Machine Learning- Paper Review

Student Id: 1001830083 Name: Saachi Shivhare

1. DeepPink: reproducible feature selection in deep neural networks

Yang Young Lu Department of Genome Sciences, University of Washington Seattle, WA 98195

Deep Learning is a class of ML algorithm which uses multiple layers to progressively extract higher-level features from the raw input. The adjective "deep" in deep learning comes from the use of multiple layers in the network. The networks have at least two layers of perceptrons, one for the input layer and one for the output. Sandwich one or more “hidden” layers between the input and the output and you get a “deep” neural network; the greater the number of hidden layers, the deeper the network. Due to the complicated nature of deep neural networks (DNNs) and other deep learning methods, they have been mostly treated as black box tools. Interpretability of scientific results is becoming increasingly important, as researchers aim to understand why and how a machine learning system makes a certain decision. Identifying explainable features that contribute the most to DNN predictions has received much attention.

Approach:

Existing methods either fit a simple model in the local region around the input [33, 38] or locally perturb the input to see how the prediction changes [35, 6, 34, 37]. Though these methods can yield insightful interpretations, they focus on specific architectures of DNNs and can be difficult to generalize. The fragility of these widely used methods and demonstrated that even small random perturbation can dramatically change the feature importance.

It would be highly problematic if the choice of highlighted region varied dramatically in the presence of very small amounts of noise. In such a setting, it is desirable for practitioners to select explanatory features in a fashion that is robust and reproducible, even in the presence of noise.

Practitioners select explanatory features in a fashion that is robust and reproducible, even in the presence of noise.

FDR: The false discovery rate (FDR) can be exploited to measure the performance of feature selection algorithms. Informally, the FDR is the expected proportion of falsely selected features among all selected features, where a false discovery is a feature that is selected but is not truly relevant. Larger value of s implies that the constructed knockoff features are more different from the original features and thus can increase the power of the method.

$$\text{FDR} = \mathbb{E}[\text{FDP}] \text{ with FDP} = \frac{|\widehat{S} \cap \mathcal{S}_0^c|}{|\widehat{S}|}.$$

$\mathcal{S}_0$ Subset of features responsible for model output

$\mathcal{S}_0^c$ Complement of $\mathcal{S}_0$

$\widehat{S}$ features selected by a feature selection procedure

Knockoff Filter: To achieve FDR control, Model-X Knockoffs frameworks are used. Knockoff features are computed using the below formula.

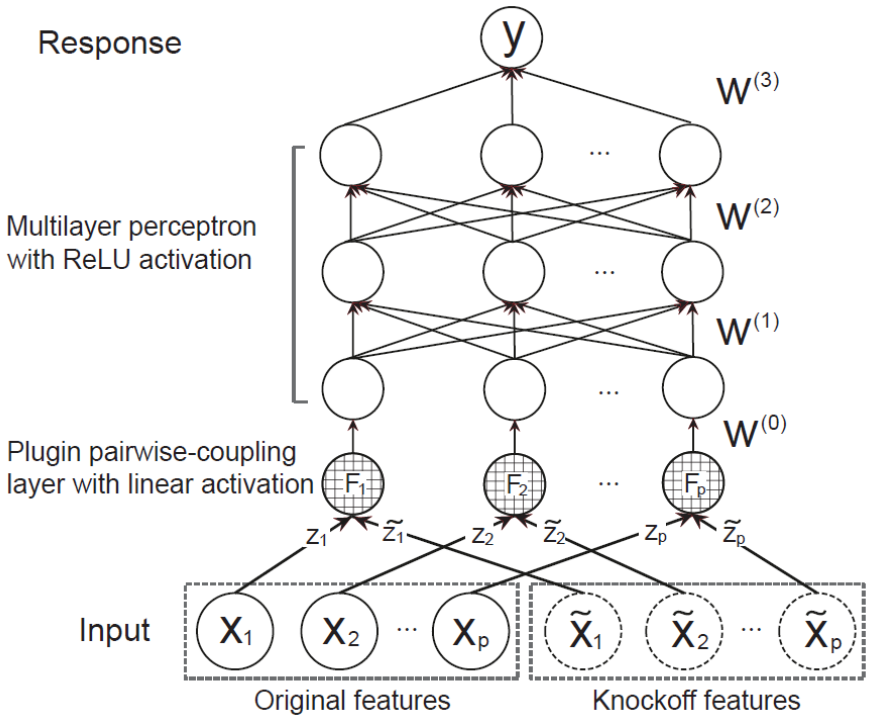
$$\tilde{\mathbf{x}}|\mathbf{x} \sim N\left(\mathbf{x} - \text{diag}\{\mathbf{s}\}\boldsymbol{\Sigma}^{-1}\mathbf{x}, 2\text{diag}\{\mathbf{s}\} - \text{diag}\{\mathbf{s}\}\boldsymbol{\Sigma}^{-1}\text{diag}\{\mathbf{s}\}\right)$$

