LaRank, SGD-QN
Fast Optimizers for Linear SVMs

Antoine Bordes[*†] **& Léon Bottou**[†]

[*] LIP6, Université de Paris 6, Paris.

[†] NEC Laboratories America, Princeton.

ICML'08 Workshop PASCAL Large Scale Learning Challenge,
July 9, 2008. Helsinki, Finland.

Forewords

Two algorithms for solving two class linear SVM are presented:
– LaRank operates in the dual. (*Linear SVM Track*)
– SGD-QN operates in the primal. (*Wild Track*)

Both of them:
– are **online** algorithms.
– converge to the SVM solution after few epochs.

Why using linear SVMs? when Ronan Collobert says:
"Large-scale problems require models more complex than linear".

→ Linear SVMs = tools to study fast optimization algorithms.
→ Both LaRank and SGD-QN have more complex final purposes.

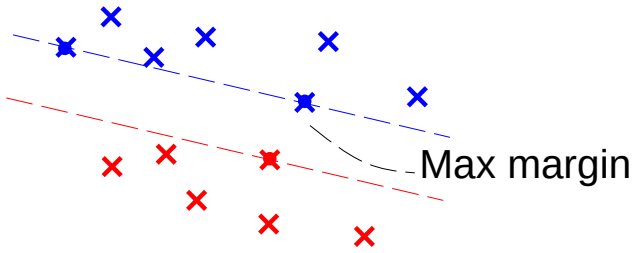
Part I

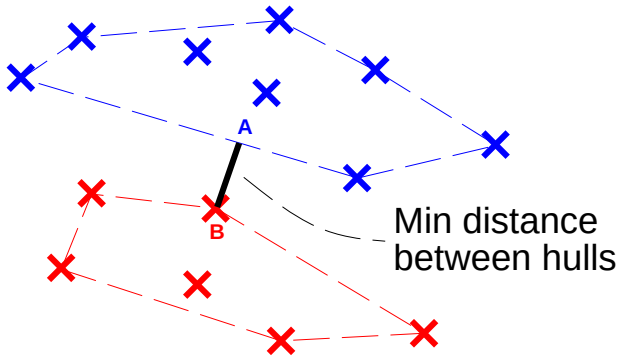
LaRank

Online Dual SVM Solver

Convex Duality

Two class SVM with linear Kernel and Hinge Loss.





Primal Problem (MaxMargin)

$$\min_{\mathbf{w}} \; \mathcal{P}(\mathbf{w}) = \frac{1}{2}\,\mathbf{w}^2 + C \sum_{i=1}^{n} \ell(y_i, \mathbf{w}^\top \mathbf{x}_i)$$

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

Dual Problem (QP)

$$\max_{\boldsymbol{\alpha}} \; \mathcal{D}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} y_i\,\alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i\,\alpha_j\,\mathbf{x}_i^\top \mathbf{x}_j$$

subject to $0 \le \alpha_i \le C$

Strong Duality

$$\mathcal{P}(\mathbf{w}^*) = \mathcal{D}(\boldsymbol{\alpha}^*) \qquad \mathbf{w}^* = \sum_i \alpha_i^*\,\mathbf{x}_i$$

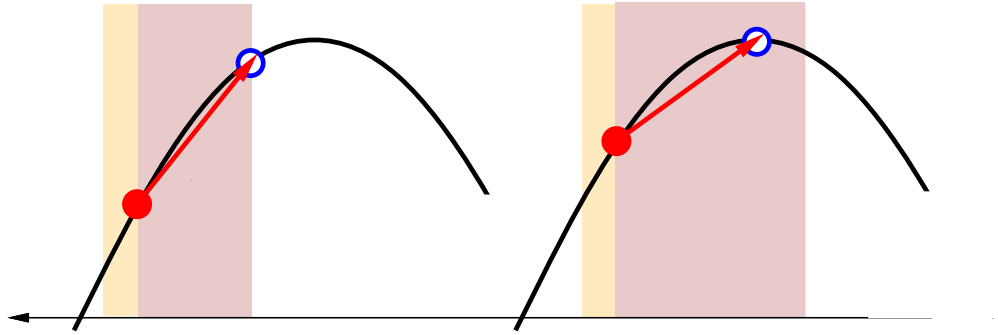
Note: There is a coefficient α_i associated to each example $\mathbf{x}_i$.

