(Un)conventional regularization for efficient machine learning

Lorenzo Rosasco
Laboratory for Computational Statistical Learning (LCSL)
University of Genova
Massachusetts Institute of Technology
Istituto Italiano di Tecnologia
lcsl.mit.edu

Nov. 23rd, 2018 – Horizon Maths 2018: Intelligence Artificielle

joint work with L. Carratino, R. Camoriano (LCSL), J. Lin (EPFL), A. Rudi (INRIA)
Beyond the statistics vs optimization dichotomy:

**Numerical resources budgeted to data quality (not just size)**

[Bottou, Bousquet '08]
Outline

Classical regularization

Regularization by optimization

Regularization & Projections

Preconditioning
Statistical learning

Find

$$w_* = \arg\min_{w \in \mathbb{R}^p} \mathbb{E}[(y - \langle w, \Phi(x) \rangle)^2]$$

given only i.i.d. samples $$(x_i, y_i)_{i=1}^n$$. 

We assume throughout that:

- $\Phi$ a given high/infinite dimensional representation.
- If $p = \infty$, $\langle \Phi(x), \Phi(x') \rangle$ can be computed in $O(1)$ (kernel methods/Gaussian processes).
Empirical risk minimization (ERM)

\[ \hat{w}_\lambda = \arg\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n (y_i - \langle w, \Phi(x_i) \rangle)^2 + \lambda \| w \|^2 \]

Theorem (Smale, Zhou '05, Caponetto De Vito '05)

If \( p = \infty \), \( \| \Phi(x) \|, |y| \leq 1 \) a.s. and \( \lambda = 1 / \sqrt{n} \)

\[ E \left[ \left( \langle \Phi(x), \hat{w}_\lambda \rangle - \langle \Phi(x), w^* \rangle \right)^2 \right] \lesssim \frac{1}{\sqrt{n}} \]

Proof

\( \forall \lambda > 0 \), \( E \left[ \left( \langle \Phi(x), \hat{w}_\lambda \rangle - \langle \Phi(x), w^* \rangle \right)^2 \right] \lesssim 1 \lambda n + \lambda \)

Remark:

Optimal bound (can be improved under further assumptions)
Empirical risk minimization (ERM)

\[ \hat{w}_\lambda = \arg\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (y_i - \langle w, \Phi(x_i) \rangle)^2 + \lambda \|w\|^2 \]

Theorem (Smale, Zhou '05, Caponetto De Vito '05)

If \( p = \infty, \|\Phi(x)\|, |y| \leq 1 \text{ a.s. and } \lambda = 1/\sqrt{n} \)

\[ \mathbb{E}[\langle \Phi(x), \hat{w}_\lambda \rangle - \langle \Phi(x), w_* \rangle^2] \lesssim \frac{1}{\sqrt{n}} \]
Empirical risk minimization (ERM)

\[ \hat{w}_\lambda = \arg\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (y_i - \langle w, \Phi(x_i) \rangle)^2 + \lambda \| w \|^2 \]

**Theorem (Smale, Zhou ’05, Caponetto De Vito ’05)**

If \( p = \infty, \| \Phi(x) \|, |y| \leq 1 \text{ a.s. and } \lambda = 1/\sqrt{n} \)

\[ \mathbb{E}[(\langle \Phi(x), \hat{w}_\lambda \rangle - \langle \Phi(x), w_* \rangle)^2] \lesssim \frac{1}{\sqrt{n}} \]

**Proof**

\[ \forall \lambda > 0, \quad \mathbb{E}[(\langle \Phi(x), \hat{w}_\lambda \rangle - \langle \Phi(x), w_* \rangle)^2] \lesssim \frac{1}{\lambda n} + \lambda \]

**Remark:** Optimal bound (can be improved under further assumptions)
ERM computations

\[ \hat{w}_\lambda = \arg\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (y_i - \langle w, \Phi(x_i) \rangle)^2 + \lambda \|w\|^2 \]

Nonparametrics \( n < p = \infty \)

\[ \hat{c}_\lambda = (\hat{\Phi}^\top \hat{\Phi} + \lambda n I)^{-1} \hat{y} \]

