Extreme multilabel learning

Charles Elkan

Amazon Fellow

December 12, 2015

Massive multilabel classification

In the world of big data, it is common to have

- many training examples (10^6 instances)
- high-dimensional data (10^6 features)
- many labels to predict ($10^{4.5}$ labels)

Numbers are for predicting medical subject headings (MeSH) for documents in PubMed.

Amazon datasets are far larger.

We are not perfect at Amazon...

Best Sellers in **Fire Pit Spark Screens**

1.

36" Firepit Screen

(2)

$169.95

3 used & new from $132.66

2.

Asia Direct 28 inch Easy Access Spark...

(16)

$55.54

4 new from $55.54

3.

Sunnydaze Fire Pit Spark Screen, Stee...

(15)

$149.95

7 new from $149.95

4.

Amazon Fire 7 case, SAVYOU Ultra Slim...

5.

Sunnydaze Fire Pit Spark Screen, 31 I...

6.

Sunnydaze Easy Access Fire Pit Spark...

They aren't perfect at PubMed...

BMC Neurosci. 2008 Sep 18;9:88. doi: 10.1186/1471-2202-9-88.

Expression of ATF3 and axonal outgrowth are impaired after delayed nerve repair.

Saito H[1], Dahlin LB.

⊕ Author information

Abstract

BACKGROUND: A delay in surgical nerve repair results in impaired nerve function in humans, but mechanisms behind the weakened nerve regeneration are not known. Activating transcription factor 3 (ATF3) increases the intrinsic growth state of injured neurons early after injury, but the role of long-term changes and their relation to axonal outgrowth after a delayed nerve repair are not well understood. ATF3 expression was examined by immunohistochemistry in motor and sensory neurons and in Schwann cells in rat sciatic nerve and related to axonal outgrowth after transection and delayed nerve repair (repair 0, 30, 90 or 180 days post-injury). Expression of the neuronal cell adhesion molecule (NCAM), which is expressed in non-myelinating Schwann cells, was also examined.

RESULTS: The number of neurons and Schwann cells expressing ATF3 declined and the length of axonal outgrowth was impaired if the repair was delayed. The decline was more rapid in motor neurons than in sensory neurons and Schwann cells. Regeneration distances over time correlated to number of ATF3 stained neurons and Schwann cells. Many neurofilament stained axons grew along ATF3 stained Schwann cells. If nerve repair was delayed the majority of Schwann cells in the distal nerve segment stained for NCAM.

CONCLUSION: Delayed nerve repair impairs nerve regeneration and length of axonal outgrowth correlates to ATF3 expression in both neurons and Schwann cells. Mainly non-myelinating Schwann cells (NCAM stained) are present in distal nerve segments after delayed nerve repair. These data provide a neurobiological basis for the poor outcomes associated with delayed nerve repair. Nerve trunks should, if possible, be promptly repaired.

PMID: 18801180 [PubMed - indexed for MEDLINE] PMCID: PMC2556676 **Free PMC Article**

Images from this publication. See all images (3) Free text

Publication Types, MeSH Terms, Substances

Publication Types

Research Support, Non-U.S. Gov't

MeSH Terms

Activating Transcription Factor 3/metabolism*
Animals
Axons/metabolism
Axons/physiology*
Female
Immunohistochemistry

So, how good are humans?

