Regularising Non-linear Models Using Feature Side-information

Amina Mollaysa [* 1]   **Pablo Strasser** [* 1]   **Alexandros Kalousis** [1]

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. In the standard learning scenario, input is represented as a vector of features and the feature side-information is most often ignored or used only for feature selection prior to model fitting. We believe that feature side-information which carries information about features intrinsic property will help improve model prediction if used in a proper way during learning process. In this paper, we propose a framework that allows for the incorporation of the feature side-information during the learning of very general model families to improve the prediction performance. 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

Side-information in machine learning is a very general term used in very different learning scenarios with quite different connotations. Nevertheless, generally it is understood as any type of information, other than the learning instances, which can be used to support the learning process; typically such information will live in a different space than the learning instances. Examples include learning with privileged information (Vapnik & Izmailov, 2015) in which during training a teacher provides additional in-

formation for the learning instances; this information is not available in testing. In metric learning and clustering, it has been used to denote the availability of additional similarity information on instances, i.e. pairs of similar and dissimilar instances, (Xing et al., 2002). In this paper we focus on side-information describing the features. We will consider learning problems in which we have additional information describing the properties and/or the relations of the features. The features will have their own vectorial descriptions in some space in which we will describe their properties.

Real world problems with such properties are very common. For example in drug efficiency prediction problems, and more general in chemical formulae property prediction problems, drugs/formulae are collections of molecules. Each molecule comes with its own description, for example in terms of its physio-chemical properties, and/or its molecular structure. In language modeling, words are features and the words' semantic and syntactic properties are their side-information. In image recognition, pixels are features and their position is the side-information, and so on. Similar ideas also appear in tasks such as matrix completion, robust PCA and collaborative filtering (Rao et al., 2015; Chiang et al., 2016; 2015). There one seeks low rank matrix decompositions in which the component matrices are constrained to follow relationships given by side-information matrices, typically matrices which contain user and item descriptors.

Despite the prevalence of such problems, there has been surprisingly limited work on learning with feature side-information. Krupka et al., 2008, used the feature side-information to perform feature selection as a preprocessing step prior to any modelling or learning. More interestingly (Krupka & Tishby, 2007) exploit the feature side-information directly within the learning process, by forcing features that have similar side-information to have a similar weight within a SVM model. One can think of this as a model regularisation technique in which we force the model structure, i.e. the feature parameters, to reflect the feature manifold as this is given by the feature side-information. In the same work the authors also provide an ad-hoc way to apply the same idea for non-linear models, more precisely polynomials of low degree. However the solution that they propose requires an explicit construc-

---
[*]Equal contribution [1]University of Applied Sciences, Western Switzerland; University of Geneva. Correspondence to: Amina Mollaysa <maolaaisha.aminanmu@hesge.ch>, Pablo Strasser <pablo.strasser@hesge.ch>, Alexandros Kalousis <Alexandros.Kalousis@hesge.ch>.

tion of the different non-linear terms, as well as appropriate definitions of the feature side-information that is associated with them. These definitions are hand-crafted and depend on the specific application problem. Beyond this ad-hoc approach it is far from clear how one could regularise general non-linear models so that they follow the feature manifold.

In this paper we present a method for the exploitation of feature side-information in non-linear models. The main idea is that the learned model will treat in a similar manner features that are similar. Intuitively, exchanging the values of two very similar features should only have a marginal effect on the model output. This is straightforward for linear models since we have direct access to how the model treats the features, i.e. the feature weights. In such a case one can design regularisers as Krupka & Tishby, 2007, did which force the feature weights to reflect the feature manifold. An obvious choice would be to apply a Laplacian regulariser to the linear model, where the Laplacian is based on the feature similarity. Such regularisers have been previously used for parameter shrinkage but only in the setting of linear models where one has direct access to the model parameters (Huang et al., 2011). However, in general non-linear models we no longer have access to the feature weights; the model parameters are shared between the features and we cannot disentangle them.

We present a regulariser which forces the learned model to be invariant/symmetric to relative changes in the values of similar features. It directly reflects the intuition that small changes in the values of similar features should have a small effect on the model output. The regulariser relies on a measure of the model output sensitivity to changes in all possible pairs of features. The model sensitivity measure quantifies the norm of the change of the model output under all possible relative changes of the values of two features. We compute this norm by integrating over the relative changes and the data distribution. Integrating over the relative changes is problematic we thus give two ways to approximate the sensitivity measure. In the first approach we rely on a first order Taylor expansion of the learned model under which the sensitivity measure boils down to the squared norm of the difference of the partial derivatives of the model with respect to the input features. Under this approach the regulariser finally boils down to the application of a Laplacian regulariser on the Jacobian of the model. In the second approach we rely on sampling and data augmentation to generate instances with appropriate relative changes over different feature pairs. We approximate the value of the regulariser only on the augmented data.

We implement the above ideas in the context of neural networks, nevertheless it is relatively straightforward to use them in other non-linear models such as SVMs and kernels.

We experiment on a number of text classification datasets in which the side-information is the word2vec representation of the words. We compare against a number of baselines and we show significant performance improvements.

