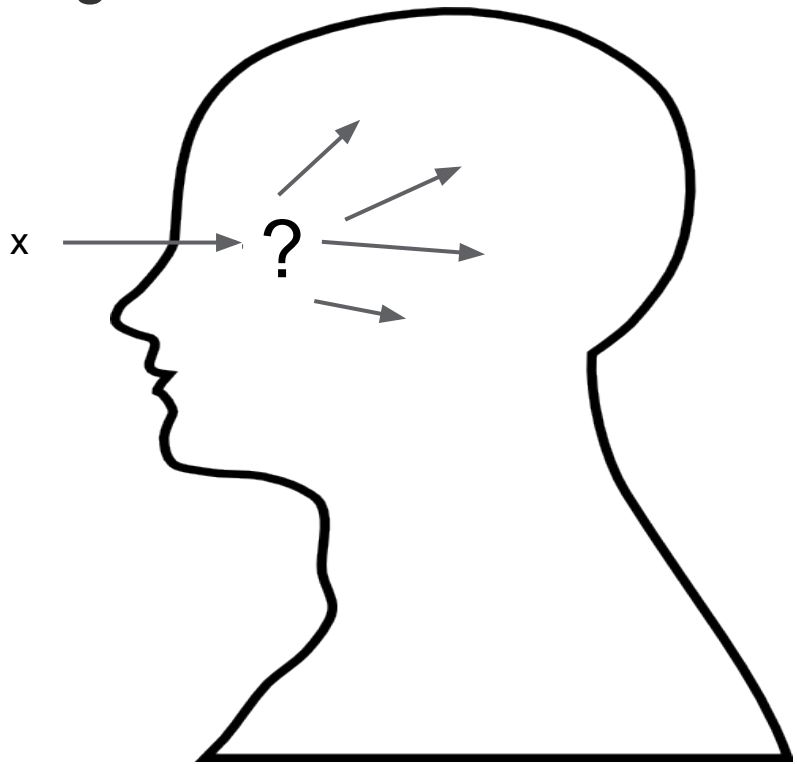
An Empirical Comparison of Sparse vs. Embedding Techniques on Many-Class Text Classification

Akshay Balasubramani and Omid Madani

The Problem of Rapid Classification (and Learning) in the Face of Myriad Categories



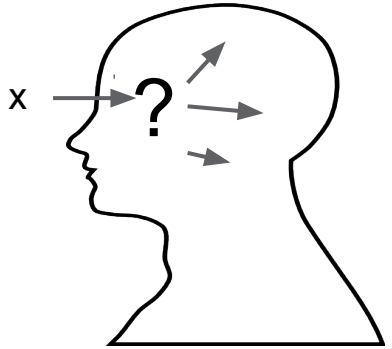
Motivation/Setting

- Many applications involve large output spaces, or ‘classes’ to predict into:
  - Text classification (web page, queries, phrases, ..), images/videos, e. g.

“Multi-Label Learning with Millions of Labels: Recommending Advertiser Bid Phrases for Web Pages”, WWW2013, Agrawal et. al. (on bid phrase recommendation)

- Many considerations in classifier design:
  - Accuracy vs. scalability (memory/time during training/prediction)
  - Distributed training/classification
  - Particularities: feature & class dimensionalities, average vector length, number of labels per instance, availability of a taxonomy, …
  - A variety of approaches have been explored ..
- Here: compare 2 linear ranking techniques, Wsabie and Indexing

Sparse Index Learning (*Indexing*)



- Domains such as text:
  - Large feature dimensionalities (e.g. millions)
  - Instances are sparse (e.g. 10s to 100s of nonzeros)
- Features are predictive:
  - Each feature needs to predict few classes (e.g. 10s of classes)
  - Motivates inducing predictive and sparse features! (via deep learning?)
- Aim to learn a sparse classifier, an *index:* Each feature predicts (connects to or indexes) relatively few classes
  - Fast training and classification time (~O(|x|d), with d connections per feature, inverted index implementation)
- Online/incremental: Feature and class sets can be dynamic/growing

