SEMINAR "EXPLAINABLE MACHINE LEARNING" 2018

Metric Learning with Adaptive Density Discrimination

PHILIPP REISER

June 30, 2018

Contents

1 Introduction

In the following we will give an overview to the paper "Metric Learning With Adaptive Density Discrimination" [Rip+15] by Oren Rippel (MIT, Facebook AI Research), Manohar Paluri (Facebook AI Research), Piotr Dollar (Facebook AI Research) and Lubomir Bourdev (UC Berkeley). In their paper they introduced a new method to improve the important field of metric learning.

1.1 Similarity Learning

In the days of big data came up the problem of managing all the data. In the area of classification of images and object detection there were made big steps in the last few year. But one big open field of research is called image retrieval that should automatically sort a database by their content or a wanted property. So one should be able to feed in a picture to the algorithm and it should return similar pictures. This field of research is also called Metric Learning or Similarity Learning which tries to find a good metric that is able to connect distance to semantic similarity. It has also many great benefits as it could enable zero-shot learning or visualization of high dimensional data. This leads us also to the field of interpretability because as a human you can assess the quality of an classifier if it is able to present you similar pictures to a given one. In this report we are going to discuss a method of distance metric learning (DML) which should be able to learn by itself important intra- and inter-class similarities.
One simple approach in similarity learning might be to define a metric on RGB-channels to distinguish classes of images. But this approach has limited applicability for real world problems if one uses only standard metrics as L_1 or L_2. Now one could choose manually difficult metrics that could be appropriate for a problem but also this gets soon impracticable.
Instead of using handcrafted metrics we can learn a appropriate metric for our problem. This leads us to the field of metric learning which we will introduce in the following. In general one can give a basic recipe after which one can generate a metric learning algorithm [BC15].
As a first step one has to choose a parametric distance or similarity function called $D_M(x, x')$ which returns a measurement of similarity. This function can be simple with few parameters but it could also be a complex function as a neural network. Then one needs to build a dataset on which the learning algorithm can be trained. The dataset should contain data pairs/triplets that contain information about similarity. There is the set $S = \{(x_i, x_j) : x_i$ and x_j are similar $\}$, the set $D = \{(x_i, x_j) : x_i$ and x_j are dissimilar $\}$ and then a set of triplets $R = \{(x_i, x_j, x_R) : x_i$ is more similar to x_j than to x_R $\}$.
The next step is to define an objective function L to be minimized with respect to the parameters M of the similarity function:

$$\hat{M} = argmin_M[L(M, S, D, R) + \lambda reg(M)] \tag{1}$$

with a regularization function reg. By minimizing this similarity function we learn hopefully a metric which is able to find meaningful similarities in images.

2 Motivation

In the following we will discuss several known problems of metric learning. One problem is the so called "predefined target neighborhood structure" that is a phenomena that occurs when one defines a relationship between similarity and distance. In practice one tries to define similarity in-prior by giving similar objects of the dataset the same label. This leads to the fact that one uses the input space as a notion of similarity. But this is contradictory to the wanted property of a similarity notion that is independent of the original feature space. So the authors suggest to use a similarity function that depends on the distance in representation space. Due to the fact that the representation is changed during training we can define an adaptive similarity.
Another common problem occurs when one tries to find out the right objective function or the so called loss. The widely spread Triplet Loss [WS09] that is used for several metric learning task has some problems that we will discuss by introducing the triplet loss.
To gain a representation space in which similarity correspondence to distance with the triplet loss you need to define triplets of the seed example r_m, the positive example r_m^+ and the negative example r_m^-. The positive example should be similar and the negative example dissimilar to the seed example. The notion r_m tells us that we are using the representations of the samples in the representation space. To train now a representation space one has to minimize the following objective:

$$L_{triplet}(\theta) = \frac{1}{M} \sum_{m=1}^{M} \{\|r_m - r_m^-\|_2^2 - \|r_m - r_m^+\|_2^2 + \alpha\}_+ \tag{2}$$

with $\{\}_+$ the hinge function and θ the parameters of the function that maps the original input data to the representation space. As one can see the objective penalizing only single triplets at once and so you get no insights to the distribution of the data. Another bad side effect is the cubic growth of computational effort with the number of triplets.

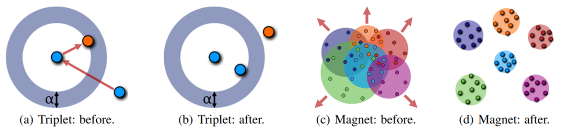


Figure 1: Graphical explanation of the intuition behind triplet loss and magnet loss

3 Magnet Loss

With the mentioned problems in mind we now go further and discuss a possible solution. The ideal DML approach should contain a similarity notion that depends adaptively on the current representation instead of using a predefined notion of similarity. Then the algorithm should be able to separate class distributions even if they overlap locally. This could be done by a loss that is informed about the distribution. Now we can introduce the magnet loss that is based on the upper ideas. This loss uses a clustering technique to get informed about the distribution that can so be manipulated. The basic intuition is showed in 1. It forces cluster attraction and repulsion instead of manipulating only single triplets.