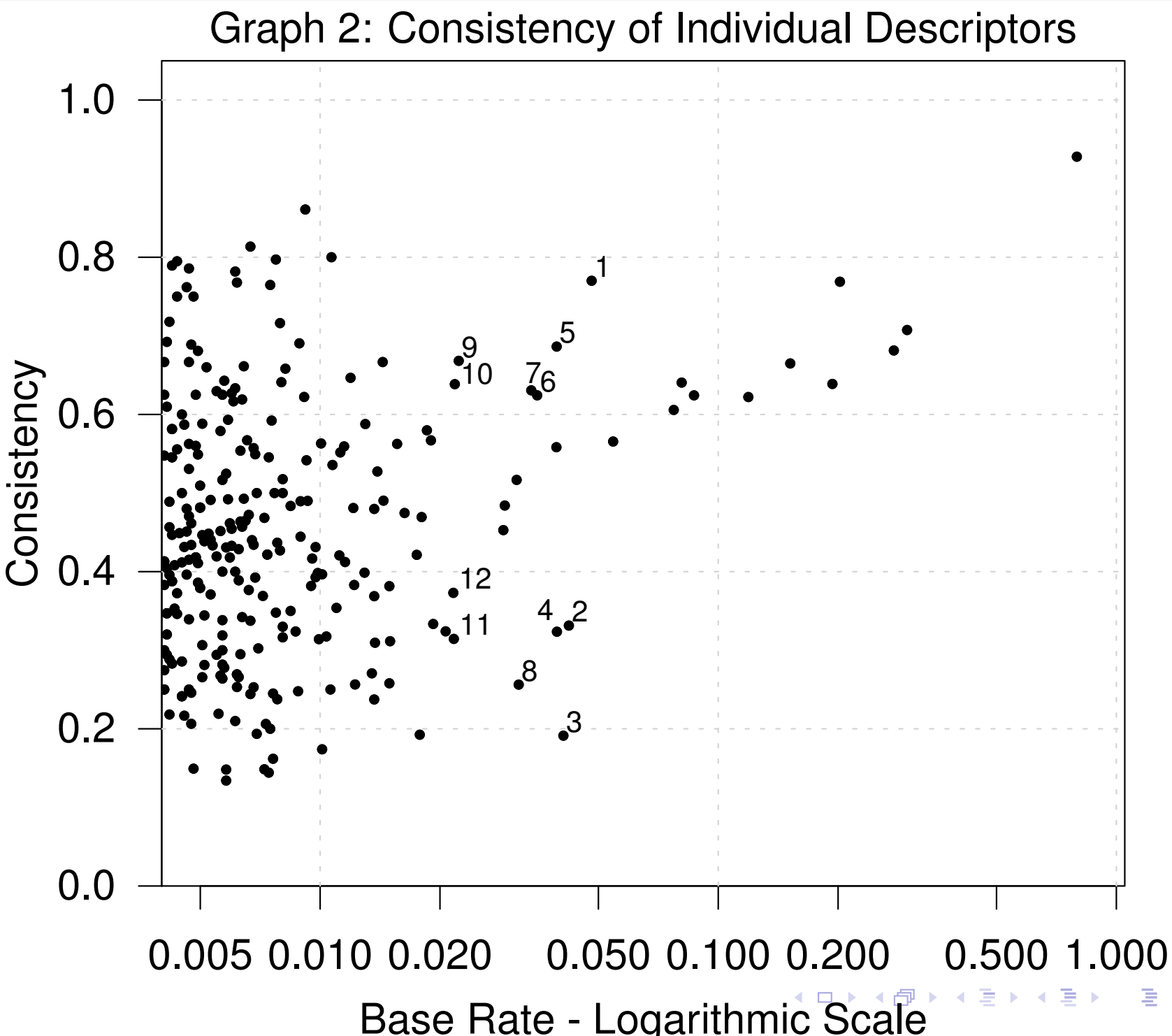
NIH can only afford to assign one human indexer per document.

How can we measure how accurate the humans are?

Method: Look for articles that were inadvertently indexed twice.

- Finding: About 0.1% of PubMed articles are duplicates, usually not exact.

Causes: Primarily plagiarism and joint issues of journals.


Graph 2: Consistency of Individual Descriptors
Consistency
1.0
0.8
0.6
0.4
0.2
0.0
0.005 0.010 0.020 0.050 0.100 0.200 0.500 1.000
Base Rate - Logarithmic Scale
1
5
9
10
7
6
12
4
2
11
8
3

Most frequent MeSH terms

Consistency for concrete terms is better than for abstract terms.

Descriptor Name	**Consistency (%)**	**95% ±**	**Base Rate (%)**
Humans	92.80	0.62	79.58
Female	70.74	1.69	29.81
Male	68.14	1.78	27.61
Animals	76.89	1.93	20.21
...			
Time Factors	19.13	3.29	4.08

Challenges

At first sight, the training dataset contains 10^{12} values.

- Can it fit in memory? (Yes–easy given sparsity.)
- What if the dataset is stored in distributed fashion?

But:

- Storing dense linear classifiers for $10^{4.5}$ labels with 10^6 features would need 200 gigabytes.

Achieving tractability

Training is feasible on a single CPU core if we have

- Sparse features ($10^{2.5}$ nonzero features per instance)
- Sparse labels ($10^{1.5}$ positive labels per instance)
- Sparse models (10^{3} features are enough for each label).

Note: Class imbalance is a non-problem for logistic regression and related methods.

