Indian Institute of Technology Kanpur

Department of Computer Science and Engineering

Machine Learning : CS771A

Image Tagging

Team Members:

Jayant Agrawal	14282
Munot Rushab	14405
Rishabh Goyal	14549
Sagar Chand	14579
Srishti Jain	14708

Supervisor:

Professor Piyush Rai

1 Problem Statement

We are working on the problem of automatic image annotation. Given an image, our task is to place it in a set of categories. The problem can be viewed as a multi-class multi-label classification problem, with the set of tags being the classes.

2 Prior Work

We have studied two papers that tackle this problem:

FastTag [1]

The algorithm learns two mappings to predict tag annotations. Considering $\boldsymbol{y_i}$ to be the one-hot vector of tags and $\boldsymbol{x_i}$ to be the image features for the i^{th} image, the two mappings are as follows:

1. A mapping $\boldsymbol{B}$ from the tag vector space to itself, $\boldsymbol{y_i} \rightarrow \boldsymbol{By_i}$ enriches the existing sparse tag vector $\boldsymbol{y_i}$ by estimating which tags are likely to co-occur with those already in $\boldsymbol{y_i}$. The mapping attempts to reconstruct the (unknown) complete tag set from the sparse tag set available during training.
2. A mapping $\boldsymbol{W}$ from the image feature space to the tag vector space, $\boldsymbol{x_i} \rightarrow \boldsymbol{Wx_i}$ predicts the complete tag set from image features. It is essentially a mapping from image features to this reconstructed tag set.

The cost function tries to reconcile the tag vector obtained by the transformation of the image features ($\boldsymbol{Wx_i}$) and the complete tag vector ($\boldsymbol{By_i}$) obtained from the incomplete tag vector $\boldsymbol{y_i}$ (Figure 1).

$$\frac{1}{n}\sum_{i=1}^{n} ||\boldsymbol{By_i} - \boldsymbol{Wx_i}||^2$$

The mapping $\boldsymbol{B}$ deals with the problem of tag enrichment. The original tag set is assumed to be incomplete. To obtain a complete set of tags, the original tag set is corrupted and the mapping $\boldsymbol{B}$ is learned from this corrupted tag vector to the original uncorrupted tag vector. It is found that applying $\boldsymbol{B}$ to the original tag vector nearly recovers the complete tag set. The final cost function is as follows:

$$l(\boldsymbol{b}, \boldsymbol{W}, \boldsymbol{x}, \boldsymbol{y}) = \frac{1}{n}\sum_{i=1}^{n} ||\boldsymbol{By_i} - \boldsymbol{Wx_i}||^2 + \lambda||\boldsymbol{W}||_2^2 + \gamma r(\boldsymbol{B})$$

The term $r(B)$ couples the heuristic argument of tag enrichment with the cost function. The cost function reduces to standard ridge regression on fixing $\boldsymbol{W}$ or $\boldsymbol{B}$.

Closed form expressions obtained in this case are:

$$\boldsymbol{W} = \boldsymbol{BYX^T}(\boldsymbol{XX^T}+n\lambda I)^{-1}$$
$$\boldsymbol{B} = (\gamma \boldsymbol{P} + \boldsymbol{WXY^T})(\gamma \boldsymbol{Q} + \boldsymbol{YY^T})^{\boldsymbol{-1}}$$

It achieves comparable results with the state of the art with the added advantage of being computationally cheap. The data can be trained in O(n) time and applied during testing in constant time.