Knockoff filter achieves FDR control in two steps:

1. Construction of knockoff features:
   Definition 1 ([10]). Model-X knockoff features for the family of random features $\mathbf{x} = (X_1, \cdots, X_p)^T$ are a new family of random features $\tilde{\mathbf{x}} = (\tilde{X}_1, \cdots, \tilde{X}_p)^T$ that satisfies two properties: (1) $(\mathbf{x}, \tilde{\mathbf{x}})_{\text{swap}(\mathcal{S})} \overset{d}{=} (\mathbf{x}, \tilde{\mathbf{x}})$ for any subset $\mathcal{S} \subset \{1, \cdots, p\}$, where swap$(\mathcal{S})$ means swapping X_j and $\tilde{X}_j$ for each $j \in \mathcal{S}$ and $\overset{d}{=}$ denotes equal in distribution, and (2) $\tilde{\mathbf{x}} \perp\!\!\!\perp Y|\mathbf{x}$, i.e., $\tilde{\mathbf{x}}$ is independent of response Y given feature $\mathbf{x}$.
2. Filtering using knockoff statistics: For every jth feature and its knockoff features counterpart we calculate the knockoff statistics Where Zj and Zj complement represent feature importance measures. A desirable property for knockoff statistics Wj ’s is that important features are expected to have large positive values, whereas unimportant ones should have small magnitudes symmetric around 0.

   $$W_j = g_j(Z_j, \tilde{Z}_j) \text{ for } 1 \leq j \leq p.$$

   Deep Pink Framework:
   In MLP by naively feeding knockoff features directly to the network, FDR is controlled but the output power was extremely low. This framework resolves the power issue. The network is fed through a plugin pairwise coupling layer containing p filters. F1, … Fp where the jth filter connects original feature and its knockoff counterpart.



- Initially, both Zj and its knock of filter weight are initialised equally.
- During training these filter weights compete with each other.
- How to decide if a particular feature is important? If a feature's Zj's magnitude is much larger than its knock off counter part
- To encourage competition between a feature and its knockoff counterpart, linear activation **function is used in the pair wise coupling layer.**
- **The outputs of the filters are then fed into a fully connected MLP to learn a mapping to the response Y.**
- **Each layer learns** a mapping from its input to a hidden space, and the last layer learns a mapping directly from the hidden space to the response Y.

Applications to HIV1 data:

- Compared to Knockoff, DeepPINK obtains equivalent or better power in 9 out of 13 cases with comparable false discovery proportion.
- DeepPINK and BHq are the only methods that can identify mutations in DDI, TDF, and X3TC for NRTI.
- Compared to BHq, DeepPINK obtains equivalent or better power in 10 out of 13 cases with much better controlled false discovery proportion
- DeepPINK shows remarkable performance for APV where it recovers 18 mutations without any mutation falling outside the TSM list.

