Scientific Data Analysis using Neural Networks

Lecture Notes by David Horn

Tel Aviv University, July 2019

1. Introduction
2. Muti-Layer-Perceptron
3. Preprocessing with PCA or SVD. Feature Filtering.
4. The Bias-Variance Problem, and averaging over networks.
5. Support Vector Machines
6. Support vector Clustering
7. Novel Formulation of Parzen Data Analysis
8. Weight Shape Decomposition (WSD) and Quantum Clustering (QC)
9. Shape Analysis of Images
10. From Shallow to Deep Networks
11. The DL Paradigm and its Supremacy
12. Examples of Applications
13. ML of ML Architectures

Chapter 1. Introduction: The discplinarity gap

The Big Data buzz of recent years has led to public interest in data analysis and to a flood of books and articles in the professional literature as well as in the general media. As a physicist, I was brought up – and later I preached the same dogma – on the understanding that we test the world around us through experimental observations, and we try to extract from them their essential features, and thus formulate the laws that govern natural phenomena. This tradition, which may be traced back to Galileo Galilei, is the foundation of modern science. Hence I find myself surprised from time to time when people talk about the new field of Data Science: haven't we done it all along in our scientific quests for centuries?

The common answer is that never before have we had to analyze such vast amounts of data, and never before have we had the computational capabilities to do so. I wish to add another important aspect putting it into the perspective of disciplinarity. The importance of large scale data analytics has entered into the commercial domain and into social sciences and humanistic studies. They have turned to experts in computer science and statistics to help them make sense of these new challenges. This has led to the birth of data scientists, data engineers and data managers, in different regimes of science and technology.

Having had multidisciplinary experience in neural computation and in bioinformatics, and collaborating with colleagues and students from other disciplines, I have learned that disciplinarity leads to a deep-rooted bias in the process of analysis and understanding of data. This may sound odd to the general public, but the truth is that tapping different experts one may get different answers as to what to do, and what is interesting, based on the disciplines they were trained in. Hence the new task force of data experts should be familiar with different approaches and different disciplinary attitudes leading to the search paths that they choose.

This collection of lectures, which were originally aimed toward an audience of physicists, tries to cover some gaps among disciplines which I have encountered during my studies with many students and collaborators. One example of a question which often comes up in conversations is "what is it that a neural network learns"? The technical details are well explained by Machine Learning, a field which becomes a pillar of data science. But the question is still an ongoing source for research in this field. In chapter 2 we discuss a point of view presented in 1987, when neural computation was making its first steps as a sub-discipline. Since this period lies in the past, well beyond the birth of deep networks in 2012, it is hardly mentioned in modern texts.

This example may explain the effort underlying this set of lecture notes. It should be viewed as complimentary to standard modern texts, some of which will be quoted in the different chapters. It is not intended to serve as an exhaustive coverage of the newly developing data science transdiscipline[1].

Warning and Apologies.

This set of lectures started as a course for practicing High Energy physicists. You may find it biased toward specific topics which I have been personally involved with. I have not made efforts to present an extensive list of references, and therefore many important contributions to neural networks and deep learning are missing. I took liberty of using many texts and figures from existing publications. This set of lectures should be regarded as an informal collection of insights rather than a novel textbook. I hope it will be valuable to scientists from different fields who strive to understand and use neural networks for data analysis.

Footnote 1. The concept of transdisciplinarity has developed in humanities and social sciences. A recent philosophical and historical summary is provided by [Osborne 2015], where it is described how and why practical studies require the collaboration across different disciplines.

Reference

Peter Osborne, 2015. Problematizing Disciplinarity, Transdisciplinary Problematics. Theory, Culture & Society 2015, Vol. 32(5–6) 3–35.
https://doi.org/10.1177/0263276415592245

Chapter 2. Multi-Layer-Perceptron: The Feed Forward Neural Network: how does it learn? Lessons from the 80s.