2. Learning Symmetric Models with Respect to Feature Similarity

We consider supervised learning 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, denoted by $\mathbf{z}_i$, contains a description of the ith feature. We call $\mathbf{Z}$ the feature side-information matrix. Note how the $\mathbf{Z}$ matrix is fixed and independent of the training instances. As in the standard supervised setting, instances, $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$, are drawn i.i.d from some non-observed probability distributions $P(\mathcal{X})$ and targets, $\mathbf{y}_i \in \mathcal{Y} \subseteq R^m$, are assigned according to some non-observed conditional distribution $P(\mathcal{Y}|\mathcal{X}), \mathcal{Y} \in \mathbb{R}^m$. In the standard setting we learn a mapping from the input to the output $\phi : \mathbf{x} \in \mathbb{R}^d \to \mathbf{y} \in \mathbb{R}^m$ using the $\mathbf{X}, \mathbf{Y}$ matrices by optimizing some loss function L. In this paper we learn the input-output mapping using in addition to the $\mathbf{X}, \mathbf{Y}$, matrices the feature side-information $\mathbf{Z}$ matrix.

We bring the feature side-information in the learning process through the feature similarity matrix $\mathbf{S} \in \mathbb{R}^{d \times d}$ which we construct from $\mathbf{Z}$ as follows. Given two features i, j, with $\mathbf{z}_i, \mathbf{z}_j$, side-information vectors the S_{ij} element of $\mathbf{S}$ contains their similarity given by some similarity function. We will denote by $\mathbf{L} = \mathbf{D} - \mathbf{S}$ the Laplacian of the similarity matrix $\mathbf{S}$; $\mathbf{D}$ is the diagonal degree matrix with $D_{ii} = \sum_j S_{ij}$.

We use the similarity and the Laplacian matrices to constraint the learned model to treat in a similar manner features that have similar side-information. This is relatively straightforward with linear models such as $\mathbf{W}\mathbf{X}^\mathbf{T}, \mathbf{W} \in \mathbb{R}^{m \times d}$, and can be achieved through the introduction of the Laplacian regulariser $\mathrm{Tr}\left(\mathbf{W}\mathbf{L}\mathbf{W}^\mathbf{T}\right) = \sum_{ij} ||\mathbf{W}_{.i} - \mathbf{W}_{.j}||^2 S_{ij}$ in the objective function where $\mathbf{W}_{.i}$ is the ith column vector of $\mathbf{W}$ containing the model parameters associated with the ith feature, (Huang et al., 2011). The Laplacian regulariser forces the parameter vectors of the features to cluster according to the feature similarity.

However in non-linear models such neat separation of the model parameters is not possible since these are shared between the different input features. In order to achieve the same effect we will now operate directly on the model output. We will do so by requiring that the change in the model's output is marginal if we change the relative proportion of two very similar features. Concretely, let i and j be such features, and $\mathbf{e}_i, \mathbf{e}_j$, be the d-dimensional unit vectors with the ith and jth dimensions respectively equal
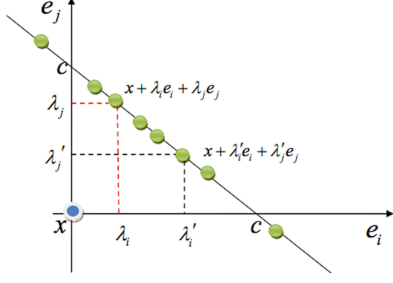
Figure 1. The blue dot is some given instance, $\mathbf{x}$. The two axes are the ith and jth features. If the two features are on the limit identical then the model's output is constant along the line defined as: $\mathbf{x} + \lambda_i \mathbf{e}_i + \lambda_j \mathbf{e}_j, \forall \lambda_i + \lambda_j = c$, where c is some constant.

to one. We want that:

$$\phi(\mathbf{x} + \lambda_i \mathbf{e}_i + \lambda_j \mathbf{e}_j) \approx \phi(\mathbf{x} + \lambda_i' \mathbf{e}_i + \lambda_j' \mathbf{e}_j) \qquad (1)$$

$\forall \lambda_i, \lambda_j, \lambda_i', \lambda_j' \in \mathbb{R}$ such that $\lambda_i + \lambda_j = \lambda_i' + \lambda_j'$

Equation (1) states that as long as the total contribution of the i, j, features is kept fixed, the model's output should be left almost unchanged. The exact equality will hold when the i, j, are on the limit identical, i.e. $S_{ij} \to \infty$. More general the level of the model's change should reflect the similarity of the i, j, features, thus a more accurate reformulation of equation 1 is:

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

$\forall \lambda_i, \lambda_j, \lambda_i', \lambda_j' \in \mathbb{R}$ such that $\lambda_i + \lambda_j = \lambda_i' + \lambda_j'$

Thus the norm of the change in the model output, that we get when we alter the relative proportion of two features i and j, while keeping their total contribution fixed, should be inversely proportional to the features similarity, i.e. large similarity, small output change. The result is that the model is symmetric to similar features and its output does not depend on the individual contributions/values of two similar features but only on their total contribution. In figure 1 we visualise the effect of the model constraint given in eq. 2. Given some instance $\mathbf{x}$ and two features i, j, that are on the limit identical the constraint forces the model output to be constant on the line defined by $\mathbf{x} + \lambda_i \mathbf{e}_i + \lambda_j \mathbf{e}_j, \forall \lambda_i + \lambda_j = c$, for some given $c \in \mathbb{R}$. We can think of the whole process as the model clustering together, to some latent factor, features that have very high similarity. The latent factor captures the original features total contribution leaving the model's output unaffected to relative changes in their values.