- \( \hat{\Phi} \) is the \( n \) by \( p \) data/feature matrix
- \( \hat{w}_\lambda = \hat{\Phi}^\top \hat{c}_\lambda \)
- \( \langle \hat{w}_\lambda, \Phi(x) \rangle = \sum_{i=1}^{n} \langle \Phi(x_i), \Phi(x) \rangle \hat{c}_\lambda^i \)

\[ O(n^3) + O(n^2) \]

\( \text{time} \) \( \text{space} \)
time $O(n^3)$ + space $O(n^2)$ for optimal $O(1/\sqrt{n})$ learning bound

The rate $1/\sqrt{n}$ is optimal, can we improve time/space requirements?
Outline

Classical regularization

Regularization by optimization

Regularization & Projections

Preconditioning
Gradient descent (GD) learning
aka L2 boosting, Landweber iteration

GD for

$$\frac{1}{n} \sum_{i=1}^{n} (y_i - \langle w, \Phi(x_i) \rangle)^2 = \frac{1}{n} \| \hat{\Phi}w - \hat{y} \|^2$$
Gradient descent (GD) learning
aka L2 boosting, Landweber iteration

GD for

$$\frac{1}{n} \sum_{i=1}^{n} (y_i - \langle w, \Phi(x_i) \rangle)^2 = \frac{1}{n} \| \hat{w} - \hat{y} \|^2$$

Nonparametrics $n \leq p = \infty$

$$\hat{c}_{t+1} = \hat{c}_t - \frac{\gamma}{n} (\hat{\Phi} \hat{\Phi}^\top \hat{c}_t - \hat{y})$$

- $\hat{w}_\lambda = \hat{\Phi}^\top \hat{c}_t$
- $\langle \hat{w}_t, \Phi(x) \rangle = \sum_{i=1}^{n} \langle \Phi(x_i), \Phi(x) \rangle \hat{c}_t^i$

$$O(n^2 t) + O(n^2)$$

\(\underline{time} + \underline{space}\)
Gradient descent (GD) learning
aka L2 boosting, Landweber iteration

GD for

\[ \frac{1}{n} \sum_{i=1}^{n} \left( y_i - \langle w, \Phi(x_i) \rangle \right)^2 = \frac{1}{n} \| \hat{\Phi} w - \hat{y} \|^2 \]

Nonparametrics \( n \leq p = \infty \)

\[ \hat{c}_{t+1} = \hat{c}_t - \gamma \left( \hat{\Phi} \hat{\Phi}^\top \hat{c}_t - \hat{y} \right) \]

- \( \hat{w}_\lambda = \hat{\Phi}^\top \hat{c}_t \)
- \( \langle \hat{w}_t, \Phi(x) \rangle = \sum_{i=1}^{n} \langle \Phi(x_i), \Phi(x) \rangle \hat{c}_t^i \)

\( O(n^2 t) + O(n^2) \)

Why should this work??
Fitting on the training set
Iteration #1
An intuition

Fitting on the training set
Iteration #2
An intuition

Fitting on the training set
Iteration #7
An intuition

Fitting on the training set
Iteration #5000
Regularization and stability

\[ \hat{w}_t = \frac{\gamma}{n} \sum_{j=0}^{t-1} (I - \frac{\gamma}{n} \hat{\Phi}^\top \hat{\Phi})^j \hat{\Phi}^\top \hat{y} \]

Large \( t \)

\[ \hat{w}_t = \frac{\gamma}{n} \sum_{j=0}^{t-1} (I - \frac{\gamma}{n} \hat{\Phi}^\top \hat{\Phi})^j \hat{\Phi}^\top \hat{y} = \approx \frac{\gamma}{n} \sum_{j=0}^{\infty} (I - \frac{\gamma}{n} \hat{\Phi}^\top \hat{\Phi})^j \hat{\Phi}^\top \hat{y} = (\hat{\Phi}^\top \hat{\Phi})^{-1} \hat{\Phi}^\top \hat{y} \]

Small \( t \)

\[ \hat{w}_t = (I - (I - \frac{\gamma}{n} \hat{\Phi}^\top \hat{\Phi}))^t (\hat{\Phi}^\top \hat{\Phi})^{-1} \hat{\Phi}^\top \hat{y} \propto \Phi^\top \hat{y} \]

compare to

\[ \hat{w}_\lambda = (\hat{\Phi}^\top \hat{\Phi} + \lambda n I)^{-1} \hat{\Phi}^\top \hat{y} \propto \Phi^\top \hat{y} \]

