New Algorithms for Large-Scale Support Vector Machines

PhD Thesis Defense
Thesis advisor: Patrick Gallinari

Antoine Bordes
LIP6 - Université Pierre et Marie Curie - DGA

Roadmap

I. Large-Scale Machine Learning.

II. Linear Support Vector Machines.

III. Non-linear Support Vector Machines.

IV. Support Vector Machines for Structured Outputs.

V. Conclusion.
Part I

Large-Scale Machine Learning
## Rough Estimates

<table>
<thead>
<tr>
<th>Platform</th>
<th>Quantity</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>YouTube</td>
<td>&gt; 45 Terabytes of videos</td>
<td>(early 2007)</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>&gt; 13 millions articles</td>
<td>(mid 2009)</td>
</tr>
<tr>
<td>Facebook</td>
<td>&gt; 200 millions active users</td>
<td>(mid 2009)</td>
</tr>
<tr>
<td>Flickr</td>
<td>&gt; 3 billions photos</td>
<td>(end 2008)</td>
</tr>
<tr>
<td>Google</td>
<td>&gt; 1,000 billions indexed pages</td>
<td>(mid 2008)</td>
</tr>
</tbody>
</table>

**Huge data quantities** to be indexed, classified, transformed, ...  

**Problem:** Intractable for human operators.  

**Solution:** Employ machine learning (ML) to build automatic tools.

Can ML handle such large data quantities?

Antoine Bordes
Orders of Magnitude

Data quantities **outgrow** computing power!

Most ML algorithms require **resources larger than data dimensions**.

ML needs **outgrow** ML capabilities!
Motivation:
A deep need for ML methods that learn on large-scale databases.

Goal:
Propose new learning algorithms reducing training time and memory requirements with no loss in accuracy.

Scope:
Focus on Support Vector Machines, popular classification methods showing great performances in numerous domains.
Support Vector Machines

Training set \((x_i, y_i)_{i=1,\ldots,n} \in \mathbb{R}^d \times \{-1; 1\}\) and kernel \(k(x, \bar{x}) = \Phi(x)\top\Phi(\bar{x})\).

Primal Problem (MaxMargin)

\[
\min_w \mathcal{P}(w) = \frac{1}{2} w^2 + C \sum_{i=1}^{n} \ell(y_i, w\top\Phi(x_i) + b)
\]

with the hinge loss \(\ell(y, \hat{y}) = \max\{0, 1 - y\hat{y}\}\)

Dual Problem (QP)

\[
\max_{\alpha} \mathcal{D}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j k(x_i, x_j)
\]

subject to \(0 \leq \alpha_i \leq C\) and \(\sum_i y_i \alpha_i = 0\)

Strong Duality: \(\mathcal{P}(w^*) = \mathcal{D}(\alpha^*)\)

Training goal: learn the parameters \(w^*\) and \(b^*\) (or \(\alpha^*\)).
Batch Training

All training examples are known beforehand and stored in memory.

**PROS:**
- optimization methods stopping at convergence.
- good generalization performances.

**CONS:**
- costly optimization.
- high memory requirements and data streams forbidden.

Antoine Bordes

PhD Thesis Defense
Online/Stochastic Training

Training examples arrive in stream and are seen one by one.

**PROS:** 
- fast and cheap in memory needs.
- interesting in a large-scale setup.

**CONS:** in practice, worse in generalization than batch methods.
Contributions

**Direction:** combine online speed with batch accuracy.

New state-of-the-art algorithms for training SVMs:

- Linear SVMs: **SGD-QN**.
- Non-linear SVMs: **LASVM**.
- SVMs for structured output prediction: **LaRank**.

Not in this presentation:

- Study of active learning for SVMs.
- Learning under ambiguous supervision.
- Original work for natural language disambiguation.
Part II

Large-Scale Linear SVMs
Part II

Context:

Linear SVMs are very powerful for text applications, for example.
Ex: data mining/indexing, spam filtering, ...

Goal:

Accelerate their training to allow them to handle more data.