QP with Direction Search

Algorithm:

1. Pick a direction in α-space.
2. Perform the best possible step in that direction:

$$\alpha_i(t+1) \;=\; \max\left\{\, 0,\;\; \min\left\{\, \alpha_i(t) - \frac{y_i \mathbf{w}_t^\top \mathbf{x}_i - 1}{\mathbf{x}_i^\top \mathbf{x}_i},\; C \,\right\}\right\}$$



3. Iterate.

Property: Always makes progress towards the solution.

Sparse Direction Search

Picking very sparse search directions for each iteration
reduces the computation more than it hurts convergence.

- **for standard SVMs** (SMO, Platt, 99):
  Only adjust two coefficients α_i and α_j,
  by opposite amounts to satisfy the equality constraint.

- **for SVMs without bias** b:
  No equality constraint in the dual problem.
  Only adjust a single coefficient α_i.

Process and Reprocess

Observations

(i) Directions that involve a fresh example seldom increase the dual because its α was and often remains zero.

(ii) Directions that only involve current support vectors often increase the dual because their nonzero α can be adjusted.

$\rightarrow$ Randomly choosing directions picks (ii) more often than (i).

LaRank

- Main idea: Alternating directions of type (i), aka *process*, and directions of type (ii), aka *reprocess*.
- Refinements: Dealing with support vectors with $\alpha = C$, rebalancing more classes of directions, adaptive selection of direction types, active selection of fresh examples, etc.

Convergence Properties

When LaRank picks a direction, it enforces two properties:

1. the gradient in this direction is greater than $\tau > 0$,
2. a movement of $\kappa > 0$ is possible without leaving the constraint polytope.

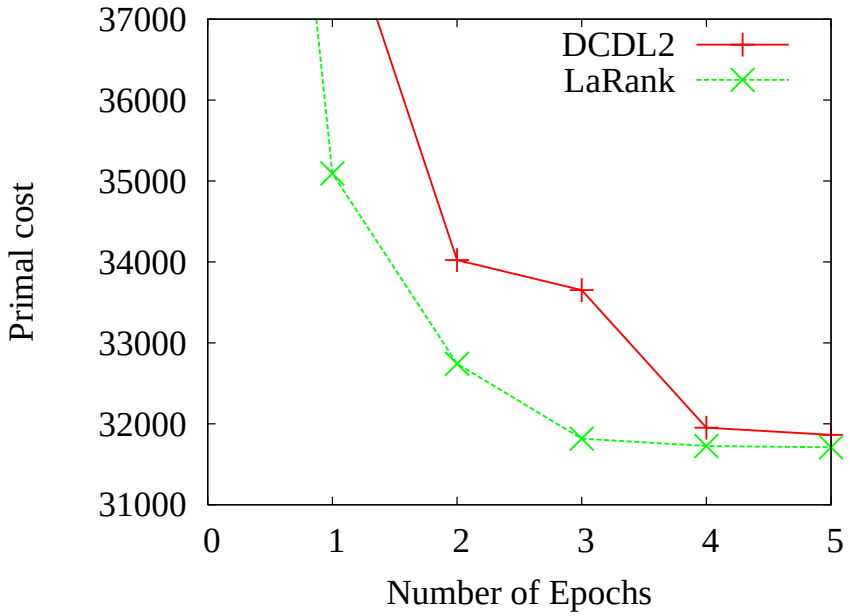
$\rightarrow$ LaRank is an *Approximate Stochastic Witness Direction Search* algorithm like SMO, LibLinear, and a few others... Then:

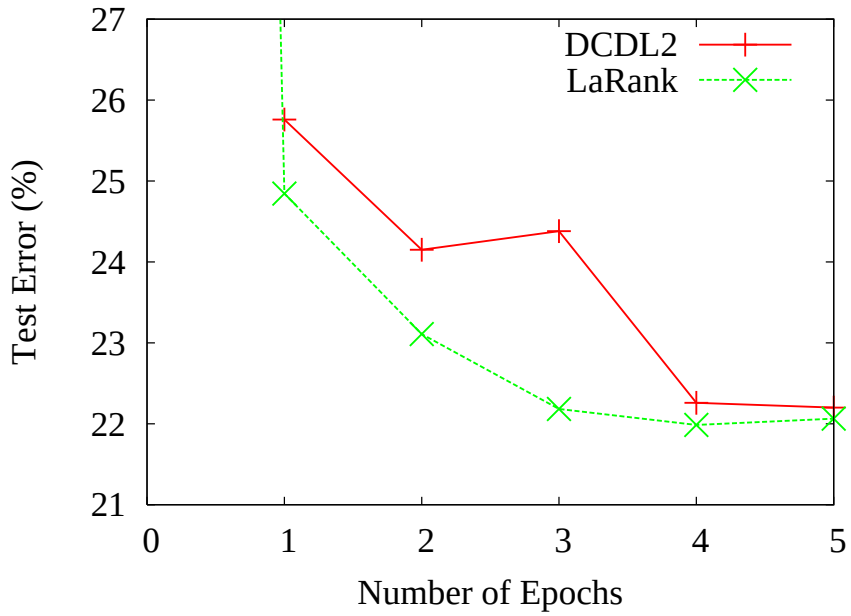
Theorem 18 of (Bordes et al., 05):
$\rightarrow$ Mild conditions on the direction picks ensure convergence.

