Published in proceedings of the 30th Conference on Neural Information Processing Systems (NIPS 2016), 05-10 December 2016, Barcelona, Spain, which should be cited to refer to this work.

Learning with Feature Side-information

Amina Mollaysa
HES-SO & University of Geneva, Switzerland
maolaaisha.aminanmu@hesge.ch

Pablo Strasser
HES-SO & University of Geneva, Switzerland
pablo.strasser@hesge.ch

Alexandros Kalousis
HES-SO & University of Geneva, Switzerland
Alexandros.Kalousis@hesge.ch

Abstract

Very often features come with their own vectorial descriptions which provide detailed information about their properties. We refer to these vectorial descriptions as feature side-information. The feature side-information is most often ignored or used for feature selection prior to model fitting. In this paper, we propose a framework that allows for the incorporation of feature side-information during the learning of very general model families. We control the structures of the learned models so that they reflect features' similarities as these are defined on the basis of the side-information. We perform experiments on a number of benchmark datasets which show significant predictive performance gains, over a number of baselines, as a result of the exploitation of the side-information.

1 Introduction

In this paper we consider settings in which, in addition to the classical data matrix $\mathbf{X} : n \times d$ containing n instances and d features, and the target matrix $\mathbf{Y} : n \times m$, we are also given a matrix $\mathbf{Z} : d \times c$, the ith row of which contains a description of the ith learning feature. We call $\mathbf{Z}$ the feature side-information matrix.

Typically feature side-information is used for feature selection as a pre-processing task, prior to any modelling or learning [4]. Work that tries to directly exploit feature-side information in learning is mainly limited to linear models or on tasks such as matrix completion, robust PCA and collaborative filtering[5, 9, 2, 3]. The authors of [5] explored the use of side-information in the non-linear setting, however, their work is constrained to a limited class of non-linear models, polynomials of a given degree, and is based on hand engineering of the features and the respective feature side-information. In this work, we use the feature side-information directly within the learning of arbitrary nonlinear models in supervised settings such as classification and regression. Intuitively, we want the learned models to treat features with similar side-information in a similar manner. In linear models, we can directly impose such constraints on the model parameters during learning since we have direct access on the parameters associated with every feature. For example, we can force the weights of similar features to be similar. It is not obvious how to do so in nonlinear models since the parameters are now shared between different features and we cannot dissentagle them. In this paper, we show how we can enforce feature similarity constraints in the case of arbitrary non-linear models under some assumptions, namely that the learned nonlinear model is differentiable with respect to input features. Briefly we constrain the derivatives of the models with respect to input features to follow the feature manifold as this is given by the feature-side information. We present two approaches to impose such constraints. In the first, we tackle the problem by explicitly computing the derivatives of the learned representation with respect to the inputs and add to the loss function a classical Laplacian regularizer applied on these derivatives. The Laplacian regularizer is defined based on the feature

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

side-information matrix. The second approach implements indirectly the same idea through data augmentation on the input instances guided by the feature side-information. We show that the two approaches are equivalent. We experiment on a number of text classification datasets, nevertheless the methods have wide applicability to very diverse application domains.

2 Incorporating the Feature Side-information

In order to incorporate the feature side-information within modeling phase, we define a feature similarity matrix based on feature side-information. Specifically, similarity matrix $\mathbf{S} \in \mathbb{R}^{d \times d}$ with element $S_{ij} = exp(-\frac{1}{2\sigma^2}(\mathbf{z}_i - \mathbf{z}_j)^T(\mathbf{z}_i - \mathbf{z}_j))$ gives a measure on the similarity of feature i, j; $\mathbf{z}_i, \mathbf{z}_j \in \mathbb{R}^c$ are the feature side-information vectors of i and j features respectively. We want to learn a mapping $\phi : \mathbf{x} \in \mathbb{R}^d \rightarrow \mathbf{y} \in \mathbb{R}^m$ using the information provided by the $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ matrices. We will use L to denote some loss function through out the paper.

2.1 Learning Symmetric Models with Respect to Feature Similarity

As already mentioned, we want to constrain the learned model so that it treats similar features in a similar manner. In order to model such a constraint we will require that given a pair of similar features (on the limit identical), a relative change in the values of these features (up to certain amount) should have only a small effect on the output of the model. This means that the function we learn does not depend on the individual contributions/values of the two similar features but only on their total contribution. As a result the learned model will have a symmetric structure, with respect to how it treats the different features, which will reflect the feature similarity. In other words exchanging the values of similar features will have only a detrimental effect on the model output, inversely proportional to their similarity. Figure 1 visualize the idea in the limit case.