A neural network is defined in terms of layers of simple computational elements (dubbed as **neurons**, a.k.a. **perceptrons**) in an architecture which is exemplified by Fig. 1. Our discussion follows [Hertz, 1991]. This structure, which is also known as a multi-layer perceptron (MLP) contains an input vector ξ, which is 5-dimensional in this case, a 2-dimensional output vector **O,** and a hidden layer whose three outputs form the vector **V.** In addition, there exist many weights denoted by w and W, which are crucial in transferring the outputs of one layer to the inputs of the following one. In this description the conventional bias, or threshold, is absent. It can be retrieved by replacing ξ_1 by -1 and identifying its connecting weights with thresholds.

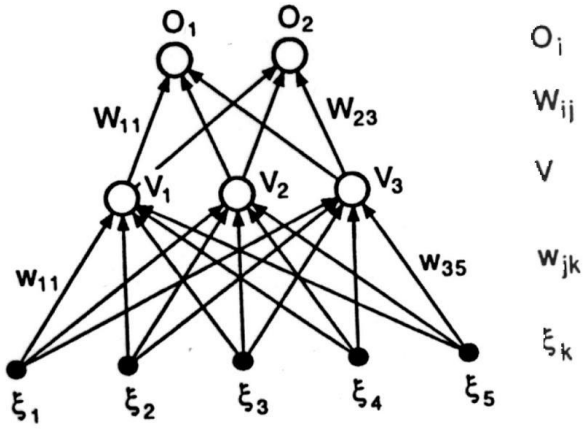


Fig. 1. A multi-layer perceptron (MLP). Reproduced from Hertz (1991).

V is calculated in terms of ξ as

$$V_j = g(\sum_k w_{jk}\, \xi_k) \qquad (1)$$

where g is a non-linear function, such as the sigmoid $g(x) = (1 + e^{-x/\sigma})^{-1}$, applied to the linear sum of inputs multiplied by the weights. The sigmoid has a strong non-linear behavior near x=0, depending on the parameter σ, while for asymptotic x it turns into a binary decision function.

If one considers a set of N different input vectors $\boldsymbol{\xi}^{\mu}$ the network will produce a set of N different output vectors $\mathbf{O}^{\mu}$ related through all weights:

$$O_i^{\mu} = g\left(\sum_j W_{ij}\ V_j^{\mu}\right) = g\left(\sum_j W_{ij}\ g(\sum_k w_{jk}\, \xi_k^{\mu})\right) \qquad (2)$$

The basic idea of constructing a MLP as a learning tool, is to use a set of N examples $\boldsymbol{\xi}^{\mu}$ for which the outputs $\boldsymbol{\theta}^{\mu}$are well-known, and change the sets of all weights **w** (meaning both w and W) so that the desired output equals as much as possible to the output that the network calculates. For this purpose, one defines the Least Mean Squares (LMS) function

$$F[\mathbf{w}] = \sum_{\mu}(\boldsymbol{\theta}^{\mu} - \boldsymbol{O}^{\mu})^2 \qquad (3)$$

and reduces it to a minimum through a procedure known as back-propagation [Rumelhart 1986], moving consecutively from the upper to the lower layer of the network, changing the weights through gradient descent.

This procedure of learning from examples is the basic tool of supervised learning. Running time and again over all N examples one modifies the set **w** until the lowest values of *F* is reached. To get a feeling of what this implies, let us consider the case of a simple rule, like addition, applied to integers represented by binary inputs, leading to integers represented by binary outputs. With integers smaller than 2^n one can represent the problem as a network with 2n binary inputs and n+1 binary outputs. The questions which arise are will such a network learn the addition rule, and, if it does, how many examples does it have to be trained on until it generalizes for all test cases, i.e. until it performs addition correctly on all examples on which it has not been trained.

Let us first consider the question if the network can cope with such a problem. For each one of the 2n binary inputs (of the two numbers which should be added) it has to assign one specific output. This amounts to choosing one out of 2^{n+1} possible Boolean rules. The number of available parameters, i.e. elements of **w**, in the setup of a MLP with an intermediate layer of size *m* is *m(4n+1)* which, for large n, lags far behind the number of possible Boolean rules, unless *m* is allowed to grow considerably. Thus a network is limited by what it can do, just as a single perceptron cannot cope with the XOR problem[1] [Minsky and Papert 1969]. Nonetheless chances are that the addition rule is doable by a network with moderate *m*. In a related problem, where the layered MLP was replaced by a set of interconnected logical gates, [Paternello and Carnavali 1987] have shown that only a small subset of all 2^{2n} examples was needed to train such a network to perform addition correctly.

