Distributed linear SVM with the Alternating Direction Method of Multipliers

Raoul Lefmann

TU Dortmund

6. Juli 2016

Outline

Literature

- S. Boyd, N. Parikh, E. Chu, B. Peleato, und J. Eckstein.
  Distributed optimization and statistical learning via the alternating direction method of multipliers.
  Found. Trends Mach. Learn., 3(1):1–122, Jan. 2011.

- C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, und S. Sundararajan.
  A dual coordinate descent method for large-scale linear svm.
  In *Proceedings of ICML 2008*, Seiten 408–415, New York, NY, USA, 2008.

- C. Zhang, H. Lee, und K. G. Shin.
  Efficient distributed linear classification algorithms via the alternating direction method of multipliers.
  In *Proceedings of AISTATS 2012*, Seiten 1398–1406, 2012.

Support vector machines

- Binary classification problem
- Dataset $\mathcal{D} = \{(x_i, y_i) \mid x_i \in \mathbb{R}^d, y_i \in \{-1, +1\}, i = 1, \ldots, m\}$
- Find hyperplane $H = \{x \mid w^T x + b = 0\}$ that separates classes with maximum margin
- Incorporate b into w:

$$x_i^T \leftarrow [1, x_i^T] \qquad w^T \leftarrow [b, w^T] \qquad d \leftarrow d + 1$$

- SVM can be formulated as an unconstrained optimization problem:

$$\min_w \frac{1}{2}\|w\|_2^2 + C \sum_{i=1}^{m} \ell(w, x_i, y_i)$$

ℓ is a loss function

SVM: loss functions

- Hinge loss (L1-SVM)

$$\min_{w} \frac{1}{2}\|w\|_2^2 + C \sum_{i=1}^{m} \max\left\{0, 1 - y_i w^T x_i\right\}$$

- Squared hinge loss (L2-SVM)

$$\min_{w} \frac{1}{2}\|w\|_2^2 + C \sum_{i=1}^{m} \max\left\{0, 1 - y_i w^T x_i\right\}^2$$

Problem: non-smooth 😐

SVM: constrained formulation (L2-SVM)

$$
\begin{aligned}
\min_{w,\xi} \quad & \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{m} \xi_i^2 & \\
s.t. \quad & y_i w^T x_i \geq 1 - \xi_i & i = 1, \ldots, m \\
& \xi_i \geq 0 & i = 1, \ldots, m
\end{aligned}
$$

Equivalent to unconstrained formulization since at solution $y_i w^T x_i = 1 - \xi_i$ and therefore $\xi_i = \max\{0, 1 - y_i w^T x_i\}$

ADMM

- Framework for distributed optimization
- Optimization problems of type

$$\begin{aligned} \min_{w,z} \quad & f(w) + g(z) \\ s.t. \quad & Aw + Bz = c \end{aligned}$$

- Uses augmented Lagrangian:

$$\mathscr{L}_\rho(w, z, \lambda) = f(w) + g(z) + \lambda^T(Aw + Bz - c) + \frac{\rho}{2}\|Aw + Bz - c\|_2^2$$

- Update steps:

$$\begin{aligned} w &\leftarrow \underset{w}{\operatorname{argmin}}\, \mathscr{L}_\rho(w, z, \lambda) && (1) \\ z &\leftarrow \underset{z}{\operatorname{argmin}}\, \mathscr{L}_\rho(w, z, \lambda) && (2) \\ \lambda &\leftarrow \lambda + \rho(Aw + Bz - b) && (3) \end{aligned}$$

ADMM: Global variable consensus

- Suppose the objective has the form:

$$\min_{w} f(w) = \sum_{i=1}^{N} f_i(w)$$

- We can compute the f_i in parallel, but sharing w causes a lot of communication
- Solution: Create local copies $w_1, \ldots, w_N$ of w in each node. Consensus is obtained by a variable z.

$$\begin{aligned} \min_{w,z} \quad & \sum_{i=1}^{N} f_i(w_i) \\ s.t. \quad & w_i - z = 0 \quad i = 1, \ldots, N \end{aligned}$$

ADMM: Global variable consensus

$$
\begin{aligned}
w_i &\leftarrow \operatorname*{argmin}_{w_i} \left\{ f_i(w_i) + \lambda_i^T (w_i - z) + \frac{\rho}{2} \|w_i - z\|_2^2 \right\} \\
z &\leftarrow \bar{w} + \frac{1}{\rho} \bar{\lambda} \\
\lambda_i &\leftarrow \lambda_i + \rho(w_i - z)
\end{aligned}
$$

- w-update and λ-update can be computed in parallel
- z-update has nice closed form solution