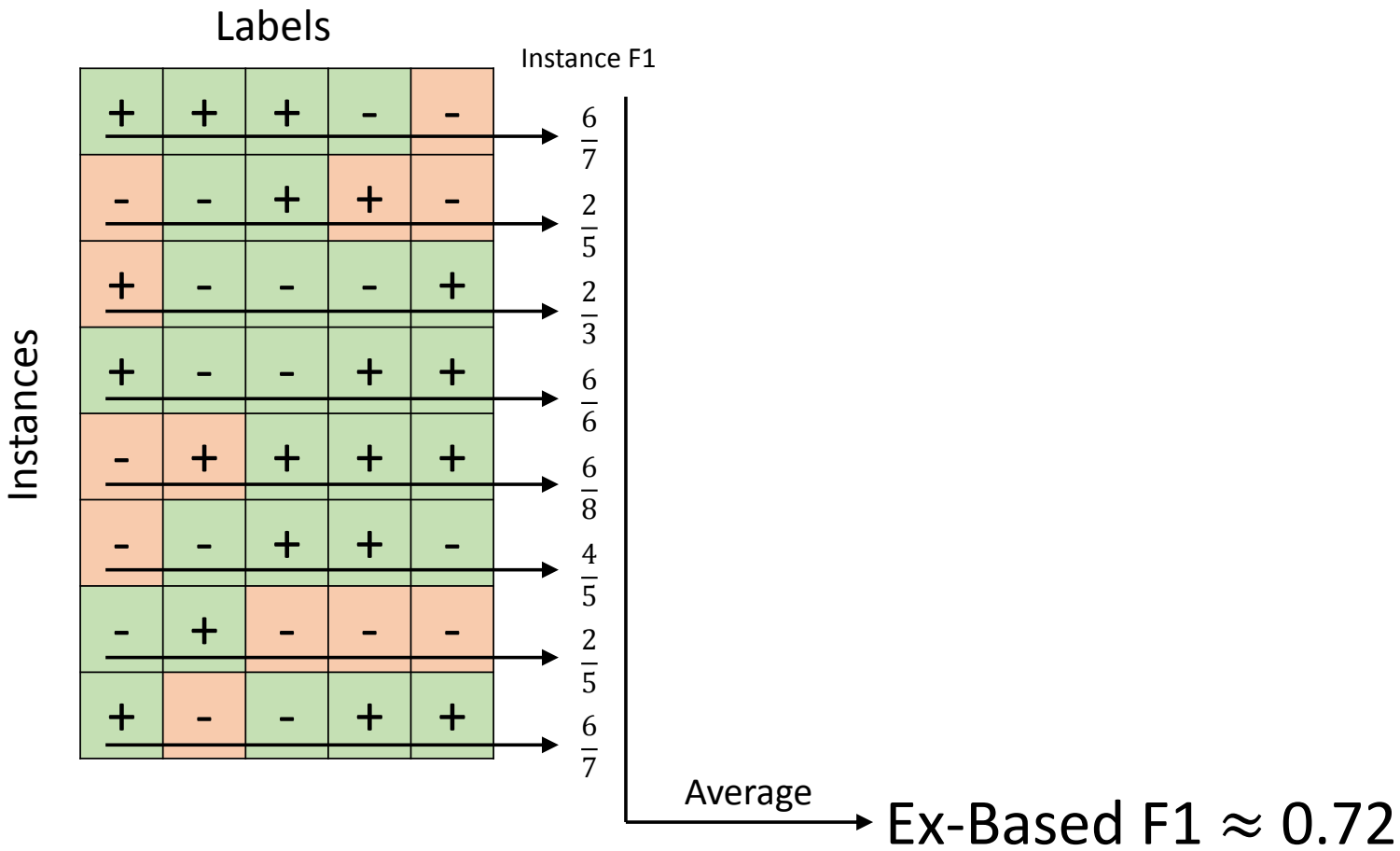
- Here, a typical class has only $10^{1.5}/10^{4.5} = 0.1\%$ positives.

How do we evaluate success?

We use F1 measure $F = \frac{2}{1/P+1/R} = \frac{2tp}{2tp+fp+fn}$.

- Why does F1 not depend on the number *tn* of true negatives?
- Intuition: For any label, most instances are negative, so give no credit for correct predictions that are easy.

Example-based F1: average of F1 for each document



Average F1 per document reflects the experience of a user who examines the positive predictions for some specific documents.

How to optimize F1 measure?

ECML 2014 paper with Z. Lipton and B. Narayanaswamy
Optimal Thresholding of Classifiers to Maximize F1 Measure.

- **Theorem:** The probability threshold that maximizes F1 is one half of the maximum achievable F1.

We can apply the theorem separately for any variant of F1.

Lessons from previous research

1. Correlations between labels are not highly predictive.
2. Optimizing the right measure of success is important.
3. Keeping rare features is important for predicting rare labels.
   - Need the word “platypus" to predict label “monotreme."
4. Standard bag-of-words preprocessing is hard to beat.
   - Use $\log(tf + 1) \cdot idf$ and L2 length normalization.

Tractability

Two ideas to achieve tractability in training:

1. Use a loss function that promotes sparsity of weights.
2. Design the training algorithm to never lose sparsity.

On PubMed data, only 0.3% of weights are ever non-zero during training.

Example of a trained sparse PubMed model

Features with the largest squared weights.

Earthquakes	
earthquake A	1.37
earthquake T	0.99
fukushima A	0.34
earthquakes A	0.30
disaster A	0.29
Disasters J	0.18
haiti A	0.18
wenchuan T	0.18
disasters A	0.17
wenchuan A	0.16
(remaining mass)	0.14

A = word in abstract, T = word in title, J = exact journal name.