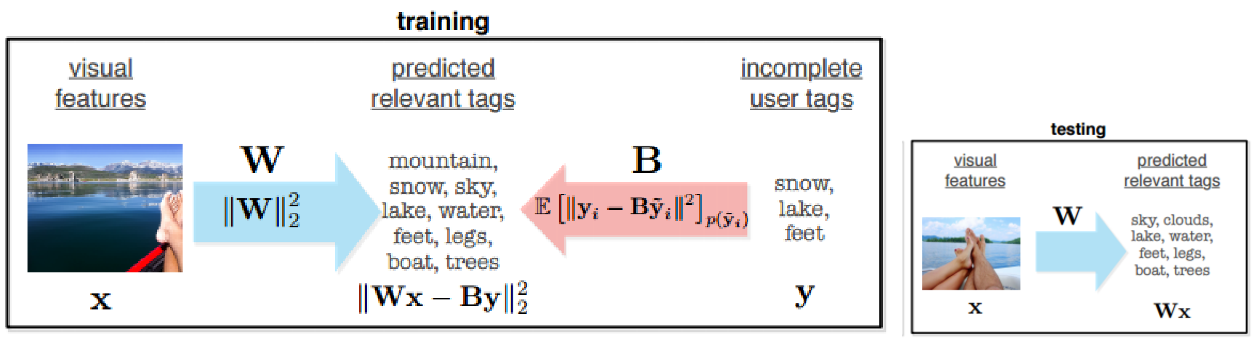

training
visual features
predicted relevant tags
incomplete user tags
W
$\|W\|_2^2$
mountain, snow, sky, lake, water, feet, legs, boat, trees
B
$\mathbb{E}\left[\|y_i - B\tilde{y}_i\|^2\right]_{p(\tilde{y}_i)}$
snow, lake, feet
x
$\|Wx - By\|_2^2$
y
testing
visual features
predicted relevant tags
W
sky, clouds, lake, water, feet, legs, boat, trees
x
Wx

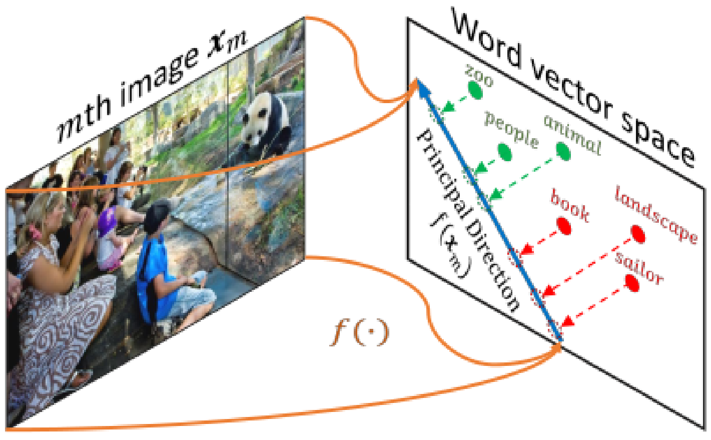
Figure 1: FastTag Explained Graphically [1]

Fast-Zero Tag [2]

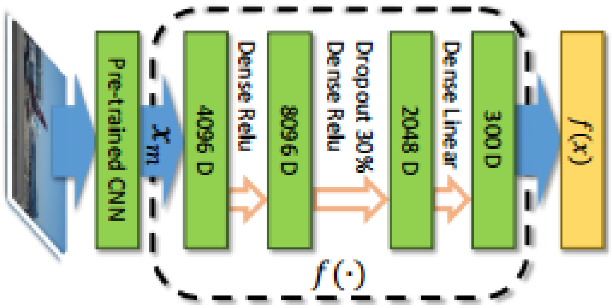
This approach learns a relationship between images and word vectors of tags. It is observed that corresponding to each image, there exists a direction in the word vector space along which the relevant tags rank ahead of the irrelevant ones (2a). The word vectors are known apriori i.e. they are not learnt along with the model. Therefore for a given image, the task is to map it to its principal direction.

Consider an image m represented as a vector, say x. Let T be the tagset. Let $Y_m \subseteq T$ be the set of relevant tags and $\bar{Y_m} = T \setminus Y_m$ be the set of irrelevant tags. Let, for the sake of brevity, Y_m also denote the matrix of word vectors of the tags.