Outline:

1. Solve the primal problem efficiently with SGD.
2. Improve SGD using $2^{nd}$ order information: SGD-QN.
Linear Support Vector Machines

SVMs for binary classification using a linear kernel \( k(x, \bar{x}) = x^T \bar{x} \).

Efficient methods to learn linear SVMs solve the Primal:

**Primal Problem (no bias)**

\[
\min_w \mathcal{P}(w) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \ell(y_i, w^T x_i)
\]

with \( \ell(y, \hat{y}) = \max \{0, 1 - y \hat{y}\} \)

How to solve this problem efficiently with a lot of data?

Antoine Bordes
1st Order Stochastic Gradient

Simple and efficient way of learning large-scale linear SVMs.
Ex: SVMSGD2 (Bottou, 07), Pegasos (Shalev-Shwartz et al., 07), . . .

**SGD algorithm:**
1. Randomly pick a training instance \((x_t, y_t)\).
2. Update the parameter vector \(w\) with,
   \[
   w_{t+1} = w_t - \frac{1}{t+t_0} \frac{1}{\lambda} g(w_t, x_t)
   \]
   where \(g(w_t, x_t) = \lambda w_t + \ell'(y_t w_t^\top x_t) y_t x_t\)
3. Iterate.

**PROS:** Fast on large-scale data.

**CONS:**
- Suffer from ill-conditioned data \(\rightarrow\) slow it down.
- Bad optimization method (although good in generalization).
2\textsuperscript{nd} Order Stochastic Gradient

2\textsuperscript{nd} order SGD reaches the learning optimum after a single pass.
(Murata et al., 99; Bottou et al., 05)

The parameter update uses the inverse of the Hessian matrix $H^{-1}$:
(supposing that $\mathcal{P}(w)$ is twice-differentiable)

$$w_{t+1} = w_t - \frac{1}{t + t_0} H^{-1} g(w_t, x_t)$$

**PROS:** Very good generalization abilities.

**CONS:** Intractable to compute and store the dense matrix $H^{-1}$.
→ Must use estimates of $H^{-1}$. 

Antoine Bordes 15 PhD Thesis Defense
Estimation with Low Rank Matrices

**Batch methods:** Ex: L-BFGS (Nocedal et al., 80)

Low rank estimate formed by the last updates of the full gradient $G(w)$.

$\Rightarrow$ But in stochastic setup: $G(w)$ is unavailable.

**Online method:** oLBFGS (Schraudolph et al., 07)

Replace $G(w_t)$, $G(w_{t+1})$ with $g(w_t, x_t)$, $g(w_{t+1}, x_t)$ on the same example $(x_t, y_t)$.

**PROS:** Good stochastic low rank estimation of $H^{-1}$.

**CONS:** Many additional computations compared to SGD.
Estimation with Diagonal Matrices

Our idea: Estimate $H^{-1}$ with a diagonal matrix $D$.

Inspired by (Becker & LeCun, 89).

Online diagonal estimation:

1. Initialize $D$ with parameters of 1st order SGD.
2. Update $D$ on-the-fly approximating $H^{-1}$:
   Taylor expansion of $P(w) +$ estimation of $G(w_t)$ and $G(w_{t+1})$ as in oLBFGS.

**PROS:** fast and cheap process (few extra computation).

**CONS:** coarser estimation of $H^{-1}$ than oLBFGS.
Algorithm:

1. Randomly pick a training instance \((x_t, y_t)\)
2. Update \(w_{t+1} = w_t - \frac{1}{t+t_0} D \ell'(y_t w_t^T x_t) y_t x_t\)
3. Every \(T\) iteration:
   a. Update \(w_{t+1} = w_t - \frac{\lambda T}{t+t_0} D w_t\)
   b. Update \(D\) (requires an extra gradient computation)
4. Iterate.