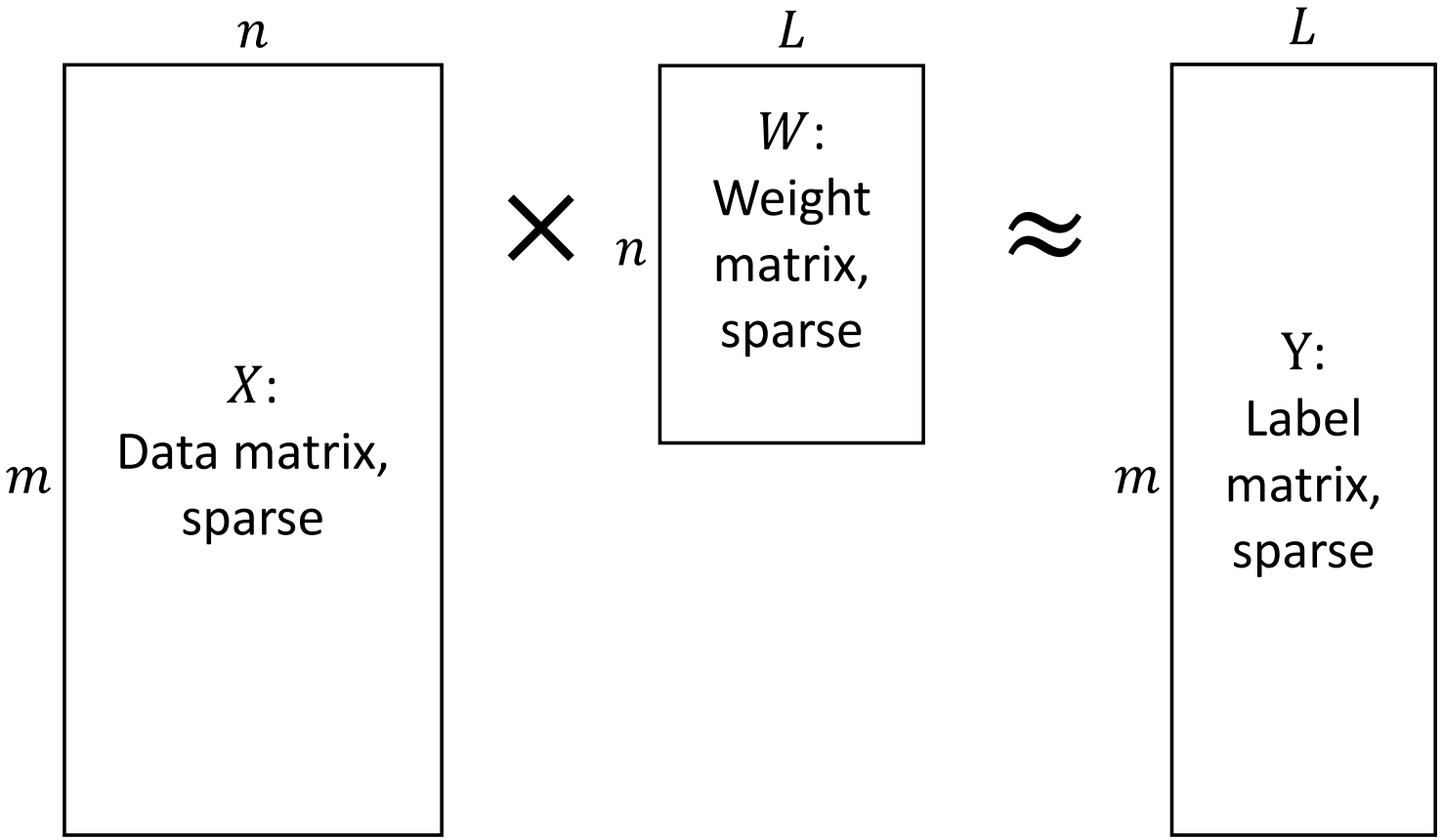
This model has perfect training and test accuracy.

The proposed method

To solve massive multilabel learning tasks:

1. Linear or logistic regression
2. Training the models for all labels simultaneously
3. Combined L1 and L2 regularization (elastic net)
4. Stochastic gradient descent (SGD)
5. Proximal updates delayed and applied only when needed.
6. Sparse data structures

Multiple linear models



Use L1 regularization to find sparse W to minimize discrepancy between $f(XW)$ and labels Y.

Weight sparsity

It is wasteful to learn dense weights when only a few non-zero weights are needed for good accuracy.

Elastic net regularizer sums L_1 and squared L_2 penalties:

$$R(W) = \sum_{l=1}^{L} \lambda_1 ||w_l||_1 + \frac{\lambda_2}{2} ||w_l||_2^2.$$

Like pure L_1, eliminates non-predictive features; like pure L_2^2, spreads weights over correlated predictive features.

Proximal stochastic gradient

We want to minimize the *regularized* loss $L(w) + R(w)$, where R is analytically tractable, such as $L_1 + L_2^2$.