Theorem (Bordes et al., 07): *With probability 1,* LaRank *reaches a* $\kappa\tau$*-approximate solution of dual problem, with no more than* $\max\{\frac{2\nu^2 nC}{\tau^2}, \frac{2nC}{\kappa\tau}\}$ *successful iterations.*
$\rightarrow$ Reaches predefined accuracy after $\mathcal{O}(n)$ iterations.

Empirical Behavior

Experiments on subsets of Alpha official training set:
(train: 100,000, test: 50,000)





$\longrightarrow$ Use of *reprocess* reduces number of epochs.

History of LaRank

Huller: (Bordes & Bottou, 05)
Online hard-margin SVM solver alternating directions in optimization steps.

LaSVM: (Bordes et al., 05)
Efficient online solver for two class SVM wih bias using optimization steps based on SMO.

LaRank:(Bordes et al., 07, 08)
– Originally proposed for structured output predictions.
– Successfully applied to multiclass classification and sequence labelling.

Part II

SGD-QN

SGD with Diagonal Quasi-Newton Scaling

Preprocessing

SGD (and SGD-QN) are very sensitive to data conditionning.
→ Preprocessing is crucial:

- In the case of dense datasets: all features are normalized with mean 0 and variance 1.
- In the case of sparse datasets: all features are normalized to have maximal absolute value 1.
- All examples are normalized to L2 norm 1 across features.

Note: we did nor use any preprocessing with LaRank.

Plain Stochastic Gradient Descent

Primal Problem can be re-written as:

$$\mathcal{P}(\mathbf{w}) \quad = \quad \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\lambda}{2} \|\mathbf{w}\|^2 + \ell(y_i, \mathbf{w}^\top \mathbf{x}_i) \right) \text{, with } \lambda = \frac{1}{nC}$$

Stochastic Gradient Descent:

1. Draw a random training example $(\mathbf{x}_t, y_t)$.
2. Compute a new value of the parameter with,

$$\mathbf{w}_{t+1} \quad = \quad \mathbf{w}_t - \frac{1}{t + t_0} \frac{1}{\lambda} \mathbf{g}_t(\mathbf{w}_t) \quad \text{where} \quad \mathbf{g}_t(\mathbf{w}_t) = \lambda \mathbf{w}_t + \ell'(y_t \, w_t^\top \mathbf{x}_t) \, y_t \, \mathbf{x}_t$$

3. Iterate.

Pros: Fast and simple; Good generalization guarantees.

Cons: Low rate of convergence, might require many epochs.

Second Order Stochasic Gradient

The update step uses the inverse of Hessian matrix , $\mathbf{H}^{-1}$:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{1}{t+t_0}\mathbf{H}^{-1}\,\mathbf{g}_t(\mathbf{w}_t)$$

Property:
Asymptotically and on average, the parameter $\mathbf{w}_n$ obtained after one pass is as close to the infinite training set solution $\mathbf{w}^*$ as the true optimum of the primal $\mathbf{w}_n^*$. (Murata et al., 99), (Bottou et al., 05)

Drawback:
Impractical to compute, store and use the dense matrix $\mathbf{H}^{-1}$.

$\rightarrow$ Need to use sparse approximations of $\mathbf{H}^{-1}$.

Low Rank Scaling Matrices

Batch Methods: full gradients $\mathcal{P}'_n(\mathbf{w}_{t-1})$ and $\mathcal{P}'_n(\mathbf{w}_t)$ available:
$\rightarrow \mathbf{H}^{-1}$ estimated using k rank one updates. Ex: LBFGS method.

Stochastic Methods: only access to $\mathbf{g}_{t-1}(\mathbf{w}_{t-1})$ and $\mathbf{g}_t(\mathbf{w}_t)$.
$\rightarrow$ noisy estimates of $\mathcal{P}'_n(\mathbf{w}_{t-1})$ and $\mathcal{P}'_n(\mathbf{w}_t)$, **it doesn't work!**

oLBFGS: (Schraudolph et al., 07)
estimate using $\mathbf{g}_{t-1}(\mathbf{w}_{t-1})$ and $\mathbf{g}_{t-1}(\mathbf{w}_t)$ for the same example $(\mathbf{x}_t, y_t)$.