Conclusion:

- In this paper a new method DeepPink is introduced which helps in interpreting a DNN model by identifying a subset of relevant input features subject to the FDR control.
- It uses pairwise coupling plugin to encourage competition with each original feature and its knockoff counterpart.
- It achieves FDR with higher power compared to the naïve combination of the knockoffs.
- One drawback is that the feature importance measures heavily depend on the specific algorithm used to fit the model.

2. Self-Taught Learning: Transfer Learning from Unlabeled data.

Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, Andrew Y. Ng, Computer Science Department, Stanford University, CA 94305 USA.

In this paper, new machine learning "self-taught learning" framework for using unlabeled data in supervised classification task has been presented. An approach to self-taught learning that uses sparse coding to construct higher-level features using the unlabeled data has also been described.

Introduction:

Labeled data for machine learning is often very difficult and expensive to obtain, and thus the ability to use unlabeled data holds significant promise in terms of vastly expanding the applicability of learning methods. It is difficult to envision a process for collecting such unlabeled images, that does not immediately also provide the class labels. This makes the classification task quite hard with existing algorithms for using labeled and unlabeled data, including most semi-supervised learning algorithms.

In this large number of unlabeled images, audio samples or text documents randomly downloaded from the internet have been used to improve performance. Many randomly downloaded images will contain basic visual patterns (such as edges) used to recognize such patterns from the unlabeled data.

Approach:

Approach to self-taught learning consists of two stages:

1. Learn a representation using only unlabeled data.
2. Apply this representation to the labeled data and use it for the classification task.

   Once the representation has been learned in the first stage, it can then be applied to different classification tasks.

Self-taught learning that uses sparse coding to construct higher-level features using the labeled data. Labeled training set of m samples $\{(x_1^{(1)}, y_1^{(1)}), (x_1^{(2)}, y_1^{(2)}), \ldots, (x_1^{(m)}, y_1^{(m)})\}$ drawn from some distribution D. Each labeled x_1 is an input feature vector y^i. Also, a set of unlabeled samples is also given.

In transfer learning, the unlabeled and labeled data should not be completely irrelevant to each other if unlabeled data is going to help in the classification task.

Results:

In this, a self-taught learning algorithm outputs a hypothesis h: ^{n} -> {1,.....,C} this mimics the input-label relationship represented by the labeled training data; this hypothesis h is the tested under the same distribution D from which the labeled data was drawn. Using the Fisher Kernel derived from the generative model described, obtain a classifier customized specifically to the distribution of sparse coding algorithms.

3. Why significant variables aren't automatically good predictors.

Adeline Loa, Herman Chernoffb,1, Tian Zhengc, and Shaw-Hwa Loc,1 a Department of Political Science, University of California, San Diego, La Jolla, CA 92093.

Gives a brief explanation and some statistical insights on why higher significance cannot automatically imply stronger predictivity. This paper also demonstrate that highly predictive variables do not necessarily appear as highly significant, thus evading the researcher using significance-based methods. What makes variables good for prediction versus significance depends on different properties of the underlying distributions has also been discussed. If prediction is the goal, then significance must be laid aside as the only selection standard.

Approach:
Significance has played a larger role in statistical inference whereas prediction has served more in identifying future data behavior. The retooling of significance has found a role in data dimension reduction for prediction. We evaluate this retooling and consider how significance and predictivity are related in the goal of good prediction. Highly significant uses assumption, but no knowledge of exact distributions whereas highly predictive uses knowledge of both fh and fd.

Prediction rate is given by below formula. Here x represents the possibly multivariate observation that can assume a finite number of values; fD and fH are its probability distributions.

$$\text{prediction rate} = 0.5 \sum_x \max(f_D(x),\ f_H(x)).$$

In the examples predictive variable sets and significant variable sets are provided with some graphs. These significance tests are compared with the I score. I score and its median are based on the data. I score t is very well correlated with the truth for modest sample sizes; at large sample sizes I is still better correlated with truth than are the training prediction rate and χ2 test. The taller and lighter the bar, the more significant the VS.

$$I = \sum_{j=1}^{m_1} \frac{n_j}{n} \frac{\left(\overline{Y}_j - \overline{Y}\right)^2}{s^2/n_j} = \frac{\sum_{j=1}^{m_1} n_j^2 \left(\overline{Y}_j - \overline{Y}\right)^2}{\sum_{i=1}^{n} \left(Y_i - \overline{Y}\right)^2},$$

Results:

It is difficult to analyze real world examples because of large number of variables. To analyze we need to know the underlying distribution.

