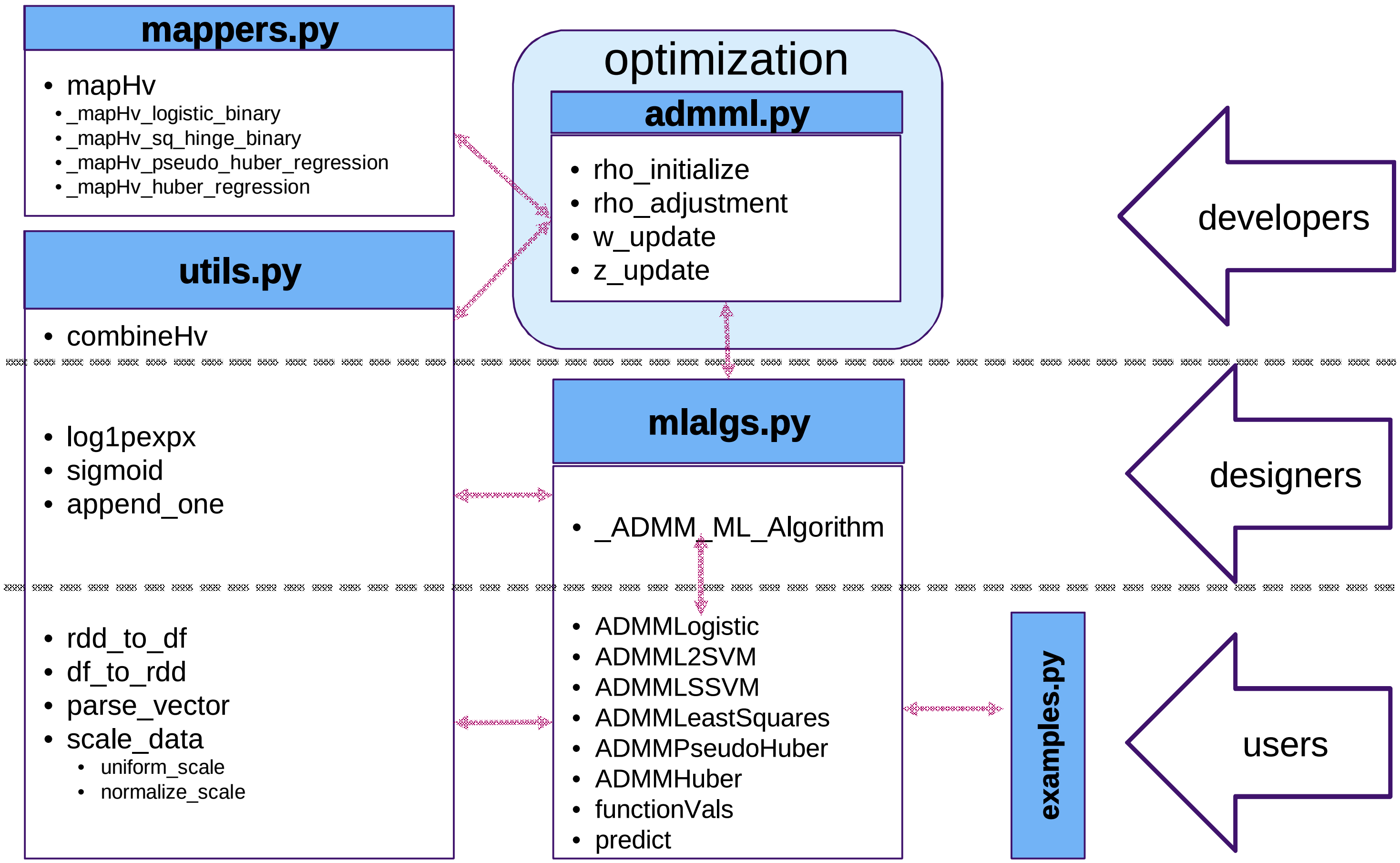
$$P^{(j)} \leftarrow \tfrac{1}{N} \sum_i p_i (1 - p_i) \mathbf{x}\mathbf{x}_i^T + \rho I$$

$$\mathbf{q}^{(j)} \leftarrow -\tfrac{1}{N} \sum_i y_i (1 - p_i) \mathbf{x}_i + \rho(\mathbf{w} - \mathbf{z}^k + \mathbf{u}^k)$$

$$\boxed{\mathbf{v}^{(j+1)} \leftarrow \mathbf{v}^{(j)} - (P^{(j)})^{-1} \mathbf{q}^{(j)}}$$

Research and Technology Center North America | CR/RTC1.3-NA | 5/21/2017

Code Structure (UML diagram)



BOSCH

Experiment (Regression)

System Configuration:

- No. of nodes = 6 (Hortonworks 2.7)
- No. of cores (per node) = 12 (@ 3.20GHz)
- RAM size (per node) = 64 GB.
- Hard disk size (per node) = 2 TB.