3.1 Model

In the following we will denote our training set of images/data with the corresponding labels as: $D = \{x_n, y_n\}_{n=1}^{N}$ where y belongs to one of C classes. The map between the images to the representation space $f(\theta)$ is in our case a CNN, precisely a GoogLeNet [Sze+14], whereas θ describes the learnable parameters of the CNN. By the notion $I_1^C, ..., I_k^C$ we refer to k clusters for each class C which are returned by the clustering algorithm (in our case K-means):

$$I_1^C, ..., I_k^C = argmin_{I_1^C, ..., I_K^C} \sum_{k=1}^{K} \sum_{r \in I_k^c} \|r - \mu_k^c\|_2^2 \tag{3}$$

with μ_k^c the center of the class c and cluster k in the representation space. Now we can define our objective called Magnet Loss:

$$L(\theta) = \frac{1}{N} \sum_{n=1}^{N} \{-log \frac{e^{\frac{1}{2\sigma^2}\|r_n - \mu(r_n)\|_2^2 - \alpha}}{\sum_{c \neq C(r_n)} \sum_{k=1}^{K} e^{-\frac{1}{2\sigma^2}\|r_n - \mu_k^c\|_2^2}}\} \tag{4}$$

with $\{\}_+$ the hinge function, $\alpha \in \mathbb{R}$ a hyperparamteter and $\sigma^2 = \frac{1}{N-1}\sum_{r \in D} \|r - \mu(r)\|_2^2$ the variance of the samples to their cluster. By normalizing with the variance we get our loss function invariant to the characteristic length scale of the problem and this has the consequence that the desired gap between the clusters α is in units of the variance. The objective function penalizes samples that have a bigger distance than α from their own cluster center and penalizes samples that are close to cluster means from other classes.

3.2 Training

The first step of training is called "neighborhood sampling". In each iteration the mini batch is constructed under the following way (to sample the entire neighborhood): First one chooses a cluster $I_1 \propto p_I()$ where p_I is chosen to adapt the neighborhood. In our case its proportional to the loss of cluster I. Next we get the nearest clusters that do not belong to the same class, called nearest impostor clusters $I_2, ..., I_M$. Then we sample D examples $x_1^m, ..., x_D^1$ for each cluster.

With these samples we compute an approximation of the magnet loss:

$$L(\theta) = \frac{1}{MD} \sum_{m=1}^{M} \sum_{d=1}^{D} \{-log \frac{e^{\frac{1}{2\hat{\sigma}^2}\|r_d^m - \hat{\mu}_m\|_2^2 - \alpha}}{\sum_{C(\hat{\mu}) \neq C(r_d^m)} e^{-\frac{1}{2\hat{\sigma}^2}\|r_d^m - \hat{\mu}\|_2^2}}\} \tag{5}$$

where $\hat{\mu}_m = \frac{1}{D}\sum_{d=1}^{D} r_d^m$ and $\hat{\sigma}_m = \frac{1}{MD-1}\sum_{m=1}^{M}\sum_{d=1}^{D} \|r_d^m - \hat{\mu}_m\|_2^2$ are the approximated mean and deviation of the clusters. By backpropagating through

the whole CNN we can change the the representation space.
The second important training component is the indexing of the clusters. The representations are indexed by pausing training and compute a forward pass of the training inputs and by applying K-means++ to the representations we get the indices. These are refreshed frequently.
The first component leads to the effect that we change our representation according to the distribution which is approximated by the clusters. Another nice side effect is the reduction of computational effort for the training procedure by manipulating entire clusters instead of only triplets. Moreover by penalizing whole clusters of points that are far away from each other we get a more consistent representation than by only penalizing triplets.

3.3 Evaluation

To evaluate a trained model the we need to assign a label to each example x_n. To manage that we use a technique called "k-nearest cluster" which computes the L closest clusters, approximated by $\mu_1, ..., \mu_L$ and then chooses the label c_n^* by the following formula:

$$c_n^* = argmax_{c=1,...,C} \frac{\sum_{\mu_l : C(\mu_l)=c} e^{-\frac{1}{2\sigma^2}\|r_n-\mu_l\|_2^2}}{\sum_{l=1}^{L} e^{-\frac{1}{2\sigma^2}\|r_n-\mu_l\|_2^2}} \tag{6}$$

This is computationally less expensive than the kNN classifier because it scales only with the number of clusters. In the experiments $L = 128$ was used as number of clusters.

