# BIAS LEARNING, KNOWLEDGE SHARING

Joumana Ghosn and Yoshua Bengio
Dept. Informatique et Recherche Opérationnelle
Université de Montréal, Montreal, Qc H3C-3J7
*ghosn@iro.umontreal.ca, bengioy@iro.umontreal.ca*

**Abstract**

Biasing the hypothesis space of a learner has been shown to improve generalisation performances. Methods for achieving this goal have been proposed, that range from deriving and introducing a bias into a learner to automatically learning the bias. In the latter case, most methods learn the bias by simultaneously training several related tasks derived from the same domain and imposing constraints on their parameters. We extend some of the ideas presented in this field and describe a new model that parameterizes the parameters of each task as a function of an affine manifold defined in parameter space and a point lying on the manifold. An analysis of variance on a class of learning tasks is performed that shows some significantly improved performances when using the model.

## 1 Introduction

It has been suggested that the bias/variance dilemma can be circumvented by willfully introducing "bias" (in the sense of a structure restricting the class of functions) in a learning model, representing known properties about the problem being tackled. Several techniques have been proposed to achieve this goal for specific applications. But deriving and introducing a bias into a model is still a very difficult and complex task.

Recently, methods have been suggested that are useful when the bias cannot be easily incorporated in the learning model or when it is not explicit enough to be used. These techniques consist in automatically learning the bias and can be used when several related tasks derived from the same domain are available. The models being trained to learn the tasks are forced to share some "knowledge" about the domain by imposing constraints on the parameters of each model. An example in which these techniques can be applied is the problem of predicting the stock returns of different companies operating within the same economic sector. Besides being influenced by general factors related to the economy, these stocks are also affected by developments specific to the sector. The analysis of the stocks can be improved by allowing the models learning about the different stocks to share the knowledge they acquire when learning from examples of each stock [4].

## 2 Multi-Task Learning

Multi-task learning methods consider that a learner is embedded in an environment where it faces many related tasks, and that knowlege acquired when learning a task can be used to better learn a new task. Several methods of knowledge transfer between models trained on different but related tasks derived from the same domain have been proposed. They could be divided into two main categories: sequential vs parallel knowledge transfer methods, and methods that aim at reducing learning time vs methods that are used to improve the generalisation performance.

A sequential transfer of knowledge is used when the related tasks are not all available at the same time. The domain information acquired by a task at a given time is used in the training of tasks that arise in the future. Each model is therefore trained separately and only future tasks can benefit from the information of previous tasks. This kind of transfer has been mostly used to reduce the training time of new models.

A parallel transfer of knowledge is used when all the related tasks are available at the same time. All the models that are to learn the tasks are trained simultaneously thus allowing each model to benefit from the information acquired by all the other models. Most of the methods of parallel transfer aim at improving the generalisation performance.

Some methods of knowledge transfer were derived for use in specific applications. Examples include the "Discriminability-Based Transfer" method [6] and the "Task Clustering" method [7]. Both methods are intended to be used for classification problems. The first method allows a sequential transfer that reduces learning time for new tasks while the second one performs a parallel knowlege transfer and has been shown to improve generalisation performances for robot navigation problems.

One of the most general-purpose methods (for which several variants exist) consists in learning an internal representation using similar tasks [1, 2]. It consists in training simultaneously several neural networks and forcing them to share their first hidden layers (i.e. the weights of the first hidden layers are the same across all the networks), while their remaining layers are distinct. The shared layers serve as a mutual source of inductive bias and the output of the last shared layer is considered to be the learned internal representation. This method can be viewed as projecting the examples of a task in a new space that is easier to analyze. The projection is based on a transform learned using examples sampled from all the tasks. This method can be extended as follows: instead of sharing the first hidden layers, these layers can be distinct, and the last layers could be shared. Such a system would be interpreted as applying a different preprocessing to the examples of different tasks and then applying the same analysis to preprocessed examples of different tasks [4].