Spark Configuration:

- No. of executors = 15.
- Cores per executor = 2.
- Executor memory = 10 GB.
- Driver memory = 2GB.

Data : $\mathbf{x} \in U[-1,1]^{20}$ and $\delta \sim \mathbb{N}(0,1)$

$$y = 5(x_1 + x_2 + x_3) - 1(x_4 + x_5) - 5(x_6 + x_7 + x_8) + 1(x_9 + x_{10}) + \ldots - x_{20} + 2 + \delta$$

- ❑ Small Data: No. of samples = 10000000 (~5GB)
- ❑ Big Data: No. of samples = 100000000 (~50GB)

Experiment (Regression)

Table. Time comparisons ADMM vs. MLLib (in sec)

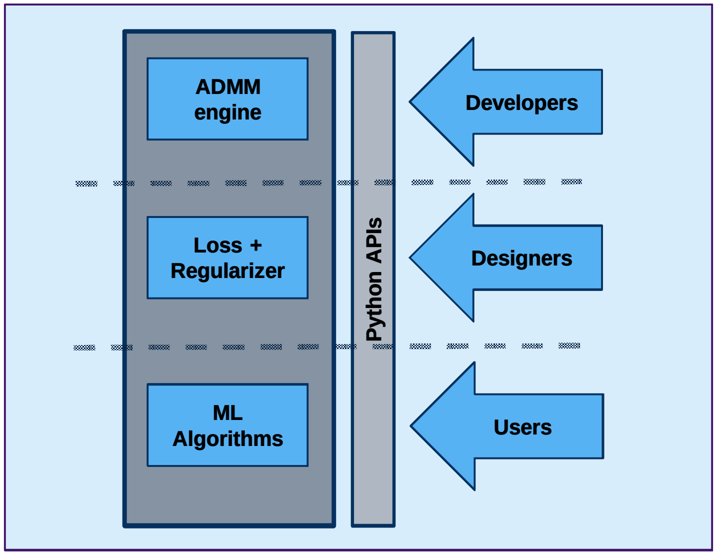
MLLIB (SGD)	ML (L-BFGS)	ADMM
Small Data: No. of samples = 10000000 (~5GB)		
1584.6 (9.6)	290.8 (4.4)*	653.5(0.75) †
Big Data: No. of samples = 100000000 (~50GB)		
—	2597 (8.5)	5811 (7.6)

* convergence in 1 iteration
†convergence in 7 iteration

- ADMM provides competitive computation speeds compared to L-BFGS.
- Initial ρ- selection and advanced adaptive strategies can lead to faster convergence. For example: following [4] convergence in 5 iterations ~ 485 sec.
- For *un-normalized data* ML / MLLib internal normalization may lead to sub-optimal solutions. For example: ML / MLLib : $f_{opt} = 15.07$ and ADMML: $f_{opt} = 12.34$

ADMML Package

- Generic ADMM based formulation: **Coverage** (ML algorithms)
- Robust **Guarantees on Convergence and Accuracy**
- Python API's give **accessibility** to users, developers and designer



https://github.com/DL-Benchmarks/ADMML

Current Algorithms

Classification (Elastic/Group - Regularized)

- Logistic Regression
- LS-SVM
- L2-SVM

Regression (Elastic/Group - Regularized)

- Least Squares (i.e. Ridge Regression, Lasso, Elastic-net, Group-Lasso etc.)
- Huber
- Pseudo-Huber

Future Roadmap

- Initial step-size selection.
- Consensus ADMM (coverage for various discontinuous loss functions, like SVMs, alpha- trimmed functions etc.)
- Multiclass algorithms (like, multinomial logistic regression, C&S-SVM etc.)

Contributors: Sauptik Dhar, Naveen Ramakrishnan, Jeff Irion, Jiayi Liu, Unmesh Kurup, Mohak Shah

Please cite: Dhar S, Yi C, Ramakrishnan N, Shah M. ADMM based scalable machine learning on Spark. IEEE Big Data 2015.