ADMM: Consensus + Regularization

- Add a regularizer on the consensus variable $g(z)$ to the problem:

$$\begin{aligned} \min_{w,z} \quad & g(z) + \sum_{i=1}^{B} f_i(w_i) \\ s.t. \quad & w_i - z = 0 \quad i = 1, \ldots, B \end{aligned}$$

- z-update changes. In general not so nice anymore ☹

Consensus SVM

- Assume the dataset $\mathcal{D}$ is split across N nodes in a network. Let $B_i = \{j \mid (x_j, y_j) \in \mathcal{D} \text{ is stored in node } i\}$.
- Reformulate SVM as a consensus problem:

$$\begin{aligned} \min_{w,z} \quad & \frac{1}{2}\|z\|_2^2 + C \sum_{i=1}^{N} \sum_{j \in B_i} \max\{0, 1 - y_j \langle w_i, x_j \rangle\}^2 \\ s.t. \quad & w_i - z = 0 \quad i = 1, \ldots, N \end{aligned}$$

- Each node learns its own local w_i, consensus is reached via z
- This is exactly consensus + regularization!

Consensus SVM: ADMM updates

Augmented Lagrangian:

$$\mathscr{L}_\rho(w, z, \lambda) = \frac{1}{2}\|z\|_2^2 + C \sum_{i=1}^{N} \sum_{j \in B_i} \max\{0, 1 - y_j \langle w_i, x_j \rangle\}^2$$

$$+ \sum_{i=1}^{N} \left[\frac{\rho}{2} \|w_i - z\|_2^2 + \lambda_i^T (w_i - z) \right]$$

$$w_i \leftarrow \operatorname*{argmin}_{w_i} \mathscr{L}_\rho(w, z, \lambda)$$

$$= \operatorname*{argmin}_{w_i} C \sum_{j \in B_i} \max\{0, 1 - y_j \langle w_i, x_j \rangle\}^2 + \frac{\rho}{2} \|w_i - z\|_2^2 + \lambda_i (w_i - z)$$

$$z \leftarrow \operatorname*{argmin}_{z} \mathscr{L}_\rho(w, z, \lambda)$$

(see next slide)

$$\lambda_i \leftarrow \lambda_i + \rho(w_i - z)$$

Consensus SVM: z-update

z-update has a closed form solution ☺

$$\frac{\partial}{\partial z}\mathscr{L}_\rho(w,z,\lambda) = \frac{\partial}{\partial z}\frac{1}{2}\|z\|_2^2 + \frac{\rho}{2}\sum_{i=1}^{N}\frac{\partial}{\partial z}\|w_i - z\|_2^2 - \sum_{i=1}^{N}\frac{\partial}{\partial z}\lambda_i^T z \overset{!}{=} 0$$

$$z + \frac{\rho}{2}\sum_{i=1}^{N}(2z - 2w_i) - \sum_{i=1}^{N}\lambda_i = 0$$

$$(1+\rho N)z - \sum_{i=1}^{N} w_i - \sum_{i=1}^{N}\lambda_i = 0$$

$$\boxed{z = \frac{\sum_{i=1}^{N}(w_i + \lambda_i)}{1+\rho N}}$$

Consensus SVM: Reformulation

We can set $\mu_i = \frac{\lambda_i}{\rho}$ to obtain a simpler formulation:

$$w_i \leftarrow \underset{w_i}{\operatorname{argmin}}\, C \sum_{j \in B_i} \max\left\{0, 1 - y_j \langle w_i, x_j \rangle\right\}^2 + \frac{\rho}{2} \|w_i - z - \mu_i\|_2^2$$

$$z \leftarrow \frac{\sum_{i=1}^{N} (w_i + \mu_i)}{N + 1/\rho}$$

$$\mu_i \leftarrow \mu_i + w_i - z$$

The w-update

- We need to find a way to compute w-update efficiently

$$\operatorname*{argmin}_{w_i} \frac{\rho}{2}\|w - v\|_2^2 + C\sum_{j=1}^{s} \max\{0, 1 - y_j w_i^T x_j\}^2$$

where $(x_1, y_1), \ldots, (x_s, y_s)$ are data on machine i and $v = z - \mu_i$

- Equivalent constrained problem:

$$\begin{aligned} \min_{w,\xi} \quad & \frac{\rho}{2}\|w - v\|_2^2 + C\sum_{i=1}^{s} \xi_i^2 & \\ s.t. \quad & y_i w^T x_i \geq 1 - \xi_i & i = 1, \ldots, s \\ & \xi_i \geq 0 & i = 1, \ldots, s \end{aligned}$$