In the "family discovery" method [5], a parameterized family of models is built. Several learners are trained separately on different but related tasks and their parameters are used to construct a mixture of affine manifolds of parameters. The manifolds are defined by the top eigenvectors obtained in a principal component analysis (PCA) of the parameters of the learners. A variant of the Expectation Maximization algorithm is then used to refit the parameters of the learners in the context of the mixture of manifolds, and the mixture of manifolds in the context of the parameters.

## 3 Bias Learning Using An Affine Manifold In Parameter Space

We introduce in this section a general framework for bias learning which consists in re-parameterizing the parameters $P_i$ of each model or learner $i$ as follows: $P_i = f(\theta, \alpha_i)$, where $P_i \in \mathbb{R}^{n_1}$, $\theta \in \mathbb{R}^{n_2}$, $\alpha_i \in \mathbb{R}^{n_3}$ with $n_1 \geq n_3$, and $n_1$ corresponds to the parameter space dimensionality. Each learner $i$ in this model is defined by a set of "private" parameters $\alpha_i$ which are transformed according to a chosen function $f$ defined by a set of "shared" parameters $\theta$. The "shared" parameters are updated using examples sampled from all the tasks while the "private" parameters are updated using examples only sampled from the corresponding task. Such a framework requires the user to choose the type of the function $f$. The choice of $f$ can be guided by apriori knowledge about the domain or can be generic if such knowledge is not available or cannot be easily translated as a function.

We propose a generic choice of $f$ that constrains the parameters $P_i$ of each learner $i$ to lie on a surface defined in the parameter space and whose dimensionality is smaller than that of the parameter space. In that case, the "private" parameters of each model represent the "position" of a point on the surface and the surface whose defining parameters are learned using all the tasks, is considered to embed knowledge about the domain. The simplest form for the surface is an affine manifold. More complex choices include mixtures of affine manifolds or non-linear surfaces. A trade-off between the "complexity" of the surface on the one hand and the number of available examples for each task, as well as the nature of the tasks on the other hand must be considered in order to avoid over-fitting related problems.

In the remainder of this paper, an affine manifold is considered to define a parameter sub-space or surface. A description of the corresponding model is provided in the following subsections. Results using this model on a class of learning tasks are provided in section 4. The generalization of this model to more complex surfaces is straightforward.

### 3.1 Defining The Global Model's Parameters

Let $P_i = \{P_{ij}\}, j = 1, ..., N$ represent the $N$ parameters of a model $i$. For a neural network, $P_i$ will represent all the $N$ weights and biases of the network. Here, all the models $i$ are assumed to have the same architecture and the same number of parameters.

In order to define a $d$-dimensional affine manifold (where $d \leq N$) in an $N$-dimensional parameter space, $(N - d)d$ parameters are needed to define the direction of the manifold, and $N$ parameters are necessary to specify its offset with respect to the origin of the parameters coordinate system.

The relationship between such a manifold and the parameters $P_i$ which must lie on it can be expressed as follows:

$$P_i = \theta\alpha_i + \beta \tag{1}$$

where $\theta$ is a $N \times d$ dimensional matrix whose first $d$ rows correspond to a $d \times d$ identity matrix and whose remaining $N-d$ rows specify the direction of the manifold. $\alpha_i$ is a $d$-dimensional vector representing the first $d$ coordinates of a point that lies on the manifold defined by the $\theta$ parameters and running through the origin of the coordinate space. $\beta$ is the offset vector.