To unclutter notation we will define the vector $\boldsymbol{\lambda} = (\lambda_i, \lambda_j, \lambda_i', \lambda_j')$. We want the constraint of eq. 2 to be valid

over all instances drawn from $P(\mathbf{x})$ as well as for all $\boldsymbol{\lambda}$ vectors that satisfy the equality constraint eq 2. A natural measure of the degree to which the constraint holds for the feature pair i, j is given by:

$$
\begin{aligned}
R_{ij}(\boldsymbol{\phi}) &= \int ||\phi(\mathbf{x} + \lambda_i \mathbf{e}_i + \lambda_j \mathbf{e}_j) \qquad (3) \\
&\quad -\phi(\mathbf{x} + \lambda_i' \mathbf{e}_i + \lambda_j' \mathbf{e}_j)||^2 \\
&\quad S_{ij} \mathbf{I}(\boldsymbol{\lambda}) P(\mathbf{x}) d\boldsymbol{\lambda} d\mathbf{x}
\end{aligned}
$$

where $\mathbf{I}(\boldsymbol{\lambda}) = 1$ if $\lambda_i + \lambda_j = \lambda_i' + \lambda_j'$, and 0 otherwise. Since we want to define a regulariser that accounts for all feature pairs and their similarities we simply have:

$$R(\boldsymbol{\phi}) = \sum_{ij} R_{ij}(\boldsymbol{\phi}) \qquad (4)$$

Calculating the regularizer is problematic due to the presence of the $\mathbf{I}(\boldsymbol{\lambda})$ function that selects the $\boldsymbol{\lambda}$ subspace over which the integration is performed. In the next two sections we will give two ways to approximate it. The first one will be analytical relying on the first order Taylor expansion of $\phi(\mathbf{x})$ and its Jacobian. The second one stochastic, essentially performing data augmentation and defining a regularisation term along the lines of eq. 2.

2.1. An analytical Approximation

We will use the first order Taylor expansion of $\phi(\mathbf{x})$ to simplify the squared term in eq. 2 by removing the $\boldsymbol{\lambda}$ variable. We will start by using the first order Taylor expansion to approximate the value of $\phi(\mathbf{x} + \lambda_i \mathbf{e}_i + \lambda_j \mathbf{e}_j)$ at $\mathbf{x}$

$$\phi(\mathbf{x} + \lambda_i \mathbf{e}_i + \lambda_j \mathbf{e}_j) \approx \phi(\mathbf{x}) + \mathbf{J}(\mathbf{x})(\lambda_i \mathbf{e}_i + \lambda_j \mathbf{e}_j)$$

$\mathbf{J}(\mathbf{x}) \in \mathbb{R}^{m \times d}$ is the Jacobian of $\phi(\mathbf{x})$ evaluated at $\mathbf{x}$. Then plugging the Taylor expansion in eq 2 we get:

$$||(\lambda_i - \lambda_i')\mathbf{J}(\mathbf{x})\mathbf{e}_i - (\lambda_j' - \lambda_j)\mathbf{J}(\mathbf{x})\mathbf{e}_j||^2 \propto \frac{1}{S_{ij}} \qquad (5)$$

and since $\lambda_i + \lambda_j = \lambda_i' + \lambda_j'$ we have $(\lambda_i - \lambda_i') = (\lambda_j' - \lambda_j)$ and eq 5 becomes:

$$||\mathbf{J}(\mathbf{x})\mathbf{e}_i - \mathbf{J}(\mathbf{x})\mathbf{e}_j||^2 = ||\nabla_i \phi(\mathbf{x}) - \nabla_j \phi(\mathbf{x})||^2 \propto \frac{1}{S_{ij}} \quad (6)$$

where $\nabla_i \phi(\mathbf{x})$ is the m-dimensional partial derivative of $\phi(\mathbf{x})$ with respect to the ith input feature. Using eq 6 we can approximate R_{ij} as follows:

$$R_{ij}(\boldsymbol{\phi}) \approx \int ||\nabla_i \phi(\mathbf{x}) - \nabla_j \phi(\mathbf{x})||^2 S_{ij} P(\mathbf{x}) d\mathbf{x}$$

from which we get the following approximation of the $R(\phi)$ regulariser:

$$
\begin{aligned}
R(\phi) &\approx \sum_{ij} \int ||\nabla_i \phi(\mathbf{x}) - \nabla_j \phi(\mathbf{x})||^2 S_{ij} P(\mathbf{x}) d\mathbf{x} \\
&\approx \int \sum_{ij} ||\nabla_i \phi(\mathbf{x}) - \nabla_j \phi(\mathbf{x})||^2 S_{ij} P(\mathbf{x}) d\mathbf{x} \\
&\approx \int \mathrm{Tr}[\mathbf{J}(\mathbf{x})\mathbf{L}\mathbf{J}^{\mathrm{T}}(\mathbf{x})] P(\mathbf{x}) d\mathbf{x} \qquad (7)
\end{aligned}
$$

which is the local linear approximation of the original regulariser eq 4 on the input instances. Since we only have access to the training sample and not to $P(\mathbf{x})$ we will get the sample estimate of eq. 7 given by

$$
\begin{aligned}
\hat{R}(\phi) &= \sum_{ij} \sum_k ||\nabla_i \phi(\mathbf{x}_k) - \nabla_j \phi(\mathbf{x}_k)||^2 S_{ij} \\
&= \sum_k \sum_{ij} ||\nabla_i \phi(\mathbf{x}_k) - \nabla_j \phi(\mathbf{x}_k)||^2 S_{ij} \\
&= \sum_k \mathrm{Tr}[\mathbf{J}(\mathbf{x}_k)\mathbf{L}\mathbf{J}^{\mathrm{T}}(\mathbf{x}_k)] \qquad (8)
\end{aligned}
$$