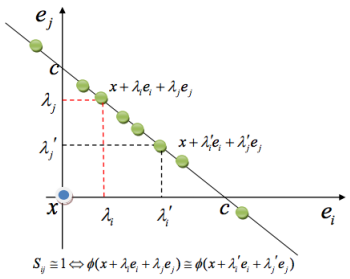

Figure 1: x, blue dot, is a given instances, the two axes are the ith and jth features. If $S_{ij} = 1$, then the model's output is constant along the line defined as: $(x + \lambda_i \mathbf{e}_i + \lambda_j \mathbf{e}_j), \forall \lambda_i + \lambda_j = c$.

Concretely, we constrain the learned model to follow feature similarity by imposing the following constraint:

$$||\phi(\mathbf{x} + \lambda_i \mathbf{e}_i + \lambda_j \mathbf{e}_j) - \phi(\mathbf{x} + \lambda_i' \mathbf{e}_i + \lambda_j' \mathbf{e}_j)||^2 \propto \frac{1}{S_{ij}} \tag{1}$$

where

$$\lambda_i + \lambda_j = \lambda_i' + \lambda_j' = c, \forall \{\lambda_i, \lambda_j, \lambda_i', \lambda_j'\} \in \Omega \tag{2}$$

$\mathbf{e}_i$ is the d-dimensional unit vector, its ith dimension is one and all others are zero; c is a constant and Ω is a small neighborhood of $\lambda_i, \lambda_j, \lambda_i', \lambda_j'$, which defines a local region in which the equation (1) holds. What condition (1) states is that if we add small values to feature i and j of instance $\mathbf{x}$ in different proportions, as long as their total contribution remains fixed, i.e. $\lambda_i + \lambda_j = \lambda_i' + \lambda_j'$, the distance of the two model outputs should be inversely proportional to the corresponding features' similarity S_{ij}. Intuitively, when features i and j are similar, the effect of exchanging one to another is quantified by their similarity measure. When feature i, j are identical, i.e. $S_{ij} = 1$, they are exchangeable. Thus we want the learned representation $\phi(\mathbf{x}_n)$ to reflect the features similarity in the

manner described in equation1. Figure 1 visualizes the main intuition. One way to achieve that is through the following function:

$$\int\int\int\int ||\phi(\mathbf{x}_n + \lambda_i \mathbf{e}_i + \lambda_j \mathbf{e}_j) - \phi(\mathbf{x}_n + \lambda'_i \mathbf{e}_i + \lambda'_j \mathbf{e}_j)||^2 S_{ij} \mathbf{I}(\lambda_i, \lambda_j, \lambda'_i, \lambda'_j) \mathbf{k}(\lambda_i, \lambda_j, \lambda'_i, \lambda'_j) d\lambda_i d\lambda_j d\lambda'_i d\lambda'_j \tag{3}$$

where

$$\begin{aligned} I(\lambda_i, \lambda_j, \lambda'_i, \lambda'_j) &= 1 \text{ if } \lambda_i + \lambda_j = \lambda'_i + \lambda'_j, \text{ and } 0 \text{ otherwise} \\ \mathbf{k}(\lambda_i, \lambda_j, \lambda'_i, \lambda'_j) &= 1 \text{ if } (\lambda_i, \lambda_j, \lambda'_i, \lambda'_j) \in \Omega, \text{ and } 0 \text{ otherwise} \end{aligned} \tag{4}$$

However, the presence of the integral makes the computation quite difficult, instead of directly applying (3), we use the first order Taylor approximation of $\phi(\mathbf{x} + \lambda_i \mathbf{e}_i + \lambda_j \mathbf{e}_j)$, $\phi(\mathbf{x} + \lambda'_i \mathbf{e}_i + \lambda'_j \mathbf{e}_j)$ at $\mathbf{x}$, and the constraint (2), to derive a local approximation of the regularizer (3):

$$\sum_{ij} \left|\left| \frac{\partial \phi(\mathbf{x})}{\partial x_i} - \frac{\partial \phi(\mathbf{x})}{\partial x_j} \right|\right|^2 S_{ij} \tag{5}$$

