Efficient Projections onto the $L_1$ Ball for Learning in High Dimensions

JOHN DUCHI (GOOGLE, UCB)
SHAI SHALEV-SHWARTZ (TTI)
YORAM SINGER (GOOGLE)
TUSHAR CHANDRA (GOOGLE)
THE HIGHER MINIMUM WAGE THAT WAS SIGNED INTO LAW ... WILL BE WELCOME RELIEF OF WORKERS ... THE 90 CENT-AN-HOUR INCREASE...
THE HIGHER MINIMUM WAGE THAT WAS SIGNED INTO LAW ... WILL BE WELCOME RELIEF OF WORKERS ... THE 90 CENT-AN-HOUR INCREASE... REGULATIONS LABOUR ECONOMICS
THE HIGHER MINIMUM WAGE THAT WAS SIGNED INTO LAW ... WILL BE WELCOME RELIEF OF WORKERS ... THE 90 CENT-AN-HOUR INCREASE...
• Common approach to topic classification:
  • Select relevant features / tokens
  • Assign weights to tokens in order to achieve low classification error rate
Feature Selection & Learning

THE HIGHER MINIMUM WAGE THAT WAS SIGNED INTO LAW ... WILL BE WELCOME RELIEF OF WORKERS ... THE 90 CENT-AN-HOUR INCREASE...

REGULATIONS LABOUR ECONOMICS

- Common approach to topic classification:
  - Select relevant features / tokens
  - Assign weights to tokens in order to achieve low classification error rate
- Alternative approach: impose $L_1$ domain constraints
\[ \ell_1 \text{ Domain Constraints} \]

\[
\min_{w} L(w) \quad \text{s.t.} \quad \|w\|_1 \leq z
\]
\[ \min_{w} L(w) \quad \text{s.t.} \quad \|w\|_1 \leq z \]
\[ \ell_1 \text{ Domain Constraints} \]

\[ \min_{w} L(w) \text{ s.t. } \|w\|_1 \leq z \]

Why \( \ell_1 \)?
\[ \min_{\mathbf{w}} L(\mathbf{w}) \quad \text{s.t.} \quad \|\mathbf{w}\|_1 \leq z \]

Why \( \ell_1 \)?
\[ \min_{\mathbf{w}} L(\mathbf{w}) \quad \text{s.t.} \quad \|\mathbf{w}\|_1 \leq z \]

Why \( \ell_1 \)?

The solution is likely to have numerous zero components (Candes’06, Donoho’06)

“CORNER”
\[ \min_{w} L(w) \text{ s.t. } \|w\|_1 \leq z \]
\[
\min_{\mathbf{w}} L(\mathbf{w}) \text{ s.t. } \|\mathbf{w}\|_1 \leq z
\]

\[
\mathbf{w}_{t+1} = \Pi_X (\mathbf{w}_t - \eta_t \nabla_t L)
\]

\[
\Pi_X (\mathbf{w}) = \arg \min \{ \|\mathbf{w} - \mathbf{v}\| \mid \mathbf{v} \in X \}
\]

\[
X = \{ \mathbf{w} \mid \|\mathbf{w}\|_1 \leq z \} 
\]
Gradient Projection with $\ell_1$
Gradient Projection with $\ell_1$
Gradient Projection with $\ell_1$
Gradient Projection with $\ell_1$
Gradient Projection with $\ell_1$

Focus on efficient algorithms for Euclidean projections onto the $\ell_1$ ball in high dimensions.
Projection onto $\ell_1$ Ball
Projection onto $\ell_1$ Ball
Projection onto $\ell_1$ Ball
Projection onto $\ell_1$ Ball
Projection onto $\ell_1$ Ball

$v_1 := v_1 - \theta$

$v_2 := v_2 - \theta$
Projection onto $\ell_1$ Ball
Projection onto $\ell_1$ Ball
Projection onto $\ell_1$ Ball
Projection onto $\ell_1$ Ball

$v_1 := \max\{0, v_1 - \theta\}$

$v_2 := \max\{0, v_2 - \theta\}$
Projection onto $\ell_1$ Ball (cont)
\[ \text{sign}(v_j) \max \{0, |v_j| - \theta\} \]
Algebraic-Geometric View
Algebraic-Geometric View