Convex optimization problem with linear constraints

The $\mathbf{w}$-update: duality

- Slater's condition holds: Choosing $w = 0$ and $\xi_i > 1$ gives a strictly feasible point. The duality gap is zero! ☺
- Dual:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2\rho}\alpha^T(Q+D)\alpha - b^T\alpha \\ s.t. \quad & \alpha_i \geq 0 \qquad\qquad\qquad\qquad i = 1, \ldots, s \end{aligned}$$

- $\alpha \in \mathbb{R}^s$
- $Q_{ij} = y_i y_j x_i^T x_j$
- D: diagonal matrix with $D_{ii} = \frac{\rho}{2C}$
- $b = [1 - y_1 v^T x_1, \ldots, 1 - y_s v^T x_s]^T$

The $\mathbf{w}$-update: dual coordinate descent

- Outer loop: update α in each iteration
- Inner loop: update each α_i separately
- Optimize one α_i at a time and then circularly move to the next variable
- The optimization for α_i has a closed form solution! ☺

The w-update: updating α_i

- Consider partial derivative w.r.t. α_i:

$$\begin{aligned}\nabla_i &\stackrel{\text{def}}{=} \frac{\partial}{\partial \alpha_i}\left[\frac{1}{2\rho}\alpha^T(Q+D)\alpha - b^T\alpha\right] \\ &= \frac{1}{\rho}\sum_{j=1}^{s}\alpha_j(Q+D)_{ij} - b_i\end{aligned}$$

- Setting $\nabla_i = 0$ and solving for α_i obtains $\tilde{\alpha}_i$.

$$\begin{aligned}\frac{1}{\rho}\sum_{j=1}^{s}\alpha_j(Q+D)_{ij} - b_i &\stackrel{!}{=} 0 \\ \frac{1}{\rho}\alpha_i(Q+D)_{ii} + \frac{1}{\rho}\sum_{j\neq i}\alpha_j(Q+D)_{ij} - b_i &= 0\end{aligned}$$

$$\boxed{\tilde{\alpha}_i = \frac{\rho b_i - \sum_{j\neq i}\alpha_j(Q+D)_{ij}}{(Q+D)_{ii}}}$$

- The updated α_i is $\tilde{\alpha}_i$ projected onto $[0, \infty)$, since $\alpha_i \geq 0$
- Rewriting the problem makes it similar to gradient descent:

$$\begin{aligned} \alpha_i &\leftarrow \max\{0, \tilde{\alpha}_i\} \\ &= \max\left\{0, \alpha_i - \frac{\rho}{(Q+D)_{ii}} \nabla_i \right\} \end{aligned}$$

- Each update step needs only the i-th row of $(Q + D)$
- Takes $\mathcal{O}(s)$ to calculate the partial derivative in each iteration

The w-update: sparsity

- The w-update can be made more efficient for sparse data
- Setting the partial derivative w.r.t. w of the Lagrangian of the primal problem to zero yields

$$w = v + \frac{1}{\rho} \sum_{j=1}^{s} \alpha_j y_j x_j$$

- Lets have another look at ∇_i:

$$\begin{aligned} \nabla_i &= \frac{1}{\rho} \sum_{j=1}^{s} \alpha_j (Q + D)_{ij} - b_i \\ &= \frac{1}{\rho} \sum_{j=1}^{s} \alpha_j y_i y_j x_i^T x_j + \underbrace{\frac{1}{\rho} \sum_{j=1}^{s} \alpha_j D_{ij}}_{=\alpha_i D_{ii}} - (1 - y_i v^T x_i) \end{aligned}$$

$$
\begin{aligned}
\nabla_i &= \frac{1}{\rho} y_i \left[\sum_{j=1}^{s} \alpha_j y_j x_j \right] x_i + \alpha_i D_{ii} - 1 + y_i v^T x_i \\
&= y_i \left[v + \frac{1}{\rho} \sum_{j=1}^{s} \alpha_j y_j x_j \right] x_i + \alpha_i D_{ii} - 1 \\
&= y_i w^T x_i + \alpha_i D_{ii} - 1
\end{aligned}
$$

- The main cost is created by computing $w^T x_i$
- If we save x_i in a sparse form the computation cost is $\mathcal{O}(\bar{n})$ where $\bar{n}$ is the average number of non-zero features
- Updating w^T also takes time $\mathcal{O}(\bar{n})$

Project details

- Implementation of the method in Julia
- Show correctness by comparing results with JuMP solution on toy data
- If we can run it on a cluster we might see some speedup results ☺
- Realistic application: probably something like spam classification

Questions

- How about other loss functions? Non-squared Hinge loss works, but what about for example SVR?
- Generalization to non-linear case possible?
- Do we have to update all α_i in each iteration of outer loop? How about online setting?
- Effect of small tweaks:
  - Random permutation of data such that label distribution is similar in all nodes
  - Random permutation of order of α_i updates
  - Starting point of α optimization and stopping criterion (inexact minimization)