So the sample based estimate of the regulariser is a sum of Laplacian regularisers applied on the Jacobian of each one of the training samples. It forces the partial derivatives of the model with respect to the input, or equivalently the model's sensitivity to the input features, to reflect the features similarity in the local neighborhood around each training point. Or in other words it will constrain the learned model in a small neighborhood around each training point to have similar slop in the dimensions that are associated with similar features. Note that if $\phi(\mathbf{x}) = \mathbf{W}\mathbf{x}$ then $\mathbf{J}(\mathbf{x}_k) = \mathbf{W}$ and $\mathrm{Tr}[\mathbf{J}(\mathbf{x}_k)\mathbf{L}\mathbf{J}^{\mathrm{T}}(\mathbf{x}_k)]$ reduces to the standard $\mathrm{Tr}[\mathbf{W}\mathbf{L}\mathbf{W}^{\mathrm{T}}]$ Laplacian regulariser on the columns of $\mathbf{W}$ associated with the input features. Adding the sample based estimate of the regulariser to the loss function we get the final objective function which we minimize with $\phi(\mathbf{x})$ giving the following minimization problem under the analytical approximation:

$$
\min_{\phi} \sum_k L(\mathbf{y}_k, \phi(\mathbf{x}_k)) + \lambda \sum_k \mathrm{Tr}[\mathbf{J}(\mathbf{x}_k)\mathbf{L}\mathbf{J}^{\mathrm{T}}(\mathbf{x}_k)] \quad (9)
$$

The approximation of the requlariser is only effective locally around each training point since it relies on first order Taylor expansion. When the learned function is highly nonlinear, it can force model invariance only to small relative changes in the values of two similar features. However, as the size of the relative changes increases and we move away from the local region the approximation is no longer effective. The regulariser will not be powerful enough to make the invariance hold away from the training points. If we want a less local approximation we can either use higher order Taylor approximation which is computationally prohibitive or rely on a more global approximation through

data augmentation as we will see in the next section. Note also that the presence of the Jacobian in the objective function means that if we optimise it using gradient descent we will need to compute second order partial derivatives which come with an increasing computational cost.

2.2. A stochastic Approximation

Instead of using the first order Taylor expansion to simplify the squared term required by the regulariser we can use sampling to approximate it. Concretely for a given feature pair, i, j, and a given instance $\mathbf{x}$ we randomly sample p quadruples $\lambda_i^{(l)}, \lambda_j^{(l)}, \lambda_i^{(l)'}, \lambda_j^{(l)'} \in \mathbb{R}$ such that $\lambda_i^{(l)} + \lambda_j^{(l)} = \lambda_i^{(l)'} + \lambda_j^{(l)'}$, $l := 1 \ldots p$, which we use to generate p new instance pairs as follows:

$$
\mathbf{x} \to \begin{cases} \mathbf{x} + \lambda_i^{(l)}\mathbf{e}_i + \lambda_j^{(l)}\mathbf{e}_j \\ \mathbf{x} + \lambda_i^{(l)'}\mathbf{e}_i + \lambda_j^{(l)'}\mathbf{e}_j \end{cases}
$$

We can now use the training sample and the sampling process to get an estimate of $R_{ij}(\phi)$ by:

$$
\begin{aligned}
\sum_k \sum_l ||(\phi(\mathbf{x}_k + \lambda_i^{(l)}\mathbf{e}_i + \lambda_j^{(l)}\mathbf{e}_j) \\
-\phi(\mathbf{x}_k + \lambda_i^{(l)'}\mathbf{e}_i + \lambda_j^{(l)'}\mathbf{e}_j))||^2 S_{ij}
\end{aligned}
$$

and of the final regulariser $R(\phi)$ by:

$$
\begin{aligned}
\tilde{R}(\phi) &= \sum_{ij} \sum_k \sum_l ||(\phi(\mathbf{x}_k + \lambda_i^{(l)}\mathbf{e}_i + \lambda_j^{(l)}\mathbf{e}_j) \\
&\quad -\phi(\mathbf{x}_k + \lambda_i^{(l)'}\mathbf{e}_i + \lambda_j^{(l)'}\mathbf{e}_j))||^2 S_{ij} \quad (10)
\end{aligned}
$$

So the final optimization problem will now become:

$$
\min_{\phi} \sum_k L(\mathbf{y}_k, \phi(\mathbf{x}_k)) + \lambda \tilde{R}(\phi) \qquad (11)
$$

Note that the new instances appear only in the regulariser and not in the loss. The regulariser will penalise models which do not have the invariance property with respect to pairs of similar features. In practice when computing $\tilde{R}(\phi)$ we do not want to go through all the pairs of features but only through the most similar. We do not want to spend sampling time on data augmentation for dissimilar pairs since for these there is no effective constraint on the values of the model's output. So we simplify the sum run only over the pairs of similar features. One motivation for the stochastic approach was the fact that the analytical one relies in an approximation which is only effective locally in the neighborhood of each learning instance. In the stochastic approach we have control on the size of the neighborhood over which the constraint is enforced through the Euclidean norm of the change vector (λ_i, λ_j); the larger its

value the larger the neighborhood. The smaller the neighborhood the closer we are to the local behavior of the analytical approximation. We should note here that the sampling of stochastic approximation will naturally blend with the stochastic gradient optimization that we will use to optimize our objective functions.

2.3. Optimization

We learn ϕ with a standard feed forward neural network with sigmoid activation functions applied on the hidden layers using stochastic gradient descent. The objective function of the analytical approach contains the Jacobian of the model with respect to its input. Calculating the gradient over this results in the introduction of second order partial derivatives of the model with respect to the inputs and the model parameters. Bishop, 1992, gave a backpropagation algorithm for the exact calculation of the Hessian of the loss of a multi-layer perceptron. We have adapted this algorithm so that we can compute the gradient of objective functions that contain the Jacobian with respect to the input features, we give the complete gradient calculation procedure in the appendix.