$\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$, $\mathbf{v}_4$, $\mathbf{v}_5$, $\mathbf{v}_6$, $\mathbf{v}_7$
Algebraic-Geometric View
\[(v_1 - \theta) + (v_2 - \theta) + (v_4 - \theta) + (v_5 - \theta) = z\]
\[ (v_1 - \theta) + (v_2 - \theta) + (v_4 - \theta) + (v_5 - \theta) = z \]

\[ \Rightarrow \quad \theta = \frac{v_1 + v_2 + v_4 + v_5 - z}{4} \]
\[ (v_1 - \theta) + (v_2 - \theta) + (v_4 - \theta) + (v_5 - \theta) = z \]

\[ \Rightarrow \theta = \frac{v_1 + v_2 + v_4 + v_5 - z}{4} \]
Chicken and Egg Problem
• Had we known the threshold we could have found all the zero elements
• Had we known the threshold we could have found all the zero elements

• Had we known the elements that become zero we could have calculated the threshold
If $v_j < v_k$ then if after the projection the $k$’th component is zero, the $j$’th component must be zero as well.
If $v_j < v_k$ then if after the projection the $k$’th component is zero, the $j$’th component must be zero as well.
If $v_j < v_k$ then if after the projection the $k$’th component is zero, the $j$’th component must be zero as well.
If $v_j < v_k$ then if after the projection the $k$’th component is zero, the $j$’th component must be zero as well.
If $v_j < v_k$ then if after the projection the $k$’th component is zero, the $j$’th component must be zero as well.

\[
\begin{align*}
\theta & \quad v_3 \\
& \quad v_6
\end{align*}
\]
If two feasible solutions exist with $k$ and $k+1$ non-zero elements then the solution with $k+1$ elements attains a lower loss.
If two feasible solutions exist with $k$ and $k+1$ non-zero elements then the solution with $k+1$ elements attains a lower loss.
• Sort vector to be projected

\[ \Rightarrow \mu_1 \geq \mu_2 \geq \mu_3 \geq \ldots \geq \mu_n \]

• If \( j \) is a feasible index then

\[ \mu_j > \theta \Rightarrow \mu_j > \frac{1}{j} \left( \sum_{r=1}^{j} \mu_r - z \right) \]

• Number of non-zero elements \( \rho \)

\[ \rho = \max \left\{ j : \mu_j - \frac{1}{j} \left( \sum_{r=1}^{j} \mu_r - z \right) > 0 \right\} \]
Calculating Projection (G)
Calculating Projection (G)
Calculating Projection (G)

\[ v_4 - (v_4 - z) > 0 \]
Calculating Projection (G)

\[ v_5 - \frac{1}{2}(v_4 + v_5 - z) > 0 \]
\[ \rho = 3 \quad \theta = \frac{1}{3} (v_2 + v_4 + v_5 - z) \]
\[ \rho = 3 \quad \theta = \frac{1}{3} (v_2 + v_4 + v_5 - z) \]
Calculating Projection (G)

\[ \rho = 3 \quad \theta = \frac{1}{3} \left( v_2 + v_4 + v_5 - z \right) \]
Efficient Projection Alg.

- Assume we know number of elements greater than $v_j$
  \[ \rho(v_j) = |\{v_i : v_i \geq v_j\}| \]

- Assume we know the sum of elements greater than $v_j$
  \[ s(v_j) = \sum_{i: v_i \geq v_j} v_i \]

- Then, we can check in constant time the status of $v_j$
  \[ v_j > \theta \iff v_j > \frac{1}{\rho(v_j)} (s(v_j) - z) \iff s(v_j) - \rho(v_j)v_j < z \]

- Randomized median-like search [$O(n)$ instead $O(n \log(n))$]
• In many applications the dimension is very high
  [text application: 2 million tokens]
• In many applications the dimension is very high
  [text application: 2 million tokens]

• Small number of non-zero elements in each example
  [text application: \(\sim\) thousand tokens per document]
In many applications the dimension is very high
[ text application: 2 million tokens]

Small number of non-zero elements in each example
[ text application: ~ thousand tokens per document ]

Online/stochastic updates only modify the weights corresponding to non-zero features in example
• In many applications the dimension is very high
  [ text application: 2 million tokens ]

• Small number of non-zero elements in each example
  [ text application: ~ thousand tokens per document ]

• Online/stochastic updates only modify the weights
  corresponding to non-zero features in example

• Goals:
  • linear time in the number of non-zero features
  • sub-linear in the full dimension
• Use red-black (RB) tree to store only the non-zero components of the weight vector. Non-zero components are stored w/o global shift $\Theta_t = \sum_{s \leq t} \theta_t$

• Each online/stochastic update deletes & then inserts non-zero elements of an example in $O(k \log(n))$ time

• Store in each node of RB additional information that facilitates efficient search for “pivot” $\theta_t$

• Upon projection, removal of a whole sub-tree is performed in logarithmic time using Tarajan’s (83) algorithm for splitting RB tree
RB Tree for Efficient Proj.
RB Tree for Efficient Proj.