Indexing: Prune + Rank

- Past work compares to one-versus-rest, top-down, nearest neighbors, etc:
  - “Learning when Concepts Abound”, JMLR 2009, Madani, et. al.
  - “Large-Scale Many-Class Prediction via Flat Techniques.” O. Madani and J. Huang. PASCAL Challenge on Hierarchical Text Classification, 2009.
  - Also: “Learning to Rank with (a Lot of) Word Features”, Bai et al, 2010.
    - where for document ranking, Wsabie did bit better than pruning variants, depending on pruning level. See also Wsabie papers on image annotation.
- This work:
  - Multiclass margin-based perceptron with periodic pruning, comare in text classification

Some Set Up and Notation

$F \equiv$ *Feature set, huge, eg millions and beyond*

$Y \equiv$ *Class set, huge, eg 10s of thousands and beyond, often* $|Y| << |F|$

$x \equiv$ *vector corresponding to an instance,* $|x|$ *is number of nonzeros, often sparse, eg 10s*

$D \equiv$ *dataset (large, millions and beyond)*

$y(x) \equiv$ *true classes/labels of x, possibly multiple (handful) and noisy*

$W \equiv$ *weight matrix to learn in indexing, rows are classes, columns are features, each columns is kept sparse when learning*

Wx *is scoring of classes for x, implemented efficiently*

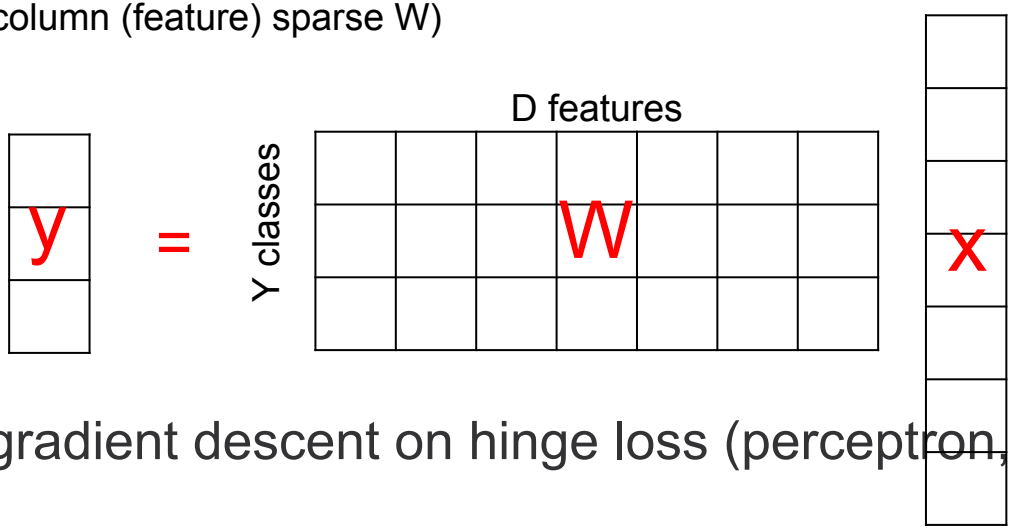
$score(x, y) \equiv$ *score of class y for x*

Indexing

- Uses a linear prediction model:

prediction: $\vec{y} = W\vec{x}$ (Learn a column (feature) sparse W)

$$y = \arg\max_{i \in 1,\ldots,Y} y^i$$
$$= \arg\max_{i \in 1,\ldots,Y} W^i \vec{x}$$



- Trained with stochastic gradient descent on hinge loss (perceptron, SVM) + periodic pruning

Training Multiclass Perceptron + Pruning

- Begin with empty W (implemented as a collection of hash maps, one per feature)
- Repeat, on each vector x:
  - Score/rank classes (Wx)
  - Take a true label y in y(x) (at random)
  - If margin not met on y, score(x,y) <= 0:
    - update W (affects only non-zero features of x):
      - Add x to W_y(x), subtract x from W_y'
  - Prune connections of features in x (when necessary)