Operations scheduling:

- Update can be split and scheduled differently:
  Operation 2. \(=\) cheap and Operation 3. \(=\) costly.
- \(T\) depends on the sparsity of the training instances.
Experiments

**Task:** Binary classification from (PASCAL LSChallenge, 08).

**Data:** ALPHA dataset (synthetic, ill-conditioned).

<table>
<thead>
<tr>
<th>Classes</th>
<th>Train. Ex.</th>
<th>Tst. Ex.</th>
<th>Features</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>100,000</td>
<td>50,000</td>
<td>500</td>
<td>Linear</td>
</tr>
</tbody>
</table>

**Comparisons:** SVMMSGD2, oLBFGS and SGD-QN.
Summary Part II

**Motivation:** Accelerate SGD training to handle more data.

**Achievement:** The new SGD-QN algorithm.
1. It requires less iterations than SGD.
2. It is finally faster than SGD for linear problems.

**Highlight:**
SGD-QN won the Pascal Large Scale Learning Challenge in 2008. (equal 1\textsuperscript{st} over 40 international competitors).
Part III

Large-Scale Non-linear SVMs
Context:

– Limited applicative range for linear SVMs → non-linearity.
  Ex: image recognition, vision, . . .
– Non-linearity complicates the model → dual optimization

Goal:

Propose a new learning algorithm saving time and memory.

Outline:

1. Solve the dual SVM problem with gradient ascent.
2. Combine online & batch with Process/Reprocess: **LASVM**.
Non-linear SVMs

Binary classification using a non-linear kernel \( k(x, \bar{x}) = \Phi(x)^\top \Phi(\bar{x}) \).

Limitations of SGD-QN (and SGD):

For non-linear SVMs, solving the dual problem is more efficient:

**Dual Problem (QP)**

\[
\max_{\alpha} \quad D(\alpha) = \sum_{i=1}^{n} y_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j \ k(x_i, x_j) \\
\text{subject to} \quad 0 \leq \alpha_i \leq C \quad \text{and} \quad \sum_i \alpha_i = 0
\]

**Support Vectors (SV):** all \( x_i \) with \( \alpha_i \neq 0 \).

**Parameter vector:** \( \mathbf{w}^* = \sum_i \alpha_i^* \Phi(x_i) \)

New optimization problem → new algorithm.
Solving the Dual QP

Dual ascent with direction search:

1. Pick a direction in the dual space defined by $\alpha$.
2. Perform the best possible step in that direction.
3. Iterate.

Properties:
- Analytical computation of the gradient step (+ box constraints).
- Always progress towards the solution.

How to choose the successive ascent directions?

Antoine Bordes
Choosing Directions

**Batch strategy:**
All examples known beforehand → follow the highest full gradient.
Ex: SMO (Platt, 99), LibSVM (Chang et al., 05), ... → accurate but costly in training time.

**Online strategy:**
Stream of examples → directions imposed by the new example.
Ex: AvgPerceptron (Freund et al., 99), Passive-Aggressive (Crammer et al., 06). → cheap and fast but less accurate.

**Our strategy:**
Mix direction choices to combine online speed with batch accuracy.
Process/Reprocess

Online alternation of directions defined by:

- a fresh example → Process.
- the memory-stored SV with the highest gradient → Reprocess.

Equivalent view:

- Time management: computation cost vs dual increase.
- Memory management: online insertion and removal of SVs.
**Algorithm:**

1. **Initialization** with few examples of each class.
2. **Online iterations:** repeat a predefined number of times,
   - Randomly pick a training instance \((x_t, y_t)\).
   - Perform \(\text{Process}(x_t, y_t)\).
   - Perform \(\text{Reprocess()}\) once.
3. **Finishing:** \(\text{Reprocess()}\) until convergence (optional).