\( \lambda \) large
Theorem (Bauer, Pereverzev, R. ’07)

If $p = \infty$, $\|\Phi(x)\|, |y| \leq 1$ a.s. and $t = \sqrt{n}$

$$\mathbb{E}[(\langle \Phi(x), \hat{w}_t \rangle - \langle \Phi(x), w_* \rangle)^2] \lesssim \frac{1}{\sqrt{n}}$$
Statistics

Theorem (Bauer, Pereverzev, R. '07)

If \( p = \infty, \|\Phi(x)\|, |y| \leq 1 \) a.s. and \( t = \sqrt{n} \)

\[
\mathbb{E}[\left(\langle \Phi(x), \hat{w}_t \rangle - \langle \Phi(x), w_* \rangle \right)^2] \lesssim \frac{1}{\sqrt{n}}
\]

Proof

\[ \forall t > 1, \quad \mathbb{E}[\left(\langle \Phi(x), \hat{w}_t \rangle - \langle \Phi(x), w_* \rangle \right)^2] \lesssim \frac{t}{n} + \frac{1}{t} \]

Remarks:

- Same bound as Tikhonov regularization.
Computational regularization

time $O(n^2 \sqrt{n})$ + space $O(n^2)$ for optimal $O(1/\sqrt{n})$ learning bound

Regularization by stochastic optimization

control statistics and time at once

What about other optimization methods?
Beyond gradient descent

What about other optimization methods?

- Accelerated methods (Conjugate Gradient, Nesterov, Heavyball).

- Stochastic methods (SGD).
SGD with minibatches

\[ \hat{w}_{t+1} = \hat{w}_t - \gamma_t \frac{1}{b} \sum_{i=b(t-1)+1}^{bt} \left( \langle \hat{w}_t, \Phi(x_{j_i}) \rangle - y_{j_i} \right) \Phi(x_{j_i}) \]

- \( b \) minibatch size (\( b = 1 \) SGD, \( b = n \) GD)
- \( t = \left\lceil \frac{n}{b} \right\rceil \) one pass
Theorem (Lin R. ’16)

If \( p = \infty, \|\Phi(x)\|, |y| \leq 1 \) a.s. if

1. \( b = 1, \gamma_t \simeq \frac{1}{\sqrt{n}}, \) and \( t = n \) iterations (1 pass over the data);
2. \( b = \sqrt{n}, \gamma_t \simeq 1, \) and \( t = \sqrt{n} \) iterations (1 pass over the data);
3. \( b = n, \gamma_t \simeq 1, \) and \( t = \sqrt{n} \) iterations (\( \sqrt{n} \) passes over the data);

then,

\[
\mathbb{E}[(\langle \Phi(x), \hat{w}_t \rangle - \langle \Phi(x), w_* \rangle)^2] \lesssim \frac{1}{\sqrt{n}}.
\]
Theorem (Lin R. ’16)

If \( p = \infty, \|\Phi(x)\|, |y| \leq 1 \) a.s. if

1. \( b = 1, \gamma_t \simeq \frac{1}{\sqrt{n}}, \) and \( t = n \) iterations (1 pass over the data);
2. \( b = \sqrt{n}, \gamma_t \simeq 1, \) and \( t = \sqrt{n} \) iterations (1 pass over the data);
3. \( b = n, \gamma_t \simeq 1, \) and \( t = \sqrt{n} \) iterations (\( \sqrt{n} \) passes over the data);

then,

\[
\mathbb{E}[\langle \Phi(x), \hat{w}_t \rangle - \langle \Phi(x), w_* \rangle]^2 \lesssim \frac{1}{\sqrt{n}}.
\]

Proof

\[ \forall \gamma > 0, t > 1, \quad \mathbb{E}[\langle \Phi(x), \hat{w}_t \rangle - \langle \Phi(x), w_* \rangle]^2 \lesssim \frac{1}{\gamma t} + \frac{1}{\sqrt{n}} \left( \frac{\gamma t}{\sqrt{n}} \right)^2 + \frac{\gamma}{b} \left( 1 + \frac{\gamma t}{\sqrt{n}} \right) \]
Other Flavors of SGD

- Cyclic, reshuffle SGD (R. Villa, ’15)

- SGD with averaging (Bach et al. 14-..., Pillaud, Rudi, Bach ’18)- larger step-size.