Where $\frac{\partial \phi(\mathbf{x})}{\partial x_i}$ is the m length vector giving the derivative of the learned function with respect to the ith input feature. Attention, this should not be confounded with the derivatives of the model with respect to its parameters typically used in learning in methods such as gradient descent. We add this simplified constrain (5) to the loss function to forces the learned model to reflect the feature similarity.

$$\min_{\phi} \sum_k L(\mathbf{y}_k, \phi(\mathbf{x}_k)) + \lambda_1 \sum_k \sum_{ij} ||\frac{\partial \phi(\mathbf{x}_k)}{\partial x_i} - \frac{\partial \phi(\mathbf{x}_k)}{\partial x_j}||^2 S_{ij} \tag{6}$$

Thus the resulting optimization problem is composed of a loss function and an additional term which can be seen as a regularizer that forces the derivative of the model with respect to the input features, or in other words the model's sensitivity to the input features, to reflect the features similarity.

2.2 Learning Symmetric Models with Data Augmentation

Optimizing (6) can be computationally expensive due to the presence of derivative of the model output with respect to the inputs in the objective function which later brings in the computation of the Hessian if we are to use gradient-based optimization. As already mentioned the main intuition underlying the optimization problem (6) is to force the model to be symmetric with respect to similar features. The model's output should not change with changes to the values of similar features provided that their total contribution remains constant. Instead of relying on the derivatives of the model with respect to the input features we propose an alternative approach which relies on data augmentation and a simpler regularizer which does not require the use of the model derivatives. Given an instance $\mathbf{x}$, we generate a number of perturbed instances by modifying the values of each pair (i, j) of similar features taking care to keep their total contribution fixed. Concretely, let us denote by M_i the set of indeces of the features that are similar to feature i. Then from $\mathbf{x}$ we generate instance couples as follows:

$$\mathbf{x} \rightarrow \left\{ \begin{array}{l} \mathbf{x} + \lambda_i \mathbf{e}_i + \lambda_j \mathbf{e}_j \\ \mathbf{x} + \lambda'_i \mathbf{e}_i + \lambda'_j \mathbf{e}_j \end{array} \right. i \in \{1...d\}, j \in M_i, \lambda_i + \lambda_j = \lambda'_i + \lambda'_j = c \tag{7}$$

and require that the model outputs on such instance pairs follows the feature similarity using a simple regulariser on the model output. The final optimization problem is given by:

$$\min_{\phi} \sum_k L(\mathbf{y}_k, \phi(\mathbf{x}_k)) + \lambda_1 \sum_k \sum_i \sum_{j \in M_i} ||(\phi(\mathbf{x}_k + \lambda_i \mathbf{e}_i + \lambda_j \mathbf{e}_j) - \phi(\mathbf{x}_k + \lambda'_i \mathbf{e}_i + \lambda'_j \mathbf{e}_j))||^2 S_{ij} \tag{8}$$

where the second term runs over the pairs of augmented instances and requires that the model's outputs follow the feature similarity of the pair of features that was used to generate the instances.

3 Experimental results

We learn ϕ using a standard feed forward neural network. We have experimented on eight document classification datasets also used in [6]. Documents are represented as bag of words. We use as feature side-information the word2vec representation of the words [7]; other possibilities include knowledge-based side-information, e.g. based on WordNet [8]. We removed words that appear less