4. Efficient Sparse Coding Algorithm.

Honglak Lee, Computer Science Department Stanford University, Stanford, CA 94305

Sparse coding provides a class of algorithms for finding succinct representations of stimuli; given only unlabeled input data, it discovers basis functions that capture higher-level features in the data. However, finding sparse codes remains a very difficult computational problem.
In this paper, efficient sparse coding algorithms that are based on iteratively solving two convex optimization problems are presented.

1. L1-regularized least squares problem.
2. L2-constrained least squares problem.

In sparse coding number of bases obtained is greater than the input dimension unlike other unsupervised learning techniques like PCA.

Approach:

The goal of sparse coding is to represent input vectors approximately as a weighted linear combination of a small number of (unknown) “basis vectors.” The basis set can be overcomplete (n > k) and can thus capture a large number of patterns in the input data. Also, the reconstruction error must be minimized.

The use of either L1 penalty or epsilonL1 penalty as the sparsity function, the optimization problem is convex in B (while holding S fixed) and convex in S (while holding B fixed),2 but not convex in both simultaneously. In this paper, we iteratively optimize the above objective by alternatingly optimizing with respect to B (bases) and S (coefficients) while holding the other fixed.

For learning the bases B, the optimization problem is a least squares problem with quadratic constraints. There are several approaches to solving this problem, such as generic convex optimization solvers (e.g., QCQP solver) as well as gradient descent using iterative projections.

For learning the coefficients S, the optimization problem is equivalent to a regularized least squares problem.

Feature sign search algorithm maintains an active set of potentially nonzero coefficients and all other coefficients must be zero and searches for the optimal active set and coefficient signs. It proceeds in a series of "feature-sign steps".

In every step, it is given a current guess for the active sets and the signs, computes the analytical solution, then updates the solution, the active set and the signs using an efficient discrete line search between the current solution and x.

Results:

In this paper, author has formulated sparse coding as a combination of two convex optimization problems and presented efficient algorithms for each: the feature-sign search for solving the L1-least squares problem to learn coefficients, and a Lagrange dual method for the L2-constrained least squares problem to learn the bases for any sparsity penalty function.

These algorithms have been tested on a variety of datasets and gives significantly better performance compared to previous methods.

Author also justified the explanation of neuron's end stopping and surround suppression phenomenon using sparse coding.

5. Learning Social Networks from Web Documents Using Support Vector Classifiers

In this paper, author has proposed an approach to generate a social network from a collection of web documents. Every person can be presented by using their corresponding documents.

Assumption here is that the social network is partially explored which is a training dataset. To extract the missing relations to complete the network, the support vector classifier is used.

Social networks are represented either by graphs or matrices or adjacency matrix.

Approach:

Actor Modeling and relationship modeling are used.

There are two assumptions while learning a social network from incomplete network.

1. A subset of relations are represented by adjacency matrix.
2. The textual data associated to the actors.

There are three steps involved in this. First being modeling the actor in the social network. Second, modeling the relations between the actors and third is to train the classifier to learn social network.

Actor Modeling: In this, each actor is represented by their web documents. All the documents associated with an individual are merged together to build a unique document vector and each document is associated to one actor.

The global weights are calculated using two methods local weighting and global weighting technique.

Relationship Modeling: To model the relationship between two actors is to estimate the similarity of their document vector. The similarity measure offers very poor results because it models each relation with only one variable. The other approach is to aggregate the documents vector of the actors in both sides of the relation and create new aggregated document vector. Taking an example, let d_i and d_j be the document vectors associated to actor a_i and a_j. The relation between two actors are modeled by aggregating their vectors by an operator such as MIN, MAX or Product.