- Averaging as regularization (Neu, R. ’18)
Computational regularization

- Time $O(n^2)$ + space $O(n^2)$ for optimal $O(1/\sqrt{n})$ learning bound

Regularization by stochastic optimization

- Improved time while keeping optimal statistical error

What about memory costs?
Outline

Classical regularization

Regularization by optimization

Regularization & Projections

Preconditioning
Tackling memory with random projections

- Sketching & random features
- Nyström methods & subsampling
Subsampling aka Nyström methods

Consider

$$\bar{x}_1, \ldots, \bar{x}_M \subset x_1, \ldots, x_n$$

and

$$w_M = \sum_{j=1}^{M} \Phi(\bar{x}_j)c_j,$$

a random projection on a subspace.

Connections to
- Nyström methods
- Galerkin methods
- Column subsampling
Linear algebra perspective

RandNLA=randomized numerical linear algebra (Halko et al. '11)

Back to ridge regression.

\[
\langle \hat{w}_\lambda, \Phi(x) \rangle = \sum_{i=1}^{n} \langle \Phi(x_i), \Phi(x) \rangle c^i \\

\quad c = \left( \hat{\Phi} \hat{\Phi}^T + \lambda n I \right)^{-1} \hat{y}
\]

Linear System

\[
\begin{align*}
\hat{G} & \quad c = \hat{y}
\end{align*}
\]
Nyström/Column subsampling

Take $\overline{x}_1 \ldots, \overline{x}_M \subset x_1, \ldots, x_n$, $M < n$.

$$
\langle \hat{w}_{\lambda, M}, \Phi(x) \rangle = \sum_{i=1}^{M} \langle \Phi(\overline{x}_i), \Phi(x) \rangle c^i
$$

$$
(\hat{G}_{nM}^T \hat{G}_{nM} + \lambda n \hat{G}_{MM}) c = \hat{G}_{nM}^T \hat{y}
$$

Linear System
Statistical guarantees

Theorem (Rudi, Camoriano, R. '16)

If $p = \infty$, $\|\Phi(x)\|, |y| \leq 1 \text{ a.s. if}$

$$\lambda = \frac{1}{\sqrt{n}}, \quad M = \sqrt{n}$$

then,

$$\mathbb{E}[(\langle \Phi(x), \hat{w}_{\lambda,M} \rangle - \langle \Phi(x), w_* \rangle)^2] \lesssim \frac{1}{\sqrt{n}}$$

Remarks:

- Same bound again...
- Improve previous bounds (Bach et al. '12, Alaoui, Mahoney '14)
- Regularization by projection!
Computational regularization

\[
time O(n^2) + \text{space } O(n^{\sqrt{n}}) \text{ for optimal } O(1/\sqrt{n}) \text{ learning bound}
\]

Regularization by projection

control statistics, time and memory costs at once

Can we improve computational costs?
Outline

Classical regularization

Regularization by optimization

Regularization & Projections

Preconditioning
**Preconditioning**

**Idea:** define equivalent linear system with better condition number

\[
(\hat{G} + \lambda_n I) c \rightarrow \rightarrow B^\top (\hat{G} + \lambda_n I) B \beta = B^\top \hat{y}, \quad c = B\beta.
\]

Ideally $BB^\top = (\hat{G} + \lambda_n I)^{-1}$, so that $t = O(1/\lambda) \rightarrow t = O(1)!$

(Fasshauer et al '12, Avron et al '16, Cutajar '16, Ma, Belkin '17)
**Preconditioning**

**Idea:** define equivalent linear system with better condition number

\[(\hat{G} + \lambda nI)c = \hat{y} \implies B^\top (\hat{G} + \lambda nI)B\beta = B^\top \hat{y}, \quad c = B\beta.\]

Ideally \(BB^\top = (\hat{G} + \lambda nI)^{-1}\), so that

\[t = O(1/\lambda) \implies t = O(1)!