We will give now the computational complexity of each the two methods. We will denote by l the number of layers, m the output dimension of the network, h_k the number of hidden units of the kth layer and we will define $h_{\max} = \max\{h_k | k = 1, \ldots, l-1\}$. The computational complexity of computing the gradient for a single instance of the objective function of the analytical approach is $\mathcal{O}(m \times h_1 \times d^2)$ for networks with a single hidden layer and $\mathcal{O}(l \times m \times h_{\max}^2 \times d^2 + l \times m \times h_{\max}^3 \times d)$ for networks with more than one hidden layers. To reduce this computational complexity in our experiments we sparsify $\mathbf{S}$ by keeping only the entries corresponding to top 20% biggest elements and zero out the rest. The complexity now becomes $\mathcal{O}(m \times h_1 \times d)$ for one layer networks and $\mathcal{O}(l \times m \times h_{\max}^2 \times d + l \times m \times h_{\max}^3)$ for networks with more than one layer. The computational complexity of the stochastic approach is $\mathcal{O}(l \times h_{\max}^2 \times m \times p + l \times h_{\max} \times m \times p)$ while the computational complexity for standard feed forward network is $\mathcal{O}(l \times h_{\max}^2 + l \times h_{\max} \times m)$.

3. Related Work

Krupka & Tishby, 2007, use feature side-information, they call it meta-features, within a linear SVM model. They force the SVM's weights to be similar for features that have similar side-information. They achieve that through the introduction of a Gaussian prior on the feature weight vector. The covariance matrix of the Gaussian is a function of the features similarity. The authors show how to extend their approach from linear to polynomial models. However, their approach requires explicit calculation of all the higher or-

der terms limiting its applicability to low order polynomials. Similar work is the Laplacian-based feature regularisation, (Huang et al., 2011), which constraints the feature weights to reflect relations that are given by the Laplacian. The Laplacian matrix is constructed from available domain knowledge, what here we call feature side-information. It can also be constructed from the data; for example as a function of the feature correlation matrix.

The Laplacian Regulariser (LR) has connections to some Lasso-based regularisers. With LR feature weights reflect the feature similarity given by the side information. The fused, graph, and group lasso variants (Tibshirani et al., 2005; Yuan & Lin, 2006) bring in and out of the model feature groups; here feature similarity is whether features belong in the same group or not (1/0 similarity) given by domain knowledge. If yes their weight values will either be all zero (out of the model) or different than zero (in the model) but not necessarily the same. Fused lasso in addition to that pushes subsequent (connected) features to have similar weights being even more similar to LR. Such lasso-based regularisers are also applicable only in linear models where we have a clear grasp of how each feature is treated by the model. It is not clear how to use them on non-linear models. The analytical approach can be adapted easily for such regularisers making the approach interesting/relevant for regularisers other than the Laplacian. On a similar note Rosasco et al., 2013, applied lasso regularisation on the Jacobian of kernel-based models to learn non-linear models that are sparse with respect to the input features.

The Taylor expansion in the analytical approximation of the regulariser brings in the Jacobian of the model. Regularisers that use the Jacobian have previously been successfully used to control the stability/robustness of models to noisy inputs. Relevant work includes contractive auto encoders, (Rifai et al., 2011b), and the work of (Zhai & Zhang, 2015). (Rifai et al., 2011b) use the Frobenius of the Jacobian at the input instances to force the model to be relatively constant in small neighbors around the input instances. Such a regulariser introduces invariance to small input variations.

Optimizing the Jacobian in networks with more than one layer is cumbersome, thus very often the stochastic approach is preferred over the analytic e.g. (Zhai & Zhang, 2015). Denoising autoencoders, (Vincent et al., 2010) follow the stochastic paradigm and require that small random variations in the inputs have only a limited effect on the model output. Zheng et al., 2016, used Gaussian perturbations to stabilise the network's output with respect to variations in the input, essentially augmenting the training data. Regularisers on higher order derivatives, Hessian, are also used, (Rifai et al., 2011a), in such cases the stochastic approach is the only choice due to the prohibitive cost of optimizing the Hessian term. Data augmentation is

also used when we have additional prior knowledge on the instance structures to which the models should be invariant. In imaging problems such structures include translations, rotations, scalings,etc, (Simard et al., 1991; Decoste & Schölkopf, 2002). (Corduneanu & Jaakkola, 2002) learn models with built-in invariance by imposing class invariance in dense regions of the feature space; we impose invariance to directions determined by the side information.

Closer to our work are (Krupka & Tishby, 2007) and the works that use Laplacian based regularisers for model regularisation, e.g. (Huang et al., 2011). However to the best of our knowledge all previous work was strictly limited to linear models. We show how to apply such regularisers and constraints to general classes of non-linear models.

4. Experiments

We will experiment and evaluate our regularisers in two settings, a synthetic and a real world one. We will compare the analytical and the stochastic regulariser, which we will denote by AN and ST respectively, against popular regularisers used with neural networks, namely ℓ_2 and Dropout (Srivastava et al., 2014), over different network architectures. In the real world datasets we also give the results of the Word Mover's Distance, WMD, (Kusner et al., 2015) which makes direct use of the side-information to compute document distances. Obviously our regularisers and WMD have an advantage over ℓ_2 and dropout since it exploit side-information which ℓ_2 and dropout do not.

We trained both the analytical and the stochastic models, as well as all baselines against which we compare, using Adam (Kingma & Ba, 2014). We used $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$ for one hidden layer networks, and $\alpha = 0.0001$ for the networks with more hidden layers. We initialize all networks parameters using (Glorot & Bengio, 2010). Due to the large computational complexity of the analytical approach we set the mini-batch size m to five. For the stochastic model, as well as for all the baseline models, we set the mini-batch size to 20. For the analytical model we set the maximum number of iterations to 5000. For the stochastic model we set the maximum number of iterations to 10000 for the one layer networks and to 20000 for networks with more layers. We used early stopping where we keep 20% of the training data as the validation set. Every five parameter updates we compute the validation error. Training terminates either if we reach the maximum iteration number or the validation error keeps increasing more than ten times in a row.