Imbalance Social Network data. To deal with class imbalance is to artificially re-balance the training data. Upscale the minority class or down scale the majority class.

$$S = 1 - \frac{2r}{n(n-1)}$$

n: # of actors
r: # of relations

Performance Evaluation Measures are done using F score as micro averaged and macro averaged F-measure. This is used for calculating the performance of classifier of both classes.

Results:

As the percentage of majority class increases, recall drops linearly while precision remains almost constant.

Macro-averaged F-measure was used to evaluate the extracted social network instead of micro averaged F measure as it gave a better result.

Document vector aggregation model is proposed instead of document similarity.

6. A deep learning approach to unsupervised ensemble learning.

Uri Shaham, Xiuyuan Cheng, Omer Dror, Ariel Jaffe, Boaz Nadler, Joseph Chang, Yuval Kluger

In this paper, author has demonstrated that deep learning methods can be applied in the context of crowdsourcing and unsupervised ensemble learning and provide state-of-the-art results.

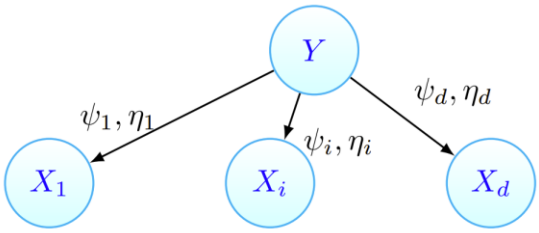
First, we prove that the popular model of Dawid and Skene, which assumes that all classifiers are conditionally independent, is equivalent to a Restricted Boltzmann Machine (RBM) with a single hidden node.

The DS model relied on two key assumptions:

1. classifiers make perfectly independent errors.
2. these errors are uniformly distributed across all instances.

To address the second issue above, several authors proposed richer models.

Approach:
A deep learning approach to unsupervised ensemble problems with possibly dependent classifiers, where the conditional independence assumption is strongly violated.



The DS model has an equivalent parametrization in terms of a Restricted Boltzmann Machine (RBM) with a single hidden node. Hence, under this model, the posterior probability of the true labels can be estimated from a trained RBM.

RBM based Deep Neural Net (DNN) can be applied to unsupervised ensemble learning and propose a heuristic for determining the DNN architecture. Construct an RBM based DNN (stacked RBMs). DNN is used to perform the ensemble learning. In this learning process the first step is to train RBM with d hidden units and then compute the singular value decomposition of the weight matrix W, and determine its rank.

The rank is m<=d, we re-train the RBM setting the number of hidden units to m. If m > 1, add another layer on top of current layer and proceed recursively. The process stops when m = 1, so last layer of DNN contains single node.

Results:

Compared with Votes: Majority voting, which is the maximum likelihood prediction, assuming that all classifiers are conditionally independent and have the same accuracy.

Compared with DS: Approximate maximum likelihood predictions under the Dawid and Skene model. Specifically, we use Spectral Meta Learner (Parisi et al., 2014), and Restricted Likelihood (Jaffe et al., 2014).

Compared with CUBAM: The method of Welinder et al. (2010), which assumes conditional independence, but allows the accuracy of each classifier to vary across different regions of the input domain.

7. Convolutional neural networks for graphs

Mathias Niepert, Mohamed Ahmed, Konstantin Kutzkov NEC Labs Europe, Heidelberg, Germany.

In this paper, a framework for learning convolutional neural networks for arbitrary graphs has been proposed. These graphs may be undirected, directed, and with both discrete and continuous node and edge attributes.