(Fasshauer et al ’12, Avron et al ’16, Cutajat ’16, Ma, Belkin ’17)
Recall Nyström

\[(\hat{G}_{nM}^\top \hat{G}_{nM} + \lambda n \hat{G}_{MM}) c = \hat{G}_{nM}^\top \hat{y}\]

Preconditioning

\[BB^\top = \left( \frac{n}{M} \hat{G}_{MM}^2 + \lambda n \hat{G}_{MM} \right)^{-1},\]

Baby FALKON

\[\langle \hat{w}_{\lambda,M,t} , \Phi(x) \rangle = \sum_{i=1}^{M} \langle \Phi(\tilde{x}_i), \Phi(x) \rangle \ c^i \quad \quad c_t = B \beta_t\]

\[\beta_t = \beta_{t-1} - \frac{T}{n} B^\top \left[ \hat{G}_{nM}^\top (\hat{G}_{nM} B \beta_{t-1} - y_n) + \lambda n \hat{G}_{MM} B \beta_{t-1} \right]\]
Gradient descent $\mapsto$ conjugate gradient

Computing $B$

\[
B = \frac{1}{\sqrt{n}} T^{-1} A^{-1}, \quad T = \text{chol}(G_{MM}), \quad A = \text{chol}\left(\frac{1}{M} T T^\top + \lambda I\right),
\]

where $\text{chol}(\cdot)$ is the Cholesky decomposition.
Theorem (Rudi, Carratino, R. '17)

If \( p = \infty, \|\Phi(x)\|, |y| \leq 1 \) a.s. if

\[
\lambda = 1/\sqrt{n}, \quad M = \sqrt{n}, \quad t = \log n
\]

then

\[
\mathbb{E}[\left(\langle \Phi(x), \hat{w}_{\lambda,M,t} \rangle - \langle \Phi(x), w_* \rangle \right)^2] \leq \frac{1}{\sqrt{n}}
\]

Remarks:

▶ Same bound again... improved time cost!
▶ Improved results by considering adaptive sampling.
Computational regularization

Time $O(n\sqrt{n})$ + space $O(n\sqrt{n})$ for optimal $O(1/\sqrt{n})$ learning bound

Maybe optimal?
## Some experiments

<table>
<thead>
<tr>
<th></th>
<th>MillionSongs</th>
<th></th>
<th></th>
<th></th>
<th>YELP</th>
<th></th>
<th></th>
<th>TIMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSE</td>
<td>Relative error</td>
<td>Time(s)</td>
<td>RMSE</td>
<td>Time(m)</td>
<td>c-err</td>
<td>Time(h)</td>
<td></td>
</tr>
<tr>
<td>FALKON</td>
<td>80.30</td>
<td>$4.51 \times 10^{-3}$</td>
<td>55</td>
<td>0.833</td>
<td>20</td>
<td>32.3%</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>Prec. KRR</td>
<td>-</td>
<td>$4.58 \times 10^{-3}$</td>
<td>289†</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Hierarchical</td>
<td>-</td>
<td>$4.56 \times 10^{-3}$</td>
<td>293*</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>D&amp;C</td>
<td>80.35</td>
<td>-</td>
<td>737*</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Rand. Feat.</td>
<td>80.93</td>
<td>-</td>
<td>772*</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Nyström</td>
<td>80.38</td>
<td>-</td>
<td>876*</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>ADMM R. F.</td>
<td>-</td>
<td>$5.01 \times 10^{-3}$</td>
<td>958†</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>BCD R. F.</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.949</td>
<td>42‡</td>
<td>34.0%</td>
<td>1.7‡</td>
<td></td>
</tr>
<tr>
<td>BCD Nyström</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.861</td>
<td>60‡</td>
<td>33.7%</td>
<td>1.7‡</td>
<td></td>
</tr>
<tr>
<td>KRR</td>
<td>-</td>
<td>$4.55 \times 10^{-3}$</td>
<td>-</td>
<td>0.854</td>
<td>500‡</td>
<td>33.5%</td>
<td>8.3‡</td>
<td></td>
</tr>
<tr>
<td>EigenPro</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>32.6%</td>
<td>3.9‡</td>
<td></td>
</tr>
<tr>
<td>Deep NN</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>32.4%</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Sparse Kernels</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>30.9%</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Ensemble</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>33.5%</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

**Table:** MillionSongs, YELP and TIMIT Datasets. Times obtained on: † = cluster of 128 EC2 r3.2xlarge machines, ‡ = cluster of 8 EC2 r3.8xlarge machines, ‡ = single machine with two Intel Xeon E5-2620, one Nvidia GTX Titan X GPU and 128GB of RAM, * = cluster with 512 GB of RAM and IBM POWER8 12-core processor, ‡ = unknown platform.
Some more experiments