In the stochastic approach we do the sampling for the generation of the instance pairs within the stochastic gradient descent process. Concretely for each instance $\mathbf{x}$ in a mini batch we randomly chose a feature pair i, j, from

the set of similar feature pairs. We sample a quadruple $\{\lambda_i, \lambda_j, \lambda_i', \lambda_j'\}$ from $\mathbb{R}$ respecting the constraint: $\lambda_i + \lambda_j = \lambda_i' + \lambda_j'$ from which we generate the respective instance pairs. We repeat the process p times each time sampling a new feature pair i, j, and a new quadruple. We fix the set of similar feature pairs to be the top 20% of most similar feature pairs. Thus within each mini-batch of size m we generate $m \times p$ instance pairs and we accumulate the norm of the respective model output differences in the objective. In the experiments we set $p = 5$

4.1. Artificial Datasets

We design a simple data generation process in order to test the performance of our regularisers when the data generation mechanism is compatible with the assumptions of our models. We randomly generate an instance matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ by uniformly sampling instances from $\mathbb{R}^d$. We create $d/2$ feature clusters as follows. To each one of these clusters we initially assign one of the original input features without replacement. We assign randomly and uniformly the remaining $d/2$ features to the clusters. We use the feature clusters to define a latent space where every feature cluster gives rise to a latent feature. The value of each latent feature is the sum of the values of the features that belong to its cluster. On the latent space representation of the training data we apply a linear transformation that projects the latent space to a new space with lower dimensionality m. On this lower dimensionality space we apply an element-wise sigmoid and the final class assignment is given by the index of the maximum sigmoid value. The similarity S_{ij} of the i, j, features of the original space is 1 if they ended up in the same cluster and 0 otherwise. We set $d = 3000$, $n = 5000$, $m = 5$. We will call this dataset A1. The generating procedure gave a very sparse $\mathbf{S}$ matrix with only 0.04% of its entries being non-zero. Each feature had an average of 1.3 similar features. We used 4000 instances for training and the rest for testing. During training 20% of the instances are used for the validation set. We train all algorithms on the original input space. For all regularisers we used a network with single hidden layer with 100 hidden units. We tune the hyperparameters based on the performance on the validation set. We select the λ hyperparameters of of AN, ST, and ℓ_2 from $\{10^k | k = -3, \ldots, 3\}$; we select the λ of dropout from $[0.1, 0.2, 0.3, 0.4, 0.5]$. We set the c in the augmentation process, that controls the size of the neigborhood within which the output constraints should hold, to one.

Both the analytical and the stochastic regulariser bring performance improvements of roughly 10% when compared to the ℓ_2 regulariser and to Dropout, results in table 1. In figure 2 we plot the error on the validation set as a function of epoch number and the real time for the four regularisers. The analytical and stochastic regulariser require consider-

ably less epochs to converge and that to signicigantly lower error values than either ℓ_2 or dropout. However, since the computational complexity of each step is larger, the real time to convergence is larger than that of ℓ_2 and dropout.
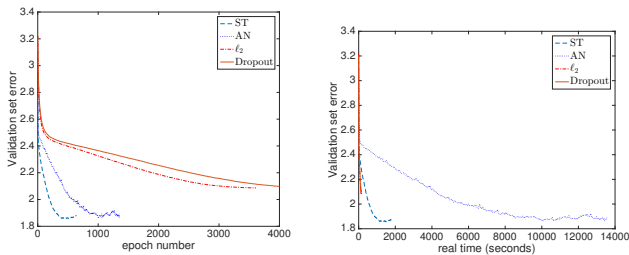

Figure 2. Validation set error for the different regularisers: as a function of epoch number (left), as a function of realtime (right).

The regularizer we propose constrains the model structure by forcing the model to reflect the feature similarity as these are given in the similarity matrix. Thus we expect the structure of the similarity matrix to have an impact on the performance of the regulariser. To see that let us consider the trivial case in which **S** is diagonal. In this case the input and the latent spaces are equivalent. and the regulariser has no effect since there are no similarity constraints. If all features are identical, i.e. $S_{ij} = 1, \forall i, j$, then the latent space will have a dimensionality of one, and the regulariser has the strongest effect.

To explore this dependency we generate two additional synthetic datasets where we use the same generating mechanism as in A1 but vary the proportion of features we cluster together to generate latent factors. Concretely in the synthetic dataset we will call A2 we randomly select a set A of $d/2$ features over which we will perform clustering to define latent factors. We use the remaining set B of $d/2$ features directly as they are in the latent space. We cluster the features of the A set to $d/4$ clusters—latent factors, making sure that as in A1 each cluster has at least one feature in it. As a result the final latent space has a dimensionality of $d/4 + d/2 = 3d/4$. To generate the class assignments we proceed as in A1. To generate the third dataset, A3, we select $d/4$ features to generate A and the remaining for B. We now cluster the features in A to $d/8$ clusters, again making sure that there is at least one feature pre cluster. The dimensionality of the latent space is now $d/8 + 3d/4 = 7d/8$; class assignments are generated as above. We used the same values for n, d, m as in A1. As we move from A1 to A3 we reduce the number of features that are similar to other features, thus we increase the sparsity of **S**. For A2 and A3 the percentage of non-zero elements is 0.021% and 0.011% respectively, compared to 0.04% we had in A1. So A1 is the datasets that has most constraints while A3 is the one with the least constraints. We apply the different regularisers in these two datasets

using exactly the same protocol as in A1. The results are also given in table 1. As we see the classification error of both ST and AN increases as the dataset sparsity increases and it approaches that of the standard regularisers.