VALUE  # ELEMENTS IN RIGHT SUB-TREE  SUM ELEMENTS IN RIGHT SUB-TREE

2 1 2  6 1 6  8 1 8  13 1 13

5 2 11  12 2 25
**Pivot Search with RB Tree**

**Procedure PivotSearch**($\mathcal{T}, v, \rho, s$)

Compute $\hat{\rho} = \rho + r(v)$; $\hat{s} = s + \sigma(v)$

**IF** $\hat{s} < v\hat{\rho} + z$  **//** $v \geq \text{pivot}$

**IF** $v^* > v$  **THEN** $v^* = v$; $\rho^* = \hat{\rho}$; $s^* = \hat{s}$

**IF** leaf$_\mathcal{T}(v)$  **RETURN** $\theta = (s^* - z)/\rho^*$

**CALL** PivotSearch(left$_\mathcal{T}(v)$, $\hat{\rho}$, $\hat{s}$)

**ELSE**  **//** $v < \text{pivot}$

**IF** leaf$_\mathcal{T}(v)$  **RETURN** $\theta = (s^* - z)/\rho^*$

**CALL** PivotSearch(right$_\mathcal{T}(v)$, $\rho$, $s$)

**ENDIF**

**END**
Pivot Search with RB Tree
Experimental Results

• Losses:
  • squared error
  • logistic regression (binary & multiclass)
• Datasets: synthetic, MNIST, Reuters Corpus Vol. 1
• Algorithms for comparison:
  • Specialized coordinate descent for SE (FHT’07)
  • Interior Point (IP) method with L₁ Boundary Const.
  • Mirror (entropic) descent & Exponentiated Gradient
Synt. Logistic Regression

• Instances:
  • each component dist. N(0,1)
  • 50% relevant features
  • 10% label noise
• Average over 100 runs
Synt. Logistic Regression

N=4000 M=6000

Approximate Flops

$\frac{f^* - f}{f}$

$L1 - Batch$
$L1 - Stoch$
IP$

Approximate Flops $x 10^9$
Results for MNIST Data

- Learned predictor of the form
  \[ k(x, j) = \sum_{i \in S} w_{ji} \sigma_{ji} K(x_i, x), \quad \sigma_{ji} = \begin{cases} 1 & \text{if } y_i = j \\ -1 & \text{otherwise.} \end{cases} \]

- S: support-set, found using multicass Perceptron

- 60,000 training examples, 28x28 pixel images

- Multiclass logistic regression with L1

\[
\min_w \quad \frac{1}{m} \sum_{i=1}^{m} \log \left( 1 + \sum_{r \neq y_i} e^{k(x_i, r) - k(x_i, y_i)} \right) \\
\text{s.t.} \quad \|w_j\|_1 \leq z, \quad w_j \succeq 0.
\]
Results for MNIST Data

EG (Mirror Descent):

\[ w_j^{(t+1)} = \frac{w_j^{(t)} e^{-\eta_t \nabla_j(w^{(t)})}}{Z_t} \]
804,414 articles, 1,946,684 word bigrams

Each article includes ~0.26% of bigrams

Compared with Exponentiated Gradient (KW’97) [extension with positive & negative weights]

Both algorithms used the same domain constraints

Learning rate \( \sim 1/\sqrt{t} \)
$L_1$ Proj. vs. EG on RCV1
Concluding Remarks

• Bertsekas described first Euclidean projection onto the simplex (see also [Gafni & Bertsekas, 84]) using sorting

• Similar algorithms rediscovered and used as dual solvers for multiclass SVM, ranking problems, online learning (CS’01, CS’02, SS’06, Hazan’06)

• $L_1$-like experts tracking: Herbster & Warmuth’01

• First efficient $L_1$ domain-constrained algorithm for high dimensional inputs

• Extensions and current research:
  • Adding hyperbox constraints, non-Euclidean projections
  • Projection onto polytopes defined by mixed-norms
  • Algorithm for $L_1$ *regularization* through projections
  • Infusing AdaBoost with $L_1$ *regularization*
\[ \mathbf{w}_{t+1} = \Pi_X (\mathbf{w}_t - \eta_t \nabla_t L) \]

**Π**

\[ \Pi_X (\mathbf{w}) = \arg \min_{\mathbf{v}} \| \mathbf{w} - \mathbf{v} \| \\ \text{subject to} \mathbf{v} \in X \]

**Training Examples**

% Sparsity

% of Total Features

% of Total Seen

\[ \theta \]

\[ \theta \]