<table>
<thead>
<tr>
<th></th>
<th>SUSY</th>
<th>HIGGS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>c-err</td>
<td>AUC</td>
</tr>
<tr>
<td>FALKON</td>
<td>19.6%</td>
<td>0.877</td>
</tr>
<tr>
<td>EigenPro</td>
<td>19.8%</td>
<td>-</td>
</tr>
<tr>
<td>SVM</td>
<td>26.4%</td>
<td>-</td>
</tr>
<tr>
<td>Hierarchical</td>
<td>20.1%</td>
<td>-</td>
</tr>
<tr>
<td>Boosted Decision Tree</td>
<td>-</td>
<td>0.863</td>
</tr>
<tr>
<td>Neural Network</td>
<td>-</td>
<td>0.875</td>
</tr>
<tr>
<td>Deep Neural Network</td>
<td>-</td>
<td>0.879</td>
</tr>
</tbody>
</table>

Table: SUSY and HIGGS Datasets. Time obtained working on: † = cluster with 512 GB of RAM and IBM POWER8 12-core processor, ‡ = single machine with two Intel Xeon E5-2620, one Nvidia GTX Titan X GPU and 128GB of RAM, * = 14 workers.
Image classification

\[ f(x) = \langle w, \Phi(x) \rangle, \quad x \mapsto \Phi_L \circ \Phi_{L-1} \cdots \circ \Phi_1(x) \]

Kernel representation  Convoluational

Imagenet

<table>
<thead>
<tr>
<th></th>
<th>Top-1 class error</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FALKON + I-v3 feat.</strong></td>
<td>22.1%</td>
</tr>
<tr>
<td>Inception-v3</td>
<td>21.2%</td>
</tr>
<tr>
<td>Inception-v2</td>
<td>23.4%</td>
</tr>
<tr>
<td>BN-Inception</td>
<td>25.2%</td>
</tr>
<tr>
<td>BN-GoogLeNet</td>
<td>26.8%</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>29.0%</td>
</tr>
</tbody>
</table>

Table: Single crop experimental results on the validation set of ILSVRC 2012.
In Sec. III-D with the Faster R-CNN baseline.

We adopt a sampling strategy to further reduce the number of positives with respect to the previous bootstrapping iteration. This practice is in order to account for the positive-negative imbalance.

We cross-validated these latter two using a one-fold cross-validation approach proposed in Sec. III-D, by varying the number of mini-batches of negatives.

In these experiments we set the number of approximated hard negatives mining.

We observe that just performing a few bootstrapping iterations could allow to achieve comparable performance than fine-tuning the last layers of Faster R-CNN's last layers (which however requires a batch of 2500)

In Table II we compare the performance provided by two output layers for class prediction and bounding box regression centers equal to the number of training points, since higher values did not improve performance, and (i) the size of the mini-batches of negatives M

We provide experimental evaluation of the impact of the other fundamental because, when

We selected, from the VOC 2007

The second component of

It can be clearly noticed that, while FALKON + R

Based on the empirical observations from Sec. IV-A, in

In the considered scenario, we train Faster R-CNN on the

In Table II we compare the performance provided by

For this step we set the number of iterations to 15K.

In Table II we compare the performance provided by

We focus on the classifier algorithm (Sec. III-C), by explaining our

and feature extractor, on PREV-TASK, such that,

and (ii) the feature extractor, on PREV-TASK, such that,

We report two different configurations of FALKON + M

We consider the target task on which we learn the

For the definition of our pipeline, we considered the

It can be clearly noticed that, while FALKON + R

Based on the empirical observations from Sec. IV-A, in

In Table II we compare the performance provided by

We focus on the classifier algorithm (Sec. III-C), by explaining our

and feature extractor, on PREV-TASK, such that,

We report two different configurations of FALKON + M

The second component of

It can be clearly noticed that, while FALKON + R

Based on the empirical observations from Sec. IV-A, in

In Table II we compare the performance provided by

We focus on the classifier algorithm (Sec. III-C), by explaining our

and feature extractor, on PREV-TASK, such that,
Summing up

- Computational regularization for efficient learning.
- Faster GP/Kernel solver ever.

Looking ahead
- Other loss functions, norms, learning problems . . . .
- Parallelization.
- Non convex problems.
- Optimal complexity.

check papers on arxiv.org