Dataset	$\vert\{S_{ij} \neq 0\}\vert$	ST	AN	ℓ_2	Dropout
A1	0.04%	43.70	44.00	52.70	53.60
A2	0.021%	50.08	51.00	55.40	55.30
A3	0.011%	56.20	52.50	54.50	55.90

Table 1. Classification error, %, of the different regularizers, and % of non zero elements of the similarity matrix **S** for the three artificial datasets.

4.2. Real World Datasets

We evaluated both approaches on the eight document classification datasets used in (Kusner et al., 2015). We removed all the words in the SMART stop word list (Salton & Buckley, 1988). Documents are represented as bag of words. To speed up training, we removed words that appear very few times over all the documents of a dataset. Concretely, in 20NEWS we reduce the dictionary size by removing words with a frequency less or equal to three. In the OHSUMED and CLASSIC datasets we remove words with frequency one and the in REUTER dataset words with frequency equal or less than two. As feature side-information we use the word2vec representation of the words which have a dimensionality of 300 (Mikolov et al., 2013). In table 2 we give a description of the final datasets on which we experiment including the number of classes (m) and average number of unique words per document.

Date set	n	d	Unique words(avg)	m
BBCsport	590	9759	80.9	5
Twitter	2486	4076	6	3
Classic	5675	7628	34.5	4
Amazon	6400	4502	28.8	4
20NEWS	11293	6859	51.7	20
Recipe	3496	4992	44.7	15
Ohsumed	3999	7643	50	10
Reuter	5485	5939	33	8

Table 2. Dataset description

We compute the similarity matrix **S** from the word2vect word representations using the heat kernel with bandwidth parameter σ, i.e. the similarity of i, j, features is given by: $S_{ij} = \exp(-\frac{1}{2\sigma^2}(\mathbf{z}_i - \mathbf{z}_j)^T(\mathbf{z}_i - \mathbf{z}_j))$. We select σ so that roughly 20% of the entries of the similarity matrix are in $[0.8, 1]$ interval.

For those datasets without a predefined train/test split (BBCsport, Twitter, Classic, Amazon, Recipe), we use five-fold cross validation and report the average error. We control the statistical significance of the results using the Mac-Nemar's test ($p-$value=0.05). We tune the hyperparame-

ters with three-fold inner cross validation. We select the λs of AN, ST, and ℓ_2 from $[0.001, 0.01, 0.1, 1, 10]$; we select the λ of dropout from $[0.1, 0.2, 0.3, 0.4, 0.5]$. We did a series of experiments in which we varied the number of hidden layers. Due to the computational complexity of the backprogation for the AN regulariser we only give results for the single layer architecture.

In the first set of experiments we use a neural network with one hidden layer and 100 hidden units, we give the results in table 3. ST is significantly better than the AN in six out of the eight datasets, significantly worse once, and equivalent in one dataset. ST is significantly better than the ℓ_2 in six out of the eight daasets, while it is equivalent in one. Compared to dropout it is four times significantly better and two times significantly worse.

When we increase the number of hidden layers to two with 500 and 100 units on the first and second layer ST method is significantly better compared to ℓ_2 three times, significantly worse three times, while there is no significant difference in two datasets. A similar picture emerges with respect to Dropout with ST being significantly better three times, significantly worse twise, while in three cases there is no significant difference. We give the detailed results in table 4.

Dataset	ST	AN	ℓ_2	Dropout	WMD
BBCsport	03.39-=-	02.17==	02.72=	02.17	04.6
Twitter	26.90+++	31.18=-	31.18-	28.44	29.00
Classic	03.54+++	04.03+=	05.13-	03.98	02.80
Amazon	06.25++-	07.80=-	07.57-	06.44	07.40
20NEWS	19.58+++	23.75=-	23.21-	21.31	27.00
Recipe	38.76+++	43.14–	41.21-	39.88	43.00
Ohsumed	34.45+==	35.75=-	35.26=	34.39	44.00
Reuter	03.84=+-	03.38+=	06.03-	03.20	03.50

Table 3. Classification error, %, with one hidden layer NNs. AN: analytical approach, ST: stochastic approach. WMD results are from (Kusner et al., 2015). The $+, -$ and $=$ signs give the significance test results of the comparison of the performance of a given regulariser to those of the regularisers in the subsequent columns; indicating respectively significantly better, worse, no difference. WMD is not included in the significance comparison since at the time of the experiments we did not have acces to the code.

Dataset	ST	ℓ_2	Dropout	WMD
BBCsport	02.04=+	02.85=	02.99	04.60
Twitter	27.93–	26.64=	26.74	29.00
Classic	03.71+=	04.51-	03.69	02.80
Amazon	05.96++	07.49-	06.75	07.40
20NEWS	20.72++	22.49=	22.48	27.00
Recipe	41.53–	39.61=	39.31	43.00
Ohsumed	35.03==	34.95=	35.14	44.00
Reuter	04.39-=	03.88=	04.16	03.50

Table 4. Classification error, %, with two hidden layers network. Table interpretation as in table 3

5. Conclusion and Future Work

Many real world applications come with additional information describing the properties of the features. Despite that, quite limited attention has been given to such setting. In this paper we develop a regulariser that exploits exactly such information for general non-linear models. It relies on the simple intuition that features which have similar properties should be treated by the learned model in a similar manner. The regulariser imposes a stability constraint over the model output. The constraint forces the model to produce similar outputs for instances the feature values of which differ only on similar features. We give two ways to approximate the value of the regulariser. An analytical one which boils down to the imposition of a Laplacian regulariser on the Jacobian of the learned model with respect to the input features and a stochastic one which relies on sampling.