If the principal direction for an image m is $\boldsymbol{w}$ then $\boldsymbol{w} \cdot \langle \boldsymbol{p} - \boldsymbol{n} \rangle \geq 0$ where $\boldsymbol{p} \in Y_m$ and $\boldsymbol{q} \in \bar{Y_m}$



(a) Gist of Fast0Tag



(b) Deep Net Architecture

RankSVM

In this approach the principal direction is learnt by using a Ranking SVM that separates all possible pairs of $\boldsymbol{p} - \boldsymbol{n}$ from all possible pairs of $\boldsymbol{n} - \boldsymbol{p}$.
Further it is assumed that there is a linear relationship between the principal direction $\boldsymbol{w}$ and the image x. Formally, $\exists$ a matrix A such that

$$\boldsymbol{w} = Ax$$

The matrix A can be learned by Linear Regression.

Deep Learning Methods

The task of learning a principal direction in the word vector space and a mapping from the image features to the principal direction are combined and addressed using deep learning techniques. The network uses VGG[4] features to learn the principal direction $f(x)$. The network used can be seen in figure 2b.

3 Suggested Improvements

3.1 FASTTAG [1]

3.1.1 Using Neural Network for learning classifiers

We propose a Multi-Layer Neural Network(Figure 3) with Multi-Task Loss to learn non-linear mappings **B** and **W**(refer Section 2), instead of learning linear mappings in fasttag [1]. To learn **B**, the Tag Set, y_i is first corrupted(Section 2) to get the corrupted Tag vector $\tilde{y}_i$, which is then fed into the two-layer network. The network contains a Dense ReLu Layer(fully connected layer followed by a ReLu layer) followed by a fully connected layer. The standard Sigmoid Cross Entropy Loss is used to learn **B**.

To learn **W**, the standard VGG [4] features(x_i : 4096 D) are extracted and fed into the three layer Network which consists of two Dense ReLu Layers followed by a fully connected Layer to get the output Wx_i. The enriched Tag Set By_i is obtained by passing y_i through a two-layer network whose weights are shared with the network used to learn **B**. Euclidean Loss is used to jointly optimize **B** and **W**, which can be formulated as:

$$\frac{1}{N}\sum_{n=1}^{N}||By_i - Wx_i||^2$$

The complete objective for this network can be formulated as:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}||By_i - Wx_i||^2 + \mu \times l(y_i, \tilde{y}_i) + \lambda R(W, B)$$

where $l(y_i, \tilde{y}_i)$ is the Sigmoid Cross Entropy Loss and μ is the corresponding loss-weight. The shared layers make sure that the enrichment mapping **B** learned through co-occurrence with existing labels also agree with the content of the image. There are thus, two sources of information which help learn **B** more effectively, using the Multi-Task joint loss. Also, the network is regularized by using Batch Normalization and Dropout(30%) for the fully connected layers. Currently, the network is not giving good results and needs more refinement in it's design.