Define $\text{prox}_Q(\tilde{w}) = \arg\min_{w \in \mathbb{R}^d} \frac{1}{2}||w - \tilde{w}||_2^2 + Q(w)$.

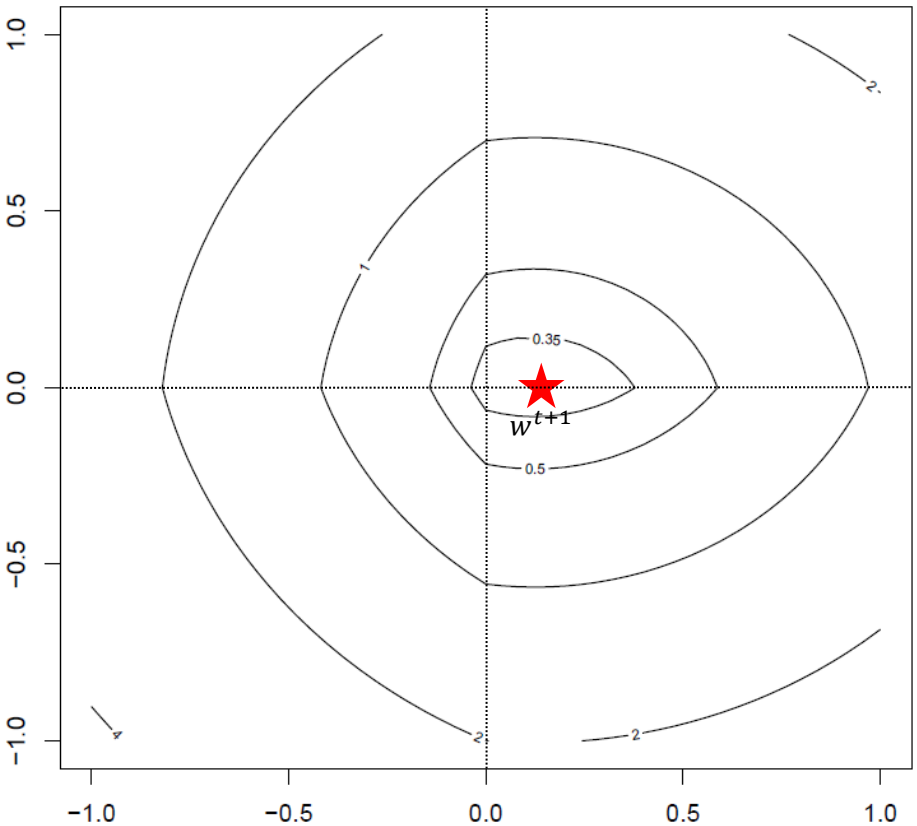
Then $w^{t+1} = \text{prox}_Q(w^t - \eta g^t)$ where $Q = \eta R$.

The proximal operator balances two objectives:

1. staying close to the SG-updated weight vector $w^t - \eta g^t$
2. moving toward a weight vector with lower value of R.

Proximal step with L_1 plus L_2^2

Minimum of the sum of the proximity and regularizer functions:



$$w^{t+1} = \text{prox}_{\eta^t R}(w^t - \eta^t g^t)$$
$$= \arg\min M^t(w) + R(w)$$