Equation 1 can therefore be written as:

$$\begin{bmatrix} P_{i1} \\ P_{i2} \\ \vdots \\ P_{id} \\ P_{id+1} \\ \vdots \\ P_{iN} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ \theta_{d+1,1} & \theta_{d+1,2} & \theta_{d+1,3} & \dots & \theta_{d+1,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \theta_{N,1} & \theta_{N,2} & \theta_{N,3} & \dots & \theta_{N,d} \end{bmatrix} \begin{bmatrix} \alpha_{i1} \\ \alpha_{i2} \\ \vdots \\ \alpha_{id} \end{bmatrix} + \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_d \\ \beta_{d+1} \\ \vdots \\ \beta_N \end{bmatrix} \tag{2}$$

The entire model defined by the manifold and the individual learning models can therefore be characterized as having $(N-d)d+N$ parameters defining the manifold and shared across all the individual models, and $d$ "private" parameters for each model $i$, which can be used to generate the parameters $P_i$ of model $i$ using equation 2.

The manifold corresponds to the bias introduced in the hypothesis space of each model. The size of this space is reduced to an affine surface whose dimensionality $d \leq N$. The direction and offset parameters are updated using all the tasks.

Another source of bias can be introduced in the system: a proximity constraint can be imposed on the parameter vectors $P_i$ of the different models in order to reduce the area on the manifold in which the parameters $P_i$ are dispersed. This condition can further reduce the hypothesis space.

The number of parameters defining the manifold reaches a peak value of $(N-d/2)(d/2)$ when $N = d/2$, leading to a high capacity. But the "effective capacity" of the global model is reduced because the parameters of the manifold are updated using examples sampled from all the tasks. Also, the effective capacity is further reduced when a proximity constraint on the $P_i$ parameters is applied. Results presented in section 4 will illustrate these ideas.

## 3.2 Cost Function And Derivatives

Let $\{X_{ik}, f_i(X_{ik})\}, k = 1, ..., K$ be a set of $K$ examples corresponding to the input and desired values for task $i$. Assuming that $M$ tasks are being trained simultaneously, the cost function for the overall model can be divided into two terms, the first one computing the mean squared error between the desired outputs for each task $i$ and the outputs generated by the corresponding model $i$, and the second one representing the proximity constraint on the $P_i$ parameters by pushing them toward the point $\beta$:

$$\begin{aligned} E &= \frac{1}{M}\sum_{i=1}^{M}\frac{1}{K}\sum_{k=1}^{K}(f_i(X_{ik}) - Y_{ik})^2 + \lambda\sum_{i=1}^{M}\sum_{j=1}^{N}\left(\sum_{t=1}^{d}\theta_{jt}\alpha_{it}\right)^2 \\ &= \frac{1}{M}\sum_{i=1}^{M}\frac{1}{K}\sum_{k=1}^{K}E_{ik} + \lambda\sum_{i=1}^{M}\sum_{j=1}^{N}\left(\sum_{t=1}^{d}\theta_{jt}\alpha_{it}\right)^2 \end{aligned} \tag{3}$$

where $E_{ik}$ is the squared error corresponding to input example $X_{ik}$, $Y_{ik}$ is the output obtained when presenting example $X_{ik}$ to model $i$, and $\lambda$ is a constant specified by the user and used to weigh the proximity constraint. $\beta$ is therefore used to shift the manifold so that it is not obliged to run through the origin and it is also used as an attraction center towards which the generated parameters $P_i$ of each model should be approaching. Note that the second term in equation 3 can be viewed as performing a weight decay on the parameters defining the direction of the manifold and on the private parameters of each model.

Given the cost function defined in equation 3, the derivatives that can be used in a gradient descent method to update the values of the private and shared parameters are:

$$\frac{\partial E}{\partial \alpha_{il}} = \frac{1}{M}\frac{1}{K}\sum_{k=1}^{K}\left(\frac{\partial E_{ik}}{\partial P_{il}} + \sum_{j=d+1}^{N}\frac{\partial E_{ik}}{\partial P_{ij}}\theta_{jl}\right) + 2\lambda\sum_{j=1}^{N}\left(\sum_{t=1}^{d}\theta_{jt}\alpha_{it}\right)\theta_{jl},$$