**Theoretical properties:**

- **Convergence:** to the batch SVM solution.
- **Tracking guarantee:** \(\text{Process/Reprocess}\) allows to track the SVM batch optimum on the course of learning.
Experiments

**Task:** Handwritten digits recognition.

![Handwritten digits]

**Data:** MNIST dataset treated in “One-vs-All”.

<table>
<thead>
<tr>
<th>Classes</th>
<th>Train. Ex.</th>
<th>Tst. Ex.</th>
<th>Features</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>60,000</td>
<td>10,000</td>
<td>780</td>
<td>Gaussian</td>
</tr>
</tbody>
</table>

**Stopping:** one epoch ([LASVMx1](#)).
MNIST “One-vs-All”
Caching often-used kernel values is essential for tractability.

LASVM depends less on the kernel cache size than LibSVM.

→ Using Process/Reprocess saves memory usage.
Motivation: Accelerate non-linear SVMs training.

Achievement: The new LASVM algorithm.
1. It reaches the batch accuracy after a fast single epoch.
2. It converges towards the batch SVM solution.
3. It makes a sparse use of memory.

Highlight:
LASVM has been used to train SVMs on 8M examples on a single CPU (in 20h with 6Go of RAM) (Loosli et al., 06).
Part IV

Structured Output Prediction
Many problems are more complex than binary classification.
Application: natural language processing, bioinformatics, . . .

In such cases, instances consist in couples (input ; output) which can both involve structures, such as sequences, trees, . . .
Part IV

Context:

- Structured output prediction → more elaborate problems.
- Dealing with structured objects worsen scaling issues.

Goal:

Propose a new learning algorithm combining the speed of online methods and the accuracy of batch ones.

Outline:

1. Express structured output prediction as a SVM problem.
**Kernels Ideas** (Taskar et al.; Altun et al., 2003)

**Problem 1:** dependencies within input and output structures.

**Joint kernel encoding:**
- A joint kernel $K$ encodes correlations of a pair (input; output).
- A cost $\Delta$ measures the errors between two output structures.

**Problem 2:** large output spaces, intractable enumeration.

**Ranking formulation:**
- We learn a function $S$ scoring any pair (input; output).
- Constraints: score of a correct pair $> \text{score of any incorrect}$.
- Combined with an efficient prediction $\rightarrow$ bypass enumeration.
SVM Formulation (Tsochantaridis et al., 05)

We can derive a SVM formulation with training examples \((x_i, y_i)\).

**Primal problem:**

\[
\min_w \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i \quad \text{subject to} \quad \begin{cases} 
\forall i \xi_i \geq 0 \\
\forall i \forall y \neq y_i \quad S(x_i, y_i) \geq S(x_i, y) + \Delta(y, y_i) - \xi_i
\end{cases}
\]

**Dual problem:**

\[
\max_{\beta} - \sum_{i,y} \Delta(y, y_i) \beta_y^i - \frac{1}{2} \sum_{i,j,y,\bar{y}} \beta_y^i \beta_{\bar{y}}^j K(x_i, y, x_j, \bar{y}) \quad \text{subject to} \quad \begin{cases} 
\forall i \forall y \beta_y^i \leq C \delta_{yy} \\
\forall i \sum_y \beta_y^i = 0
\end{cases}
\]

**Limitation:** Exponential number of constraints.

**Consequence:** any method must add them incrementally.

\[
\rightarrow \text{batch methods are very slow (full gradient intractable).}
\]
1. **Online alternation of 3 optimizations steps:**
   - *Process*: on a new example $\rightarrow$ **add** constraints.
   - Two types of *Reprocess*:
     1. *ProcessOld*: previously seen ex $\rightarrow$ **add**/ **remove** constraints.
     2. *Optimize*: only on existing constraints $\rightarrow$ **remove** constraints.

2. **Operations scheduling:**
   - *Fixed*: alternation of a predefined number.
   - *Adaptive*: sampling according to a distribution of efficiency.

3. **Tractability**: LaRank never uses the full gradient.

4. **Complexity**: Convergence after at most $O(n)$ iterations.
App. #1: MultiClass Classification

**Goal:** for any input $x$ predict its atomic label $y$.

**Components:**

- **Kernel:** $K(x, y, \bar{x}, \bar{y}) = \delta_{y\bar{y}} k(x, \bar{x})$
- **Cost:** $\Delta(y, \bar{y}) = 1 - \delta_{y\bar{y}}$

**Observations:**

- Simplest structured output task on which test LaRank.
- No real input or output structure.
- But many comparisons in the literature.