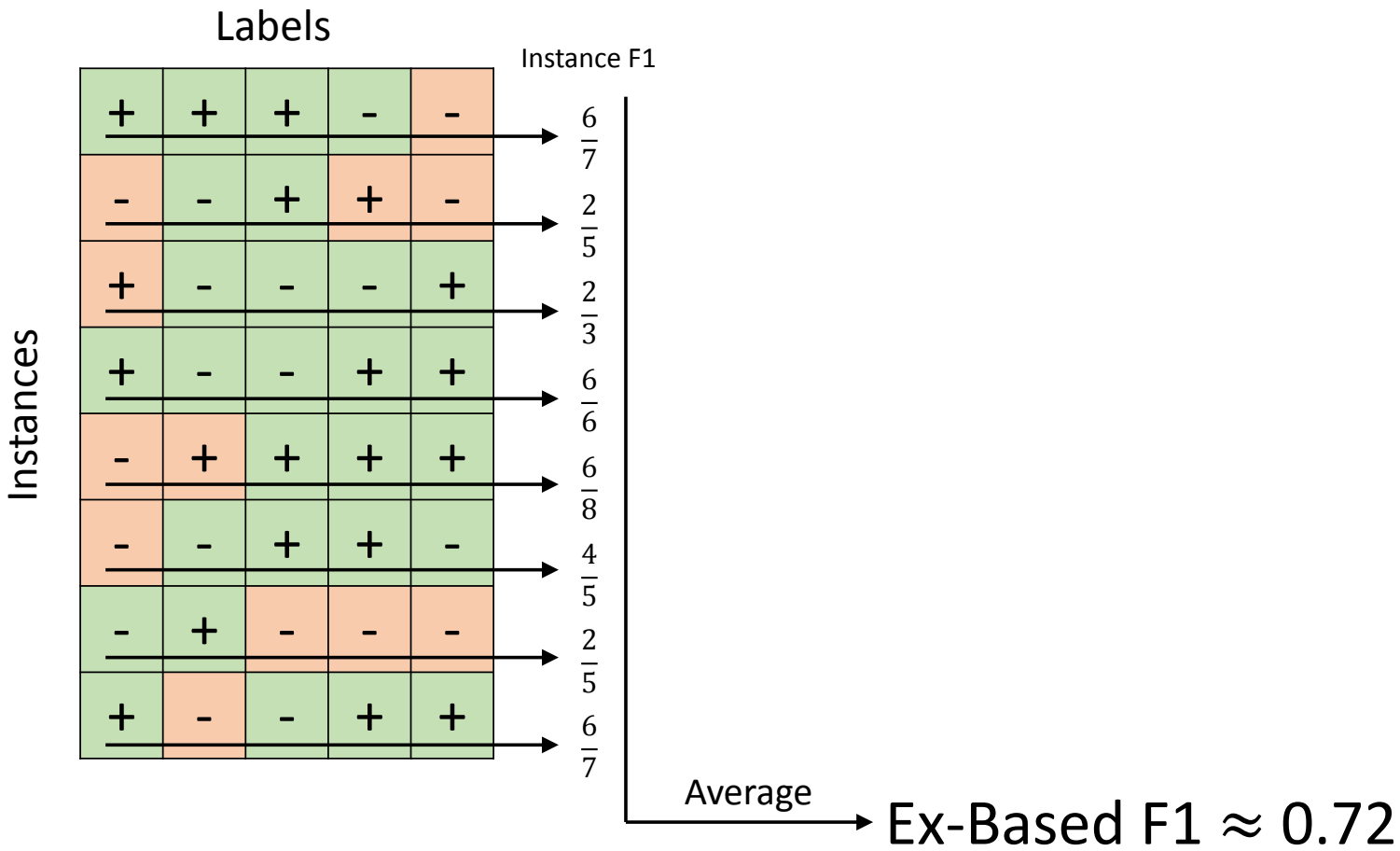
Experiments with PubMed articles

- 1,125,160 training instances—all articles since 2011
- 969,389 vocabulary words
- 25,380 labels, so 2.5×10^{10} potential weights.

TF-IDF and L2 bag-of-words preprocessing.

- AdaGrad multiplier α fixed to 1.
- L_1 and L_2 regularization strengths $\lambda_1 = 3 \times 10^{-6}$ and $\lambda_2 = 10^{-6}$ chosen on small subset of training data.

Instance-based F1: average of F1 for each document



Reflects the experience of a user who looks at the positive predictions for some specific documents.