$$\frac{\partial E}{\partial \theta_{jl}} = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{K}\sum_{k=1}^{K}\frac{\partial E_{ik}}{\partial P_{ij}}\alpha_{il} + 2\lambda\sum_{i=1}^{M}\left(\sum_{t=1}^{d}\theta_{jt}\alpha_{jt}\right)\alpha_{il} \qquad and \qquad \frac{\partial E}{\partial \beta_n} = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{K}\sum_{k=1}^{K}\frac{\partial E_{ik}}{\partial P_{in}}$$

where the subscripts on the left side of the equations are as follows: $i = 1, ..., M$, $j = d+1, ..., N$, $l = 1, ..., d$, and $n = 1, ..., N$.

## 4 Experiment

The model described in the previous section was trained to learn "invariant" boolean functions whose outputs depend only on the number of "1"s in the input, regardless of their positions. The input examples are defined in $\{0, 1\}^{14}$. Only those inputs containing 4 to 10 "1"s were considered. The remaining inputs were discarded because the number of available examples was too small for performance evaluation. 5000 examples were randomly sampled without replacement from the set of considered inputs. And all the 126 non-trivial (i.e. whose output isn't always 0 or 1 for all inputs) invariant boolean functions that can be defined on inputs containing 4 to 10 "1"s were generated.

This data was used to perform an analysis of variance to compare the performances of the new method with those obtained when no sharing and therefore no bias learning is performed. It was also used to compare the performances of the new model to those obtained when using the "family discovery" model [5] and the model for learning internal representations [1, 2]. The following setting was used: the 5000 examples were divided in 5 disjoint sets of 1000 examples each. Each set was divided into 500 examples used for testing generalisation performances and performing the analysis of variance, 200 examples for a validation set used for early stopping when training and for model selection, and 300 examples used for training.

Although there always were 300 training examples available, each experiment was repeated using 3 different sizes for the training set: 50, 100, and 300. Changing the size of the training set was used to test the performance of each method when the size of the training data varies. The size of the validation set wasn't modified in order to control the setting of the experiment.

30 functions were chosen without replacement from the set of 126 available functions. In the first part of the experiment, all 30 functions were trained separately. 23 different neural network architectures were considered for each function. Each architecture was trained using the 5 different data sets, and for each data set, the 3 different sizes for the training sets were used. The architectures contained 0 to 4 hidden layers, with varying number of units per hidden layer. This large number of architectures was considered in order to find a suitable architecture for each function. Note that the apparently optimal architecture can vary for different functions. At no time was the test set used. Only the validation set was used for early stopping and for model selection based on the proportion of classification errors (and on the mean squared error for equal proportions of classification errors).

The second part of the experiment consisted in allowing a knowledge transfer between the tasks. 2 different sets of experiments were performed: in the first one, the 30 available functions were divided in 10 disjoint combinations each containing 3 functions. In the second one, the 30 functions were divided in 5 disjoint combinations each containing 6 functions. The varying number of tasks in the combinations was considered in order to evaluate its effect on the sharing process.

The "internal representations" model was tested by training the neural networks in each combination using the 23 chosen architectures. For each architecture, $L$ experiments were performed which consisted in sharing the first $l$ layers among the neural networks in a combination, with $l = 1, ..., L$, and $L =$ number of hidden and output layers in the chosen architecture. This process was repeated for each combination, each data set and each size for the training set.

The "family discovery" model was tested by selecting for each combination, each data set and each size for the training set, the neural network architecture that led to the smallest validation set mean classification error on the tasks in the combination when those tasks were trained separately. Given the chosen architecture, $M - 2$ experiments were performed, each corresponding to using the first $m$ top eigenvectors obtained when applying a PCA to the parameters of the chosen neural networks that were trained separately, with $m = 1, ..., M - 2$, and $M =$ number of tasks in the combination. Note that when $m = M - 1$, all the eigenvectors obtained in the PCA are used to define a manifold. In that case, the $M$ points representing the parameters of the $M$ tasks in the combination are lying on the corresponding manifold and no knowledge transfer is performed. Also for each experiment, 5 different "weighing" constants were used to weigh the