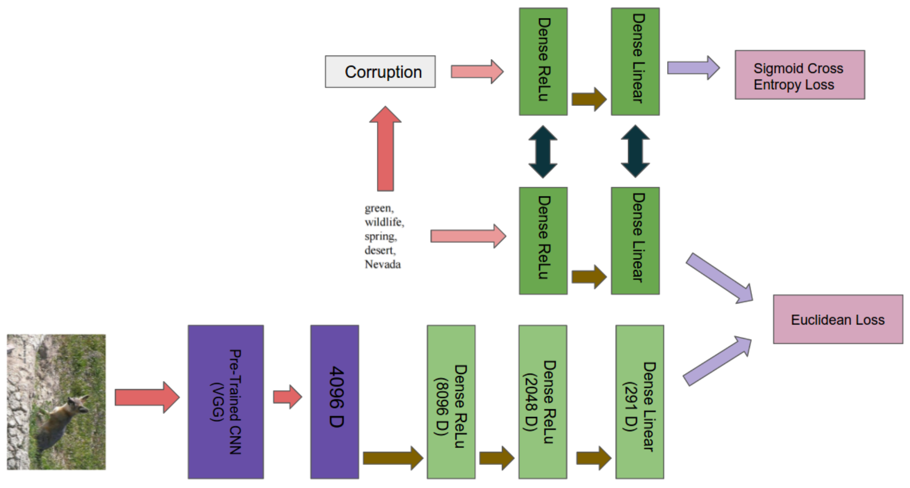


Figure 3: TagNet

3.2 FAST0TAG [2]

3.2.1 Matrix Factorization to learn the embeddings

$\boldsymbol{X}$ is a $T \times T$ dimensional Co-occurrence matrix where $\boldsymbol{X_{ij}}$ is the number of times tag i and j co-occur.
$\boldsymbol{U}$ is a $T \times K$ dimensional matrix for the embeddings of the tag set.
We want to learn embeddings ($\boldsymbol{U}$) for each tag that we would use instead of the its word vector. As we want to learn $\boldsymbol{U}$ for the tag set such that $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{U^T}$, we formulate it as a matrix factorisation problem. Instead, we learn $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V^T}$ and impose an additional constraint on the cost function which forces similarity between $\boldsymbol{U}$ and $\boldsymbol{V}$.
This is done by minimizing the following cost function:

$$L = \sum_{n=1}^{N}\sum_{m=1}^{N}(X_{nm} - u_n^T v_m)^2 + \lambda_u \sum_{n=1}^{N} ||u_n||^2 + \lambda_v \sum_{m=1}^{N} ||v_m||^2 + \lambda_{uv} \sum_{n=1}^{N} ||u_n - v_m||^2 \quad (1)$$

$$\frac{\partial L}{\partial u_n} = -2\sum_{m=1}^{N}(X_{nm} - u_n^T v_m)v_m + 2\lambda_u u_n + 2\lambda_{uv}(u_n - v_m) \quad (2)$$

$$u_n = \left((\sum_{m=1}^{N} v_m v_m^T) + (\lambda_{uv} + \lambda_u)I_K\right)^{-1}(\lambda_{uv} v_m + \sum_{m=1}^{N} X_{nm} v_m) \quad (3)$$

$$\text{Similarly, } v_m = \left((\sum_{n=1}^{N} u_n u_n^T) + (\lambda_{uv} + \lambda_v)I_K\right)^{-1}(\lambda_{uv} u_n + \sum_{n=1}^{N} X_{nm} u_n) \quad (4)$$

3.2.2 Kernelized Ridge Regression to learn mapping between Image Vector and Principal direction

To learn the mapping between the image vector and the principal direction in Rank SVM, we have tried RBF kernels and Polynomial kernels.

4 Experimental Results

4.1 Code sources

- We procured the codes for Deep Learning for Fast-Zero Tag from http://crcv.ucf.edu/projects/fastzeroshot/ and Fast Tag from www.cse.wustl.edu/~mchen/code/FastTag/fasttag.tar.gz.
- We implemented Rank-SVM for Fast-Tag ourselves.
- We also modified the above code for learning the kernelized mapping from image vector to principal direction in ranking SVM's.
- We implemented the neural network(Figure 3) for fasttag in caffe[5].
- The code base can be found at https://github.com/Rushab1/Image_Tagging/
.

4.2 DATASET: IAPRTC-12

The dataset consists of 19,627 images of sports, actions, people, animals, cities, landscapes and many other aspects of contemporary life. Tags are extracted from the free-flowing text captions accompanying each image. Over-all, 291 tags are used. We have used it for all our models.

Method	FastTag	Fast0Tag(RankSvm)	Fast0Tag(Neural Net)
F1 Score	0.32	0.30	0.45

Table 1: Experimental Results

Value of γ	0.0	0.04	0.08	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$k = 3$	0.216	0.227	0.213	0.215	0.220	0.225	0.216	0.206	0.220	0.214	0.223	0.218
$k = 5$	0.299	0.285	0.289	0.290	0.291	0.300	0.283	0.298	0.290	0.295	0.286	0.300