For a network to perform such a task there exist two conditions. First that the *architecture* of the network enables the accommodation of the task. The second is that there should be *many configurations of* $\boldsymbol{w}$ which enable it. In terms of gradient descent dynamics one needs minima of $F[\boldsymbol{w}]$ which satisfy $F[\boldsymbol{w}]=0$, providing a strict adherence to the rule which we want the network to learn. Only if there are many such minima in $\boldsymbol{w}$ space, one can hope that the training procedure will easily converge into one of them, rather than keeping to search for a suitable minimum until it exhausts all possible examples.

For a large enough hidden layer, a neural network can approximate any function to a desired accuracy, as was proved by [Hornik 1989] and [Cybenko 1989]. Nonetheless, for practical purposes it is useful to have additional hidden layers. Thus [Siu and Roychodhury 1993] have shown that multiplication of integers can be optimally represented by a MLP with two hidden layers.

As you may recall, the inability of the perceptron to solve the XOR problem hindered the advancement of neural networks. The backpropagation algorithm, as applied to MLP, has led to a blooming of neural networks since the mid-1980s. But only recently, the increase in computational power, and the realization of the versatility of novel configurations, have led to the surge of Deep Neural Networks (DNN) in research and in commercial applications.

Footnote 1.

The single perceptron, e.g. the neuron V_2 in Figure 1, is limited by what $V_2^{\mu} = g(\sum_k w_{2k}\, \xi_k^{\mu})$ can represent. Since the input of g is a linear function in ξ space, g becomes a decision hyperplane in this space such that it leads to two different binary values for all inputs which

lie, with distances larger than σ, below or above it. This property of the single neuron is known as **linear separability**. Thus the XOR function, corresponding to the following Boolean truth table, cannot be implemented in the two-dimensional vector space.

input	output
(0,0)	0
(0,1)	1
(1,0)	1
(1,1)	0

The XOR function

References

Hertz, J., Krogh, A., Palmer, R. G. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwood City, CA, 1991.

K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators", *Neural Networks* 2, 359 (1989).

Cybenko, G. Approximation by superposition of a sigmoidal function. *Math. of Control, Signals, and Systems* 2 (1989), 303-314.

Minsky, M. L., Papert, S. A. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, Cambridge, MA, 1969 (expanded edition 1988).

Rumelhart, D. E., Hinton, G. E., Williams, R. J. "Learning internal representations by error propagation". In "Rumelhart, D. E., McClelland, J. L., et al. (eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1. The MIT Press, Cambridge, MA, 1986." pp. 318-362

Rumelhart, D., Hinton, G. & Williams, R. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).

Siu, K.-Y., Roychowdhury, V. Optimal depth neural networks for multiplication and related problems. In: *Advances in Neural Information Processing Systems 5* (ed. S. J. Hanson, J. D. Cowan, C. L. Giles). Morgan Kaufmann, San Mateo, CA, 1993. Pp. 59-64.

Chapter 3. Preprocessing with PCA or SVD. Feature Filtering.

Given a set of N data points in high dimensions $\mathbf{x}_i \in \mathbb{R}^d$ one is faced with various problems such as easy visualization and noise reduction, which can be solved by performing a linear transformation to a new set of coordinates which is determined by characteristics of the data. This can be achieved by **Principal Component Analysis** (PCA), applied either to the correlation matrix $\boldsymbol{Cor} = \sum_{ij} \mathbf{x}_i \, \mathbf{x}_j$ or the covariance matrix $\boldsymbol{Cov} = \sum_{ij} (\mathbf{x}_i - \boldsymbol{\mu}) \, (\mathbf{x}_j - \boldsymbol{\mu})$ where $\boldsymbol{\mu} = \frac{1}{N} \sum_i \mathbf{x}_i$ is the average of all data points. These symmetric matrices can be diagonalized by unitary transformations whose eigenvalues have the meaning of contribution to the variance of the data, conventionally ordered from largest to smallest. The eigenvectors are referred to as Principal Components (PCs). Data visualization is then allowed by selecting a reduced set of PCs, which also serves the role of noise reduction.