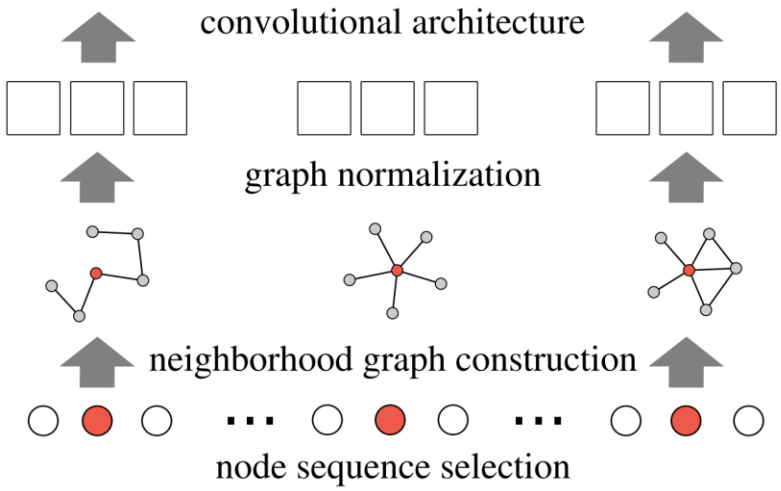
Approach:

The aim is to bring convolutional neural networks to bear on a large class of graph-based learning problems.

There are two problems:

1. Given a collection of graphs, learn a function that can be used for classification and regression problems on unseen graphs. The nodes of any two graphs are not necessarily in correspondence.
2. Given a large graph, learn graph representations that can be used to infer unseen graph properties such as node types and missing edges.

   The proposed approach builds on concepts from convolutional neural networks (CNNs) for images and extends them to arbitrary graphs. The neighborhood graphs serve as the receptive fields to read feature values from the pixel nodes.
   In these instances, one has to solve two problems: (i) Determining the node sequences for which neighborhood graphs are created and (ii) computing a normalization of neighborhood graphs, that is, a unique mapping from a graph representation into a vector space representation.
   The proposed approach, termed PATCHY-SAN, addresses these two problems for arbitrary graphs. For each input graph, it first determines nodes (and their order) for which neighborhood graphs are created. For each of these nodes, a neighborhood consisting of exactly k nodes is extracted and normalized, that is, it is uniquely mapped to a space with a fixed linear order.



PATCHY-SAN architecture has several advantages over existing approaches.
PATCHY-SAN is highly efficient, naively parallelizable, and applicable to large graphs.
It supports feature visualizations providing insights into the structural properties of graphs.
It learns application dependent features without the need to feature engineering.

Results:
The CNNs accuracy is highly competitive with existing graph kernels. A receptive field size of 10 results in the best classification accuracy.
Similar to the experience on image and text data, we expect PATCHY-SAN to perform even better for large data sets. The performance advantage to be much more pronounced for data sets with many graphs.

8. Support Vector Machine Active Learning for Image Retrieval

Simon Tong, Department of Computer Science Stanford University Sanford, CA 94305

In this paper, author has demonstrated the use of a support vector machine active learning algorithm for doing relevance feedback for image retrieval.

Relevance feedback is often a critical component when designing image databases. With these databases it is di cult to specify queries directly and explicitly. It determines a user's desired output or query concept by asking the user whether certain proposed images are relevant or not.
The SVM active learning algorithm selects the most informative images to query a user and quickly learns a boundary that separates the images that satisfy the user's query concept from the rest of the dataset.

Approach:

1. SVM_{active}

i. An SVM captures the query concept by separating the relevant images from the irrelevant images with a hyperplane in a projected space, usually a very high dimensional one. The projected points on one side of the hyperplane are considered relevant to the query concept and the rest irrelevant.

ii. It learns the classifier quickly via active learning. The active part selects the most informative instances with which to train the SVM classifier. This step ensures fast convergence to the query concept in a small number of feedback rounds.

iii. Once the classifier is trained, it returns the top-k most relevant images. These are the k images farthest from the hyperplane on the query concept side.

2. Active Learning:
   It is assumed that the instances x are independently and identically distributed according to some underlying distribution F (x) and the labels are distributed according to some conditional distribution P (y | x). For an unlabeled pool U, an active learner ` has three components: (f; q; X). The component is a classier, f : X ! f1; 1g, trained on the current set of labeled data X (and possibly unlabeled instances in U too). The second component q(X) is the querying function that, given a current labeled set X, decides which instance in U to query next. The active learner can return a classier f after each pool-query (online learning) or after some number of pool-queries.