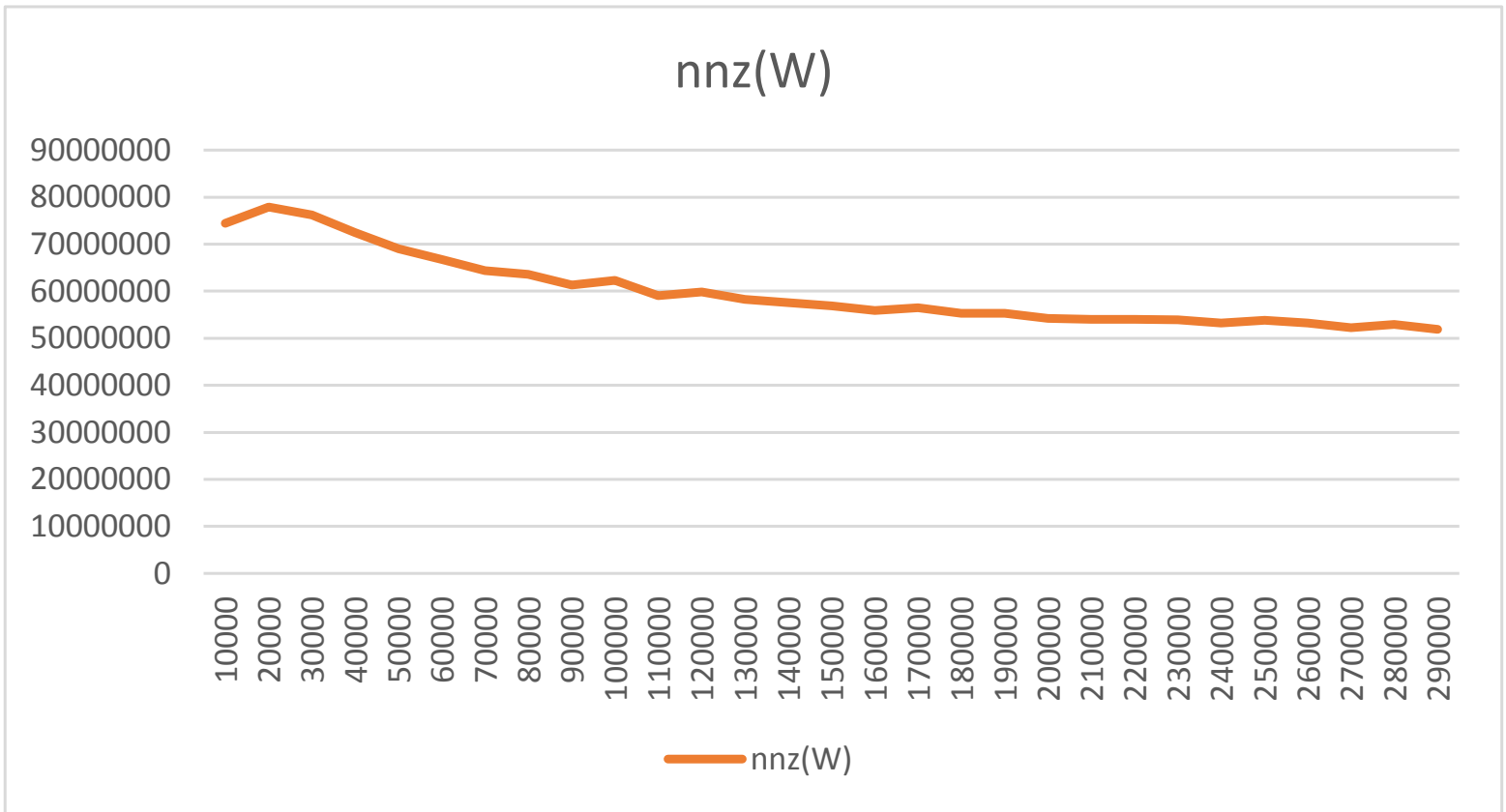
Experimental results

Fraction of labels	Per-instance F1
all	0.52
30%	0.54
10%	0.56
3%	0.59
1%	0.61

Example-based F1 computed with various subsets of the 25,380 labels, from all to the 1% of most frequent labels.

Not surprising: More common labels are easier to predict.

Sparsity during training



Of about 25 billion potential weights, during training at most 80 million are non-zero; at convergence 50 million (0.2%).

The proposed method

To solve massive multilabel learning tasks:

1. Linear or logistic regression
2. Training the models for all labels simultaneously
3. Combined L1 and L2 regularization (elastic net)
4. Stochastic gradient descent (SGD)
5. *Proximal updates delayed and applied only when needed*
6. Sparse data structures

Where to find the details

arXiv.org > cs > arXiv:1505.06449

Search o

Computer Science > Learning

Efficient Elastic Net Regularization for Sparse Linear Models

Zachary C. Lipton, Charles Elkan

(Submitted on 24 May 2015 (v1), last revised 2 Jul 2015 (this version, v3))

This paper presents an algorithm for efficient training of sparse linear models with elastic net regularization. Extending previous work on delayed updates, the new algorithm applies stochastic gradient updates to non-zero features only, bringing weights current as needed with closed-form updates. Closed-form delayed updates for the ℓ_1, ℓ_∞, and rarely used ℓ_2 regularizers have been described previously. This paper provides closed-form updates for the popular squared norm ℓ_2^2 and elastic net regularizers.
We provide dynamic programming algorithms that perform each delayed update in constant time. The new ℓ_2^2 and elastic net methods handle both fixed and varying learning rates, and both standard {stochastic gradient descent} (SGD) and {forward backward splitting (FoBoS)}. Experimental results show that on a bag-of-words dataset with $260,941$ features, but only 88 nonzero features on average per training example, the dynamic programming method trains a logistic regression classifier with elastic net regularization over 2000 times faster than otherwise.

High-level algorithm

If $x_{ij} = 0$ then the prediction $f(x_i \cdot w)$ does not depend on w_j, and the unregularized derivative wrt w_j is zero.

Algorithm 1 Using delayed updates

for $t \in 1, ..., T$ **do**
  Sample x_i randomly from the training set
  for j s.t. $x_{ij} \neq 0$ **do**
    $w_j \leftarrow \mathit{DelayedUpdate}(w_j, t, \psi_j)$
    $\psi_j \leftarrow t$
  end for
  $\boldsymbol{w} \leftarrow \boldsymbol{w} - \nabla F_i(\boldsymbol{w})$
end for

Elastic net FoBoS delayed updates

Theorem: To bring weight w_j current to time k from time ψ_j in constant time, the FoBoS update, with $L_1 + L_2^2$ regularization and learning rate $\eta^{(t)}$, is

$$w_j^{(k)} = \text{sgn}(w_j^{(\psi_j)}) \cdot \left[|w_j^{(\psi_j)}| \frac{\Phi(k-1)}{\Phi(\psi_j - 1)} - \Phi(k-1) \cdot \lambda_1 \left(\beta(k-1) - \beta(\psi_j - 1) \right) \right]_+$$

where $\Phi(t) = \Phi(t-1) \cdot \frac{1}{1+\eta^t \lambda_2}$ with base case $\Phi(-1) = 1$ and $\beta(t) = \beta(t-1) + \frac{\eta^{(t)}}{\Phi(t-1)}$ with base case $\beta(-1) = 0$.