We experiment with neural networks with the two approximations of the regulariser and compare their performance to well established model regularisers, namely ℓ_2 and dropout, on artificial and real world datasets. In the artificial datasets, for which we know that they match the assumptions of our regulariser we demonstrate significant performance improvements. In the real world datasets the performance improvements are less striking. One of the main underlying assumptions of our model is that the feature side-information is indeed relevant for the learning problem, when this is indeed the case we will have performance improvements. If it is not the case then the regulariser will not be selected, as a result of the tuning of the λ parameter.

Along the same lines we want to perform a more detailed study on how the structure of the similarity matrix, namely its sparsity and the underlying feature cluster structure, determines the regularisation strength of our regulariser. It is clear that a sparse similarity matrix will lead to a rather limited regularisation effect since only few features will be affected. This points to the fact that the regulariser should be used together with more traditional sparsity inducing regularisers, especially in the case of a sparse feature similarity matrix. Finally since we use the feature information through a similarity function it might be the case that the similarity function that we are using is not appropriate and better results can be obtained if we also learn the feature similarity. We leave this for future work.

References

Bishop, Christopher. Exact calculation of the hessian matrix for the multilayer perceptron. *Neural Computation*, 4:494–501, January 1992.

Bishop, Christopher M. *Neural networks for pattern recognition*. Oxford university press, 1995.

Chiang, Kai-Yang, Hsieh, Cho-Jui, and Dhillon, Inderjit S. Matrix completion with noisy side information. In *Advances in Neural Information Processing Systems*, pp. 3447–3455, 2015.

Chiang, Kai-Yang, Hsieh, Cho-Jui, and Dhillon, EDU Inderjit S. Robust principal component analysis with side information. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 2291–2299, 2016.

Corduneanu, Adrian and Jaakkola, Tommi. On information regularization. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pp. 151–158. Morgan Kaufmann Publishers Inc., 2002.

Decoste, Dennis and Schölkopf, Bernhard. Training invariant support vector machines. *Machine Learning*, 46 (1):161–190, 2002. ISSN 1573-0565. doi: 10.1023/A: 1012454411458.

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pp. 249–256, 2010.

Huang, Jian, Ma, Shuangge, Li, Hongzhe, and Zhang, Cun-Hui. The sparse laplacian shrinkage estimator for high-dimensional regression. *The Annals of Statistics*, 39(4): 2021–2046, 2011. ISSN 00905364.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krupka, Eyal and Tishby, Naftali. Incorporating prior knowledge on features into learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 227–234, 2007.

Krupka, Eyal, Navot, Amir, and Tishby, Naftali. Learning to select features using their properties. *Journal of Machine Learning Research*, 9(Oct):2349–2376, 2008.

Kusner, Matt J, Sun, Yu, Kolkin, Nicholas I, and Weinberger, Kilian Q. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, pp. 957–966, 2015.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

Rao, Nikhil, Yu, Hsiang-Fu, Ravikumar, Pradeep K, and Dhillon, Inderjit S. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in Neural Information Processing Systems*, pp. 2098–2106, 2015.

Rifai, Salah, Mesnil, Grégoire, Vincent, Pascal, Muller, Xavier, Bengio, Yoshua, Dauphin, Yann, and Glorot, Xavier. Higher order contractive auto-encoder. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part II*, pp. 645–660, 2011a.

Rifai, Salah, Vincent, Pascal, Muller, Xavier, Glorot, Xavier, and Bengio, Yoshua. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 833–840, 2011b.

Rosasco, Lorenzo, Villa, Silvia, Mosci, Sofia, Santoro, Matteo, and Verri, Alessandro. Nonparametric sparsity and regularization. *Journal of Machine Learning Research*, 14(1):1665–1714, 2013.

Salton, Gerard and Buckley, Christopher. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

Simard, Patrice Y., Victorri, Bernard, LeCun, Yann, and Denker, John S. Tangent prop - A formalism for specifying selected invariances in an adaptive network. In Moody, John E., Hanson, Stephen Jose, and Lippmann, Richard (eds.), *Advances in Neural Information Processing Systems 4, [NIPS Conference, Denver, Colorado, USA, December 2-5, 1991]*, pp. 895–903. Morgan Kaufmann, 1991.

Srivastava, Nitish, Hinton, Geoffrey E, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

Tibshirani, Robert, Saunders, Michael, Rosset, Saharon, Zhu, Ji, and Knight, Keith. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

Vapnik, Vladimir and Izmailov, Rauf. Learning using privileged information: Similarity control and knowledge

transfer. *Journal of Machine Learning Research*, 16: 2023–2049, 2015.

Vincent, Pascal, Larochelle, Hugo, Lajoie, Isabelle, Bengio, Yoshua, and Manzagol, Pierre-Antoine. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.

Xing, Eric P., Ng, Andrew Y., Jordan, Michael I., and Russell, Stuart J. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pp. 505–512, 2002.

Yuan, Ming and Lin, Yi. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Zhai, Shuangfei and Zhang, Zhongfei. Manifold regularized discriminative neural networks. *CoRR*, abs/1511.06328, 2015. URL http://arxiv.org/abs/1511.06328.

Zheng, Stephan, Song, Yang, Leung, Thomas, and Goodfellow, Ian J. Improving the robustness of deep neural networks via stability training. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 4480–4488, 2016.