4 Experiments

The experiments were made with a GoogLeNet as mapping $f(.;\theta)$ to the representation space. The net was slightly pretrained on ImageNet for 3 epochs.

4.1 Fine-Grained Classification

As a first experiment they tried to solve with their new approach the task of visual categorization on datasets where the classes differ only slightly from each other as Stanford Dogs, Oxford-IIIT Pet and Oxford 102 Flowers.
As comparison to Magnet Loss there were also trained a softmax classifier and triplet loss. To create an equitable comparison the hyperparameter search was done for all three models. To evaluate triplet loss kNN was used and for Magnet Loss the above mentioned kNC was used. The results can be found in 2.
As one can see Magnet Loss is slightly better than softmax and a lot better than the standard similarity learning approach Triplet Loss.

4.2 Attribute Distriburtion

One major goal to achieve by Magnet Loss was to get an expressive representation, in which similar objects of different classes are close together and dissimilar objects of the same class are far away from each other. To check this in

practice the model was trained with the attributes of the neighborhood structure and the learned representations were checked and compared to those of Triplet Loss, as can be seen in 6.
To get a quantitative measurement the Object Attributes dataset was used in are which 25 attribute annotations of 90 classes. These attributes were kept back during training. During testing the intra- and inter- class variance should be discovered by Magnet Loss and ideally correspond to the attributes. This was done by measuring the mean attribute precision as a function of the neighborhood size. This score can be found in 2. It can be seen that Triplet and Softmax were outperformed by Magnet.
A qualitative comparison of the representation space was done for Magnet Loss and Triplet Loss. In 6 are shown two projections of the representation space, created by t-SNE [MH08]. There one can see clearly that Magnet Loss learns a structured representation that is meaningful. The intra-class clusters correspond to Attributes and one can see that close inter-class clusters share same attributes from different classes. However, Triplet Loss creates an representation that is uniform and only separates the classes.

4.3 Hierarchy Recovery

In the last experiment the goal was to compare the latent class hierarchy by giving only a broad supclass. The models were trained with random pairs of classes of ImageNet Attributes and each pair was given an random Attribute which leaded to corrupted labels. After that it was tried to recover the original higher labels.
The results show that Magnet outperformes the two other approaches and is able to recover intra-class variations.

Approach	Error
Angelova & Long	51.7%
Gavves et al.	49.9%
Xie et al.	43.0%
Gavves et al.	43.0%
Qian et al.	30.9%
Softmax	26.6%
Triplet	35.8%
Magnet	**24.9%**

(a) Stanford Dogs.

Approach	Error
Angelova & Zhu	23.3%
Angelova & Long	19.6%
Murray & Perronnin	15.4%
Sharif Razavian et al.	13.2%
Qian et al.	11.6%
Softmax	11.2%
Triplet	17.0%
Magnet	**8.6%**

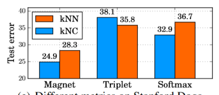
(b) Oxford 102 Flowers.

Approach	Error
Angelova & Zhu	49.2%
Parkhi et al.	46.0%
Angelova & Long	44.6%
Murray & Perronnin	43.2%
Qian et al.	19.6%
Softmax	11.3%
Triplet	13.5%
Magnet	**10.6%**

(c) Oxford-IIIT Pet.

Approach	Error
Softmax	**14.1%**
Triplet	26.8%
Magnet	15.9%

(d) ImageNet Attributes.



(e) Different metrics on Stanford Dogs.

Approach	Error@1	Error@5
Softmax	30.9%	15.0%
Triplet	44.6%	23.4%
Magnet	**28.6%**	**7.8%**

(f) Hierarchy recovery on ImageNet Attributes.

Figure 2: (a)-(d) showing the results on classification tasks on fine-grained categorization datasets (e) comparison of different evaluation metrics (f) testing the ability of reconstructing fine-grained labels by training with corrupted pairs of coarse superclasses

5 Discussion

This paper tried to figure out the modern problems of metric learning and presented a novel approach with a theoretical explanation and the authors were able to present an improvement to state-of-the-art techniques. To further optimize the method the used clustering technique (K-means++) that is applied on the representations, could be interchanged by a more natural candidate as a tree-based algorithm.