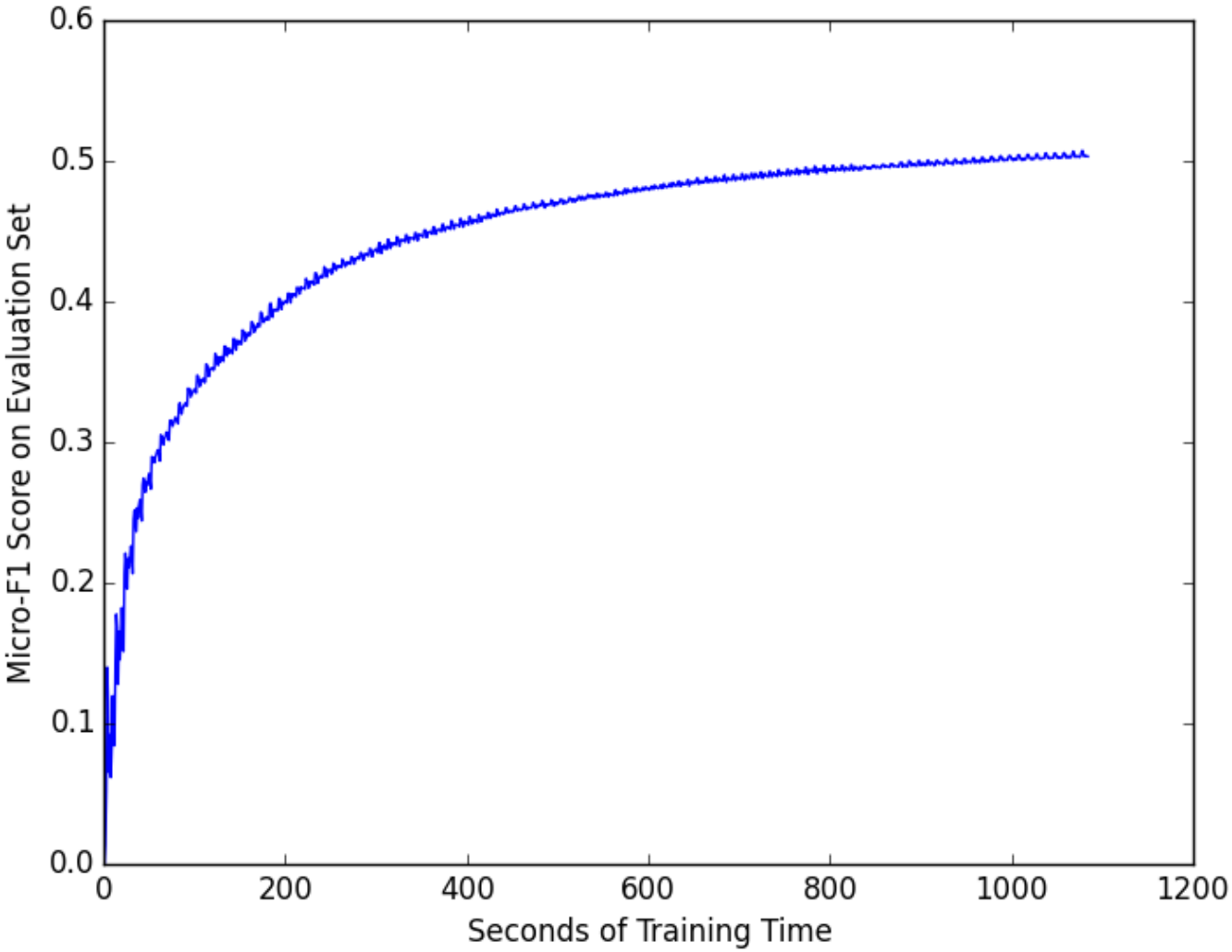
Small timing experiments

Datasets			
Dataset	Examples	Features	Labels
rcv1	30,000	47,236	101
bookmarks	87,856	2150	208

Speed in Julia on one core			
Dataset	Delayed (xps)	Standard (xps)	Speedup
rcv1	555	1.13	489.7
bookmarks	516.8	25	20.7

Speed in examples per second (xps).

Timing experiments



On the *rcv1* dataset, 101 models train in minutes on one core.

Conclusion

To learn one linear model:

With n examples, d dimensions, and e epochs, standard SGD-based methods use $O(nde)$ time and $O(d)$ space.

With d' average nonzero features per example and v nonzero weights per model, we use $O(nd'e)$ time and $O(v)$ space.

Let n' be the average number of positive examples per label. Future work: Use only $O(n'd'e)$ time.

Questions? Discussion?