   The main difference between an active learner and a regular passive learner is the querying component q. One approach to reduce the version space is to choose a pool query that halves the version space.

Results and Conclusion:

1. Active learning with SVM can provide a powerful tool for searching image databases, outperforming several traditional query refinement schemes.
2. Running time increases linearly with the size of the image database both for the relevance feedback phase and for the retrieval of the top k images.
3. Another approach for finding single relevant image is to use another algorithm to seed SVM_{active}. For example, the MEGA algorithm.

9. Supervised Dictionary Learning

In this paper, supervised dictionary learning generative framework using sparse models has been introduced.

Sparse decompositions are used with predefined dictionaries for face and signal recognition. Dictionaries are learned for a reconstruction task, and the corresponding sparse models are used as features in an SVM. In a discriminative method is introduced

for various classification tasks, learning one dictionary per class; the classification process itself is based on the corresponding reconstruction error, and does not exploit the actual decomposition coefficients.

$$\mathcal{R}^\star(\boldsymbol{x}, \mathbf{D}) = \min_{\boldsymbol{\alpha} \in \mathbb{R}^k} ||\boldsymbol{x} - \mathbf{D}\boldsymbol{\alpha}||_2^2 + \lambda_1 ||\boldsymbol{\alpha}||_1.$$

In classical sparse coding tasks, one considers a signal x in R n and a fixed dictionary D = [d1, . . . , dk] in R n×k (allowing k > n, making the dictionary overcomplete).
It is well known in the statistics, optimization, and compressed sensing communities that the ℓ1 penalty yields a sparse solution, very few non-zero coefficients in α.

Approach:

Linear Model:

$$f(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\theta}) = \mathbf{w}^T \boldsymbol{\alpha} + \hat{b}, \text{ where } \boldsymbol{\theta} = \{\mathbf{w} \in \mathbb{R}^k, b \in \mathbb{R}\} \text{ parametrizes the model.}$$

Bilinear Model:

$$f(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\theta}) = \boldsymbol{x}^T \mathbf{W} \boldsymbol{\alpha} + b, \text{ where } \boldsymbol{\theta} = \{\mathbf{W} \in \mathbb{R}^{n \times k}, b \in \mathbb{R}\}.$$

The number of parameters in (ii) is greater than in (i), which allows for richer models. Note that one can interpret W as a linear filter encoding the input signal x into a model for the coefficients α, which has a role similar to the encoder in [18] but for a discriminative task.

Input: n (signal dimensions); $(\boldsymbol{x}_i, y_i)_{i=1}^m$ (training signals); k (size of the dictionary); $\lambda_0, \lambda_1, \lambda_2$ (parameters); $0 \leq \mu_1 \leq \mu_2 \leq \ldots \leq \mu_m \leq 1$ (increasing sequence).
Output: $\mathbf{D} \in \mathbb{R}^{n \times k}$ (dictionary); $\boldsymbol{\theta}$ (parameters).
Initialization: Set $\mathbf{D}$ to a random Gaussian matrix with normalized columns. Set $\boldsymbol{\theta}$ to zero.
Loop: For $\mu = \mu_1, \ldots, \mu_m$,
Loop: Repeat until convergence (or a fixed number of iterations),

- *Supervised sparse coding:* Solve, for all $i = 1, \ldots, m$,

$$\begin{cases} \boldsymbol{\alpha}_{i,-}^\star = \arg\min_{\boldsymbol{\alpha}} \mathcal{S}(\boldsymbol{\alpha}, \boldsymbol{x}_i, \mathbf{D}, \boldsymbol{\theta}, -1) \\ \boldsymbol{\alpha}_{i,+}^\star = \arg\min_{\boldsymbol{\alpha}} \mathcal{S}(\boldsymbol{\alpha}, \boldsymbol{x}_i, \mathbf{D}, \boldsymbol{\theta}, +1) \end{cases}. \tag{10}$$