6 Appendix

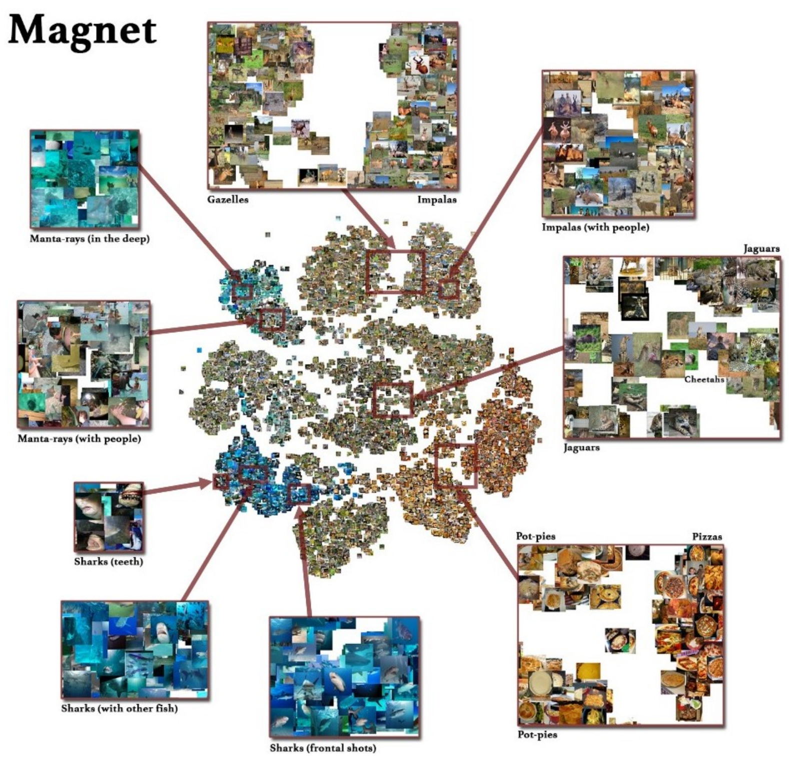


Figure 3: t-SNE projection of representation space learned by Magnet Loss

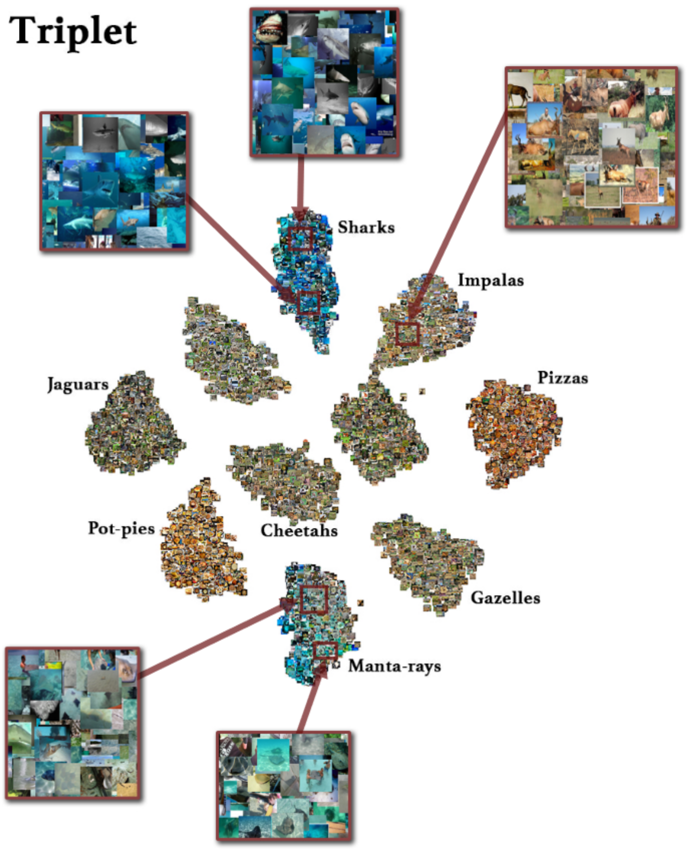


Figure 4: t-SNE projection of representation space learned by Triplet Loss

References

[Fuh05] Norbert Fuhr. *Colour similarity search in image*. 2005. URL: http://www.is.informatik.uni-duisburg.de/courses/ir_ss05/folien/colorsearch-ho.pdf.

[MH08] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605. URL: http://www.jmlr.org/papers/v9/vandermaaten08a.html.

[WS09] Kilian Q. Weinberger and Lawrence K." Saul. "Distance metric learning for large margin nearest neighbor classification". In: (2009).

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[Sze+14] C. Szegedy et al. "Going Deeper with Convolutions". In: *ArXiv e-prints* (Sept. 2014). arXiv: 1409.4842 [cs.CV].

[BC15] Aurélien Bellet and Matthieu Cord. *Similarity and Distance Metric Learning with Applications to Computer Vision*. 2015. URL: http://researchers.lille.inria.fr/abellet/talks/metric_learning_tutorial_ECML_PKDD.pdf.

[Rip+15] O. Rippel et al. "Metric Learning with Adaptive Density Discrimination". In: *ArXiv e-prints* (Nov. 2015). arXiv: 1511.05939 [stat.ML].

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.