than three times across all the documents of a given dataset. We compute the $\mathbf{S}$ similarity matrix from the word2vect word representations. For computational reasons, we threshold the entries of $\mathbf{S}$ and keep only those that are larger than 0.8. For the data augmentation approach, ideally, for each instance in the training dataset, we can augment by all pairs of similar features, however, this lead to the exploding of the instance number. Instead, during training, for each instances in the mini batch, we randomly chose a pair of similar features, we perturb the instance by the chosen pair of features a number of times using different proportion of individual contribution. Therefor, with a large number of training iteration, eventually, we believe that enough number of instances are generated to cover most of the relevant information given by the feature side-information. The value c, which is the total contribution of feature i and j added to the original instances is chosen by cross validating over the values $c \in \{0, 1, 2, 4, 6, 8, 10\}$ (those values are chosen by statistics of the training data). We compare the two approaches against popular regularizers used with neural networks, namely ℓ_2 and dropout regularization [10]. We also compare against WMD [6] which makes direct use of the side-information to calculate the distance of the document as they refer to Word Mover's Distance. It should be clear that our methods and WMD have an advantage over ℓ_2 and dropout since they exploit the side-information which ℓ_2 and dropout do not. We separate the datasets by $4:1$ proportion to a training and testing set. We tune the hyperparameters of all methods using five-fold cross validation in the training set. We choose the λ of our method and the λ of ℓ_2 from $\lambda \in \{0.0001, 0.001, 0.01, 0.1, 0, 10\}$; λ of dropout from $\lambda \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$. We give the classification errors in table 1; we denote the neural network trained with our derivative-based regulariser with NN+FD and the one trained with the augmented data and the simpler regulariser with NN+DA. We report results only with a single layer network. The best method for each dataset is marked in bold. As we can see from the table the data augmentation approach gives the best results, being the best method in four out of the eight dataset, closely followed by the derivative based method.

Dataset	n	d	NN+FD	NN+DA	NN+L2	NN+dropout	WMD
BBCsport	737	2829	2.70	2.72	**1.36**	2.72	4.6
Twitter	3108	4076	29.9	31.4	**25.93**	26.25	29.00
Classic	7093	4407	3.88	3.74	4.51	4.23	**2.80**
Amazon	8000	4502	6.25	**5.81**	8.60	7.6	7.40
20NEWS	18774	6859	16.79	**12.46**	26.76	25.72	27.00
Recipe	4370	4992	38.44	**35.70**	41.76	41.08	43.00
Ohsumed	9152	7643	36.28	**36.07**	47.27	42.35	44.00
Reuter	7674	3296	4.10	3.59	4.95	4.1	**3.50**

Table 1: Data statistics and misclassification errort: NN+L2, NN+dropout refers to the feed-forward network with L2 regularizer and dropout regularizer respectively.

3.1 Conclusion and Future Work

We have presented two approaches that use feature side-information directly in the model learning. Both of them implement the same intuition: changing the relative contribution of two similar input features, while keeping their total contribution fixed, should have only a small effect on the model's output. The first approach relies on the derivatives of the model with respect to the input features. These derivatives measure the model's sensitivity to the input features. We apply a Laplacian regulariser on them to force the model's feature sensitivity to reflect the feature manifold as the latter is defined by the feature side-information. We developed a variant of the backpropagation algorithm that allows the computation of the gradient of cost functions that include the derivatives of the model with the input (will be given in longer version of the paper). Since this approach has a high computational cost due to the presence of the derivatives in the optimization function, requiring a calculation of the Hessian with gradient descent, we propose a second variant that makes use of data augmentation to implement the same intuition and has a simpler cost function. We generate instances by using the feature similarity matrix as this is computed from the feature side-information. We performed experiments on a set of document classification datasets which show important performance gains with respect to standard regularisers as well as WMD which uses feature side-information. One basic assumption of the proposed learning models is that the feature side-information is relevant for the given learning tasks and they can be used as is. However this is questionable, a more appropriate approach and the target of future work is learning the feature similarities as a part of the final model.

References

[1] J. Bian, B. Gao, and T.-Y. Liu. Knowledge-powered deep learning for word embedding. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 132–148. Springer, 2014.

[2] K.-Y. Chiang, C.-J. Hsieh, and E. I. S. Dhillon. Robust principal component analysis with side information. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2291–2299, 2016.

[3] K.-Y. Chiang, C.-J. Hsieh, and I. S. Dhillon. Matrix completion with noisy side information. In *Advances in Neural Information Processing Systems*, pages 3447–3455, 2015.

[4] E. Krupka, A. Navot, and N. Tishby. Learning to select features using their properties. *Journal of Machine Learning Research*, 9(Oct):2349–2376, 2008.

[5] E. Krupka and N. Tishby. Incorporating prior knowledge on features into learning. In *International Conference on Artificial Intelligence and Statistics*, pages 227–234, 2007.

[6] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, pages 957–966, 2015.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[8] G. A. Miller. Wordnet: A lexical database for english. *COMMUNICATIONS OF THE ACM*, 38:39–41, 1995.

[9] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in Neural Information Processing Systems*, pages 2098–2106, 2015.

[10] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.