Table 2: F1 Scores for various values of γ for RBF Kernel

Method	**Words embeddings**	**Parameters**	**k**	**Training Data**	**Test Data**
Linear	Word2Vec	-	3	0.46	0.19
Polynomial	Word2Vec	degree 1000	3	0.14	0.14
Polynomial	Word2Vec	degree 50000	3	0.67	0.20
Polynomial	Word2Vec	degree 100000	3	0.67	0.22
RBF (much faster)	Word2Vec	gamma=0.04 (max F1)	3	0.8	0.23
Linear	Co-occurence based	-	3	Bad	Bad
Polynomial	Co-occurence based	degree 50000	3	0.37	0.23
Linear	Word2Vec	-	5	0.54	0.22
Polynomial	Word2Vec	degree 1000	5	0.20	0.18
Polynomial	Word2Vec	degree 50000	5	0.81	0.30
Polynomial	Word2Vec	degree 100000	5	0.80	0.29
RBF (much faster)	Word2Vec	gamma=0.4 (max F1)	5	0.81	0.30
Linear	Co-occurence based	-	5	Bad	Bad
Polynomial	Co-occurence based	degree 50000	5	0.61	0.29

Table 3: F1 Scores for various methods and embedding types for Fast0Tag Rank SVM; Polynomial and RBF denote Kernelized Ridge Regression

5 Inferences and Comparative Study

- In FastZEROTag with Ranking SVMs, we use a two layer transformation, the first one to learn the principal direction (using Ranking SVMs) from the observed tags, and the second to learn a mapping from inputs to the principal direction. We vary the second transformation and test with Linear Regression and Kernelized (Polynomial and RBF) Ridge Regression.
- Using a linear transformation (i.e. Linear regression) between the inputs and the principal direction does not seem to give a good result. Introducing non-linearity increases this accuracy by quite a good amount. Extremely high degree polynomial kernels/RBF kernels increase the accuracy by almost 4% for 3 tag prediction (k=3) and almost 7% for 5 tag predicted (k=5) over linear mappings. Low degree polynomial kernels give poor results. Note that high dimensional polynomial kernels take (much) more time to train than RBF kernels.
- A surprising result that comes into picture here is that Co-occurrence based tag embeddings also perform extremely well on the test set and are almost at par with the Word2Vec embeddings. However the accuracy on the training set is poor (but this doesn't matter much). This is with non-linear mappings. Linear transformations (regression) works poorly on the co-occurrence based embeddings.
- The two stage process in Fast0Tag can be combined into a single step. For this we use a neural net, as described above. The accuracy with neural networks is almost double that of simple Linear Regression, and about 1.5 times that of any of the kernelized methods.

- Fast0Tag learns the principal direction along which the relevant tags rank ahead of the irrelevant ones. This applies to unseen word vectors of tags too. So, this model has also been extended to zero-shot fast tagging. The power of this method has been validated by its extremely high F1 score.
- We implemented FastTag using Neural Net because we wanted to learn non linear mappings without compromising on the test prediction time complexity.
- A great advantage of matrix factorisation is that one can learn embedding for each class (each of which has a word vector associated with it) from the co-occurrence matrix of the tags in the training data without even using WordtoVec. It is useful when the word vectors are not available or when the classes are abstract (i.e. cannot be represented in words).

Acknowledgments

We thank our instructor, **Professor Piyush Rai** for motivating us and guiding us throughout the project.

References

[1] Minmin Chen (Amazon.com, Seattle, WA 98109), Alice Zheng (Microsoft Research, Redmond, WA 98052), Kilian Q. Weinberger (Washington University in St. Louis, St. Louis, MO 63130)*Fast Image Tagging* 2013

[2] Yang Zhang, Boqing Gong, and Mubarak Shah, Center for Research in Computer Vision, University of Central Florida, Orlando, FL 32816 *Fast Zero-Shot Image Tagging* 2016

[3] Guillaumin, M., Mensink, T., Verbeek, J., and Schmid, *C. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In Computer Vision, 2009 IEEE 12th International Conference on, pp. 309–316. Ieee* 2009

[4] Simonyan, K. and Zisserman, A *Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR ,abs/1409.1556* 2014

[5] Jia, Yangqing and Shelhamer, Evan and Donahue, Jeff and Karayev, Sergey and Long, Jonathan and Girshick, Ross and Guadarrama, Sergio and Darrell, Trevor, arXiv preprint arXiv:1408.5093, *Caffe: Convolutional Architecture for Fast Feature Embedding* , 2014