Antoine Bordes
**Experiments**

**Task:** Handwritten digits recognition.

![Handwritten digits images]

**Data:** MNIST dataset.

<table>
<thead>
<tr>
<th>Classes</th>
<th>Train. Ex.</th>
<th>Test. Ex.</th>
<th>Features</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>60,000</td>
<td>10,000</td>
<td>780</td>
<td>Gaussian</td>
</tr>
</tbody>
</table>

**Stopping:** one epoch (LaRankx1).

Antoine Bordes
MNIST
App. #2: Sequence Labeling

**Goal:** Given an input sequence $x$ of tokens, predict the corresponding sequence $y$ of labels.

**Inference scheme:** Process that recovers the output sequence given an input, assigning a label $y_t$ to each token $x_t$.

**Structured modeling hypothesis:**
1. **Conditional independence:** a label $y_t$ is only function of neighborhoods.
2. **Invariance:** this function does not depend on the time index $t$. 

Antoine Bordes
Inference Schemes

1. **Exact inference:** global maximization over the whole sequence.
   \[ \rightarrow \text{solved with dynamic programming (Viterbi)}. \]
   
   Kernel:
   \[
   K(x, y, \bar{x}, \bar{y}) = \sum_{s,t} \delta_{y_s, \bar{y}_t} k(x_s, \bar{x}_t) + \sum_{s,t} \delta_{y_s, \bar{y}_t} \delta_{y_{s+1}, \bar{y}_{t+1}}.
   \]
   
   Cost: *Hamming* loss.

2. **Greedy inference:** local dependencies on a token and few past labels.
   \[ \rightarrow \text{fast and can use longer Markov interactions}. \]
   
   Kernel:
   \[
   K(x, y, \bar{x}, \bar{y}) = \delta_{y\bar{y}} \left[ k(x, \bar{x}) + \sum_{j \in J} \delta_{y_j, \bar{y}_j} \right].
   \]
   
   Cost:
   \[ \Delta(y, \bar{y}) = 1 - \delta_{y\bar{y}}. \]
Experiments

Task: Part-Of-Speech tagging.
– Associate its grammar label to each word of a sentence.
– Example:

```
PRP VBD DT NN
| | | |
He opened the window
```

Data: Wall Street Journal dataset.

<table>
<thead>
<tr>
<th>Labels</th>
<th>Train</th>
<th>Test</th>
<th>Features</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>42,466 sequences</td>
<td>2,155 sequences</td>
<td>&gt; 130,000</td>
<td>Linear</td>
</tr>
<tr>
<td></td>
<td>(≈ 1,000,000 tokens)</td>
<td>(≈ 53,000 tokens)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Stopping: one epoch (LaRankExact and LaRankGreedy).

Antoine Bordes
Summary Part IV

**Motivation:** Accelerate learning of SVMs for structured outputs.

**Achievement:** The new LaRank algorithm.

1. Compared to batch methods, it is:
   - much faster in training.
   - as accurate in generalization.

2. Compared to online methods, it is:
   - almost as fast in training.
   - much more accurate in generalization.

**Highlight:**
LaRankGreedy reaches state-of-the-art accuracy after training only 3 min. on 42,000 WSJ sentences on a single CPU.

Antoine Bordes
Part V

Conclusion
Conclusion

A broad range of methods sharing desirable properties:
- fast;
- cheap in memory;
- competitive accuracies;
- theoretical guarantees: convergence, complexity, ...

Generic tools which can be applied to numerous applications:
- active learning: LASVM (Ertekin et al., 07);
- bioinformatics: LASVM (Morgado et al., 09);
- webpage ranking: LaRank (Usunier et al., 09);
- semi-supervised ranking: LaRank (Truong, 09);
- ...
Thank You for your Attention