- *Dictionary and parameters update:* Solve

$$\min_{\mathbf{D}, \boldsymbol{\theta}} \Big(\sum_{i=1}^m \mu \mathcal{C}\big((\mathcal{S}(\boldsymbol{\alpha}_{i,-}^\star, \boldsymbol{x}_i, \mathbf{D}, \boldsymbol{\theta}, -y_i) - \mathcal{S}(\boldsymbol{\alpha}_{i,+}^\star, \boldsymbol{x}_j, \mathbf{D}, \boldsymbol{\theta}, y_i))\big) + (1-\mu)\mathcal{S}(\boldsymbol{\alpha}_{i,y_i}^\star, \boldsymbol{x}_i, \mathbf{D}, \boldsymbol{\theta}, y_i) + \lambda_2 ||\boldsymbol{\theta}||_2^2 \Big) \text{ s.t. } \forall j, ||\mathbf{d}_j||_2 \leq 1. \tag{11}$$

Results:

1. A discriminative approach to supervised dictionary learning that effectively exploits the corresponding sparse signal decompositions in image classification tasks, and have proposed an effective method for learning a shared dictionary and multiple (linear or bilinear) models.
2. Future work will be devoted to adapting the proposed framework to shift-invariant models that are standard in image processing tasks, but not readily generalized to the sparse dictionary learning setting.
3. Investigation extensions to unsupervised and semi-supervised learning and applications to natural image classification.

10. Scalable fast rank -1 dictionary learning for fmri big data analysis .

In this paper a novel distributed rank-1 dictionary learning model has been designed.

The basic idea is the observed functional signals are the results of the linear combination from the signals of many latent sources including noises.

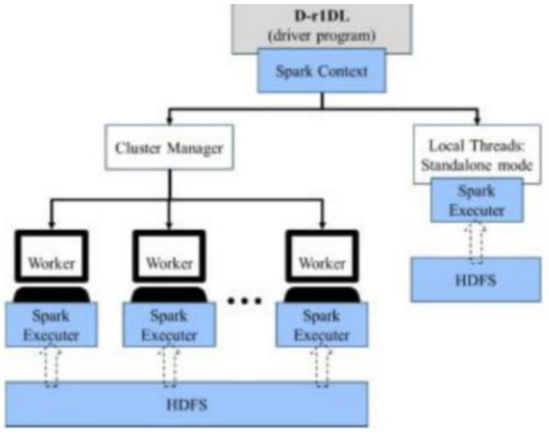
Approach:

The learning process is a fix-point algorithm by altering least square updates. Most of the routines in the algorithm will only take one vector as input and one vector as output.

The method then aim to identify the latent sources signals as well as the loading matrix.

The decomposition results consists of two parts:

1. Temporal pattern of the functional networks – regarded as basis activation patterns.
2. Spatial patterns of the functional networks.
3. Matrix vector multiplication: each node will use all the updated v vector then estimate its corresponding portion of the u vector.
4. The model was implemented and parallelized in Spark.



minimizing the following energy function $L(u, v)$:

$$L(u,v) = \|S - uv^T\|_F, \text{s.t.} \|u\| = 1, \|v\|_0 \le r. \quad (1)$$

$$v = \underset{v}{\operatorname{argmin}} \|S - uv^T\|_F, s.t. \|v\|_0 \le r,$$

$$u = \underset{u}{\operatorname{argmin}} \|S - uv^T\|_F = \frac{Sv}{\|Sv\|}, \quad (2)$$

Converging at step j if: $\|u^{j+1} - u^j\| < \varepsilon, \varepsilon = 0.01$.

$$R^n = R^{n-1} - v^T R^{n-1}, R^0 = S, 1 < n \le K, \quad (3)$$

Results:

The final goal is to provide an integrated solution for functional neuroimaging big data management and analysis.