- Pruning:
  - If connections (nonzero weight) of feature f is higher than d2 then reduce to d1: keep highest d1 in *magnitude*, eg d2=600, d1=300
  - Intuition: lower weights won't affect top rankings much, and features are predictive

Wsabie (Embed and Rank)

- Map both classes and instances to same lower dim space
- Learn this embedding (2 embeddings)
- Embedding dimension d, examples: 64, 128, 256, 512
  - Space consumption and classification time linearly increases with d
- Applications: recommendation as well as multiclass classification/ranking
  - Image/video classification (with dense features in particular)
- “Wsabie: Scaling Up To Large Vocabulary Image Annotation”, IJCAI 2009, J. Weston, S. Bengio, and N. Usunier

Questions/Hypotheses

- Wsabie, because it embeds, may underperform due to loss of granularity with so many classes..
  - Embedding may remove noise and uncover semantic relations
  - Sparse classifier prunes the features, which is an approximation too
  - Wsabie may work best on most frequent classes?
  - Both approaches approximate..
- Model capacity?

Text Classification Experiments

- Labels obtained from a subset of:

"Classifying YouTube channels: a practical system", Vincent Simonet, WWW (Companion Volume) 2013.

- ~170k instances for each of train and test
- ~100k classes
- Multiple labels per instance
- Vectors are titles of videos: 10s of nonzeros
  - Experiments with tags and description of videos yielded similar results
- Class distribution, skewed as expected
  - But most frequent class is < 0.001 of instances

Per-Instance Accuracies

- Wsabie, best of 64, 128, 256 (and choice of rate, margin, …)
- Both algorithms achieve best results in up to ~20 passes.

test performance	Wsabie	Indexing
R1 (recall or precision @ 1)	0.30	0.37
R5 (recall or cmc @ 5)	0.39	0.51

train performance	Wsabie	Indexing
R1	0.74	0.97
R5	0.85	0.99

Note: training performance picked at top held out performance..

Per-Class Accuracies

TP(y) = true positive for class y = number test instances of class y ranked top
FP(y) = number of test instances on which class y is ranked highest, but y *is not* a true label
FN(y) = number of test instances on which class y is *not* ranked highest, but y *is* a true label

$$\text{Recall} = TP / (TP + FN) \qquad \text{Precision} = TP / (TP + FP)$$

92000 unique classes

macro average →	Recall	Precision	Recall, top 50	Precision, top 50
Wsabie	0.086	0.095	0.232	0.323
Indexing	0.119	0.135	0.297	0.410

No evidence that the frequency of classes matters in these experiments (perhaps since there are no dominating classes in the data).

Size/Efficiency

- Test/train time much faster for indexing
  - A few minutes for indexing vs 2+ hours
  - ~1000s classes scored per instance for indexing
- Model sizes (depend on embedding dimension, and pruning parameters):
  - Wsabie: 4G
  - Indexing: 250M
  - NOTE: size of model for indexing increases with more data, eg to ~1G on millions of instances, Wsabie grows at a slower rate (fixed dim)

Findings

- Sparse index learning offers a favorable tradeoff in efficiency/accuracy compared to embedding for text classification
  - Fart training/test, with better accuracy on our data
- Future:
  - Further comparisons, and combo exploration
  - Adagrad (bookkeeping challenges, memory)
  - L1 and other regularization (bookkeeping challenges, memory)
  - Understanding various losses/objectives (e.g. noisy labels)

Acknowledgments

- Thanks to our Google teams (VCA, Descartes) for their support, in particular, Sanketh Shetty for the text data and assistance, to Samy Bengio and Yoram Singer for initial discussions, and to Hector Yee and Jason Weston, espeically for their much needed guidance during implementation.