Table 1: P-values obtained when comparing the mean generalisation error $e_{m_1}$ when a knowledge transfer method is used to the corresponding error $e_{m_2}$ when the tasks in a combination are trained separately. The null hypothesis is $H_0 : e = e_{m_1} - e_{m_2} \geq 0$ and the alternate hypothesis is $H_1 : e < 0$. The p-values of the best knowledge transfer method(s) are indicated in boldface (the choice of the best method was based on the results in this table and table 2 assuming a 5% significance level).

| | | New Method | | | Internal Representations | | | Family Discovery | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Training set size | | | Training set size | | | Training set size | | |
| | | 50 | 100 | 300 | 50 | 100 | 300 | 50 | 100 | 300 |
| Combination size (M) | 3 | **0.0307** | **0.0015** | **0.0123** | 0.4473 | **0.0004** | **0.0123** | 0.3552 | 0.0161 | 0.0846 |
| | 6 | 0.1576 | **0.0006** | **0.0183** | 0.1353 | **0.0003** | **0.0183** | 0.1275 | 0.0709 | **0.0293** |

Table 2: P-values obtained when comparing pairs of knowledge transfer methods $m_1$ and $m_2$. The null hypothesis is: $H_0 : e = e_{m_1} - e_{m_2} = 0$ and the alternate hypothesis is: $H_1 : e \neq 0$, where $e_{m_1}$ and $e_{m_2}$ respectively represent the mean generalisation error when using methods $m_1$ and $m_2$.

| | | New method vs Internal representations | | | New Method vs Family Discovery | | | Family Discovery vs Internal representations | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Training set size | | | Training set size | | | Training set size | | |
| | | 50 | 100 | 300 | 50 | 100 | 300 | 50 | 100 | 300 |
| Combination size (M) | 3 | 0.0041 | 0.7363 | 1 | 0.0056 | 0.0091 | 0.0027 | 0.8654 | 0.0019 | 0.0027 |
| | 6 | 0.7311 | 0.1455 | 1 | 0.4997 | 0.0020 | 0.2136 | 0.3595 | 0.0034 | 0.2136 |

constraint of closeness of the parameters of each neural network to the manifold. Those constants were used in the variant of the EM algorithm used to further improve the generalisation performances of the tasks in a combination by iteratively updating the manifold's direction and the values of the parameters of each task.

The new method described in section 3 was tested by choosing 8 neural network architectures among the 23 available architectures for training sets of size 50, 10 architectures for training sets of size 100, and one architecture for training sets of size 300. For each architecture, experiments were performed using different values of manifold dimensionality (which ranged from 1 to $N$ with steps of 10) and 4 different values for $\lambda$ (0, 0.01, 0.1 and 1.0). The number of chosen architectures was limited given that for each architecture, a large number of experiments was performed which corresponded to using different values for the manifold dimensionality.

An analysis of variance was performed to compare each knowledge transfer method with the method that consists in separately training each task in a combination and to compare the knowledge transfer methods between themselves. The p-values computed to test whether applying a knowledge transfer between groups of tasks outperforms learning them separately are presented in Table 1. And the p-values obtained when comparing the knowledge transfer methods between themselves are presented in Table 2.

In table 1, the new method and the internal representations learning method have identical p-values for training sets of 300 examples. This is due to the fact that both methods led to zero classification errors on the validation and test sets for all combinations and all data sets. The family discovery method didn't manage to achieve a zero error rate in all cases. The mean generalisation error computed over all the combinations and data sets when the tasks in each combination were trained separately was 5.15%.