Pros: Good online low rank estimation of $\mathbf{H}^{-1}$.

Cons: Additional computations (compared to SGD):
- gradient $\mathbf{g}_{t-1}(\mathbf{w}_t)$,
- update of the k rank one updates,
- product of the gradient $\mathbf{g}_t(\mathbf{w}_t)$ by the low-rank estimate of rank k.

Diagonal Scaling Matrices

Main idea: estimate $\mathbf{H}^{-1}$ by a diagonal matrix $\mathbf{D}$ $\rightarrow$ fast, cheap.

Application: for any recent $\mathbf{w}_{t-1}$ and $\mathbf{w}_t$, $\mathbf{D}$ is defined by,

$$\mathbf{w}_t - \mathbf{w}_{t-1} \approx \mathbf{D} \left[\mathcal{P}'_n(\mathbf{w}_t) - \mathcal{P}'_n(\mathbf{w}_{t-1}) \right]$$

1. Replace $\mathcal{P}'_n(\mathbf{w}_t)$ / $\mathcal{P}'_n(\mathbf{w}_{t-1})$ by $\mathbf{g}_{t-1}(\mathbf{w}_t)$ / $\mathbf{g}_{t-1}(\mathbf{w}_{t-1})$ (cf. oLBFGS):

$$\mathbf{w}_t - \mathbf{w}_{t-1} \approx \mathbf{D} \left[\mathbf{g}_{t-1}(\mathbf{w}_t) - \mathbf{g}_{t-1}(\mathbf{w}_{t-1}) \right] .$$

2. Update the diagonal estimated matrix $\mathbf{D}$ online using

$$\forall k = 1 \dots d \,, \qquad \mathbf{D}_{kk} \leftarrow \mathbf{D}_{kk} + \frac{2}{t} \left(\frac{[\mathbf{w}_t - \mathbf{w}_{t-1}]_k}{[\mathbf{g}_{t-1}(\mathbf{w}_t) - \mathbf{g}_{t-1}(\mathbf{w}_{t-1})]_k} - \mathbf{D}_{kk} \right)$$

The SGD-QN **Algorithm**

SGD-QN *(work in progress)*:

1. Draw a random training example $(\mathbf{x}_t, y_t)$
2. Compute $\mathbf{g}_t^\ell(\mathbf{w}_t) = \ell'(y_t \mathbf{w}_t^\top \mathbf{x}_t)\, y_t\, \mathbf{x}_t$
3. Update $\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{1}{t+t_0} \mathbf{D}\, \mathbf{g}_t^\ell(\mathbf{w}_t)$
4. Every T iteration:
   a. Update $\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\lambda T}{t+t_0} \mathbf{D}\, \mathbf{w}_t$ *(weight decay)*
   b. Update $\mathbf{D}$ *(require an extra gradient computation)*
5. Iterate.

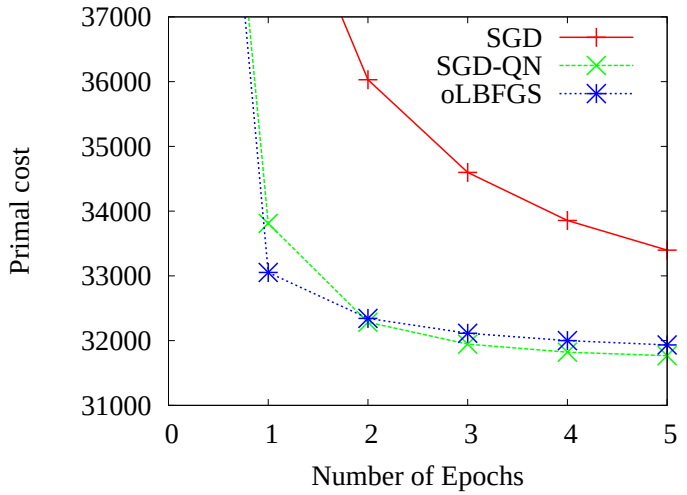
Operations scheduling:

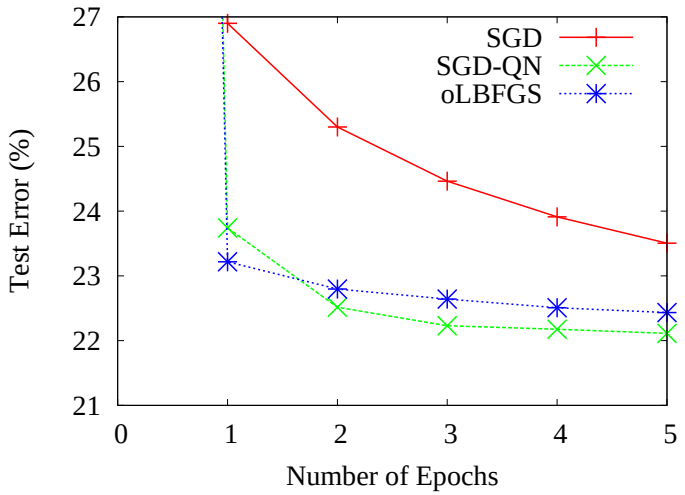
– Operations 4.a and 4.b can be scheduled differently.

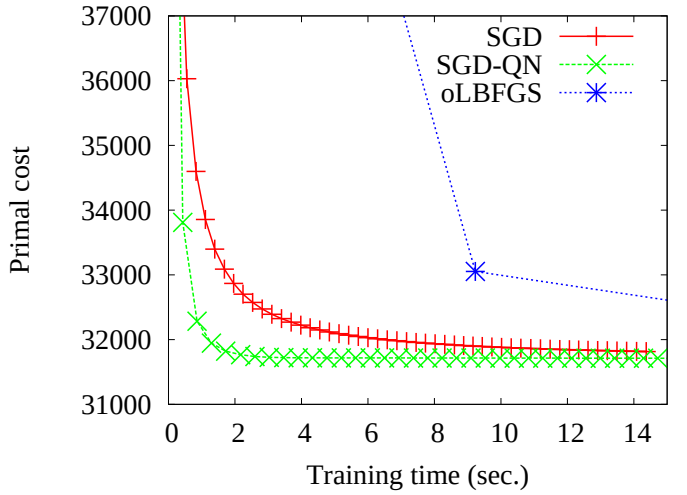
– T depends on the sparsity of the examples.

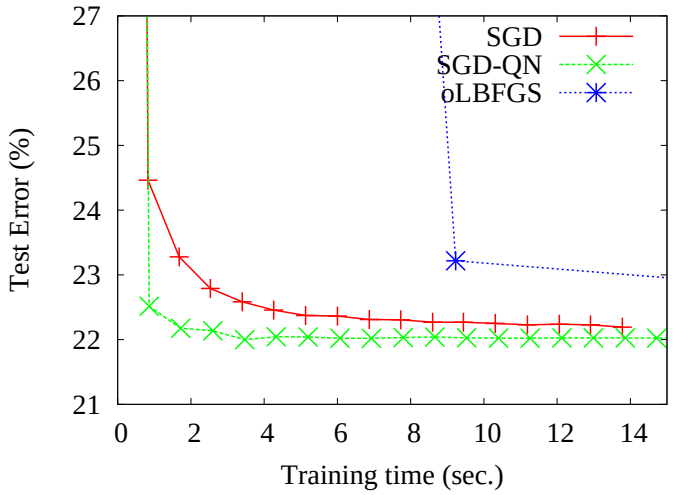
(Note: Léon Bottou's SGD uses the same scheduling trick.)

Comparing First and Second Order SGD









Part III

Remarks About Wild Track Criteria

Absolute Error Rates

Distinguish the relative reduction of:

(a) *achieved error rate - best achievable error rate*
(b) *achieved error rate*. (scenario used in the challenge).

(a) determines the computational cost. (Bottou & Bousquet 08).

(b) is often more important in practice but can be very different when the bayes error is relatively large.

There is a point where doubling the number of examples will:

- cost a lot of computation,
- have a clear effect on criterion (a),
- virtually have no effect on criterion (b).

Question

Is it worth submitting a method using every training examples if it only reduces criteria (a)?

Reward Small-Scale Results

Complex models could not win the competion because the criteria "eliminate the engineering bias" using multiple calibration points.

For example given AuTime and AuPRC criteria, strategies like:

- seeking more accuracy on small sets,
- artificially augmenting the computational effort on small sets,
- not even training on the largest set but reporting a result obtained on a subset,
- ...

$\rightarrow$ Improves both AuTime and AuPRC ranking and the apparent exponent of the effort.

Questions

- Are they still large-scale?
- Should we "eliminate the engineering bias" when discussing large-scale?

Thank You