For training sets of size 50, the new method outperformed the internal representations learning method in 66.67% of all available combination/data set pairs. For training sets of size 100, this value dropped to 46.67% with a 2.67% tie (i.e. both methods led to the same generalisation performances). When comparing the new method to the family discovery method, the corresponding rates were 62.67% and 82.87% (with a 1.33% tie) for training sets of size respectively equal to 50 and 100.

When evaluating each knowledge transfer method, we observed some interesting phenomena: for the internal representations learning method, in most cases, the optimal architectures were those that had several hidden layers with a "small" number of units per layer, thus allowing to control the number of shared parameters. Also, in some cases, the optimal sharing consisted in sharing all the layers of the networks including the output layers. In the family discovery method, we noted that the optimal number of eigenvectors when considering combinations of size 6 was much smaller than the number of relevant eigenvectors obtained in the PCA (i.e. the eigenvectors having large corresponding eigenvalues). This observation might support the notion that the manifold that best fits the parameters of the tasks in a combination doesn't necessarily correspond to the manifold that allows to obtain the best generalisation performances. Another problem in this method that considerably slowed down and in many cases hindered the learning process is the fact that the optimal parameters for each network in a combination obtained in a step in the iterative process are used to initialize the same networks in the next step: they can be very large, thus saturating several hidden and output units. For the new method, we noticed that the optimal manifold dimensionalities were in most cases smaller than $0.3N$ or larger than $0.7N$ where $N$ corresponds to the parameter space dimensionality. And the optimal values for $\lambda$ were 0.1 and 1. A further analysis of the results showed that large $\lambda$ values were useful when the dimensionality of the manifold was close to $\frac{N}{2}$. Large $\lambda$ values can control the capacity of the learning models. A more detailed analysis of all three methods can be found in [3].

## 5 Future Work And Conclusion

The cost function considered in equation 3 consists in trying to optimize the mean error computed over all the models in a combination. A variation of this function that could lead to better performances would be to separately measure the error of each combination and stop modifying the parameters of a model when its error reaches an apparently optimal value, while pursuing the optimization of the remaining models. We expect such a function to outperform the one defined in equation 3 when the degree of similarity between the tasks varies.

Another important subject that must be tackled is to define the meaning of "similarity". When are two tasks similar? What kind of similarity can be helpful to bias the hypothesis space? What happens when the similarity between the tasks varies? A possible solution to the last question could be to extend the model presented in this paper and to "learn" a mixture of manifolds that would bias in different ways different groups of tasks.

The model presented in this paper is being tested on real-world problems, in particular on financial data and object recognition problems.

A model for biasing the hypothesis space of a learner was presented. It can be used when several related tasks are available, and consists in re-parameterizing the parameters of the learner of each task as a function of an affine manifold defined in parameter space and a point lying on the manifold. The direction and offset of the manifold are "learned" using examples sampled from all the tasks. Results of an analysis of variance for a problem were presented that show significantly improved performances when using this model.

**Acknowledgments** We would like to thank Claude Nadeau for his help with the specification of the model used for the analysis of variance.

## References

[1] J. Baxter. Learning model bias. In M. Mozer, D. Touretzky, and M. Perrone, editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1996.

[2] R. Caruana. Algorithms and applications for multitask learning. In *The 13th International Conference on Machine Learning*, Bari, Italy, 1996.

[3] J. Ghosn and Y. Bengio. Bias learning, knoweldge sharing. Technical report. In preparation.

[4] J. Ghosn and Y. Bengio. Multi-task learning for stock selection. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 946–952. MIT Press, 1997.

[5] S. M. Omohundro. Family discovery. In M. Mozer, D. Touretzky, and M. Perrone, editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1996.

[6] L. Y. Pratt. *Artificial Neural Networks for Speech and Vision*, chapter Non-literal transfer Among Neural Network Learners, pages 143–169. Chapman and Hall, 1993.

[7] S. Thrun and J. O'Sullivan. Discovering structure in multiple learning tasks: The tc algorithm. In *International Conference on Machine Learning*. Morgan Kaufmann, 1996.