When the data is presented by an *nxm* matrix $\boldsymbol{M}$, whose *n* rows specify *n* instances (data-points) and *m* columns specify the features of all instances, we are no longer dealing with a Euclidean space. Different features may have completely different meanings, but a reduction of these features by linear combinations into some low dimensional correlation space may still be very desirable[1]. This can be achieved by the **Singular Value Decomposition** (SVD)

$$\boldsymbol{M} = USV^T$$

where both *U* and *V* are unitary matrices (whose relevant dimensions are dictated by *n* and *m* respectively) and where *S* (with dimensions *nxm*) has positive non-vanishing elements only on its diagonal, conventionally ordered from the largest to the smallest eigenvalue. It follows then that

$$\boldsymbol{M}\boldsymbol{M}^T = USS^TU^T \quad \text{and} \quad \boldsymbol{M}^T\boldsymbol{M} = VS^TSV^T$$

i.e., SVD leads to two different PCA realizations, applied to the correlation matrices of instances and of features correspondingly, sharing the same set of eigenvalues.

The SVD procedure is presented diagrammatically in Fig. 1. In this example, where m=9 features are involved, we present only the first 9 columns of *U* and the first 9 rows of *S*. The other rows of *S* contain only zeros; hence other columns of *U* do not matter in this calculation.

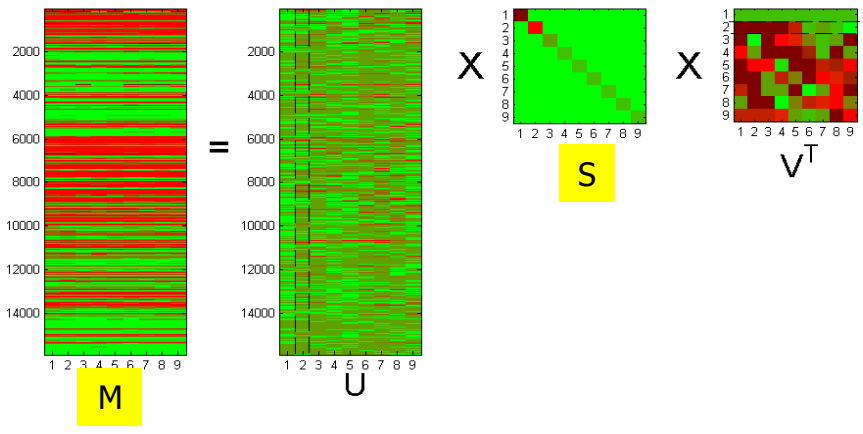


Fig.1 Illustration of SVD, in which the matrices U and S are trimmed to show only 9 columns and 9 rows correspondingly. Since all other rows of S contain only zeros, all the non-shown entries do not contribute to the illustrated calculation. Following [Varshavsky 2007]

Dimensional reduction can be achieved by approximating $\boldsymbol{M}$ with $\boldsymbol{X} = US^rV^T$ where S^r is a reduced version of S keeping only the first r leading eigenvalues. This turns out to be the best approximation of $\boldsymbol{M}$ by an r-dimensional matrix in the LMS sense, i.e. it minimizes the sum of squared differences of all $\boldsymbol{M}$ and $\boldsymbol{X}$ entries.

As a first example of the use of dimensional reduction let us look at a very recent result from the UK biobank, displayed in Fig. 2. It demonstrates how very large data can be simply ordered through their PCs which follow from an ancestral background analysis. The authors [Bycroft 2018] note that "the first two principal components separate out individuals with sub-Saharan African ancestry, Europe ancestry and east Asian ancestry. Individuals who self-report as mixed ethnicity tend to fall on a continuum between their constituent groups. Further principal components capture population structure at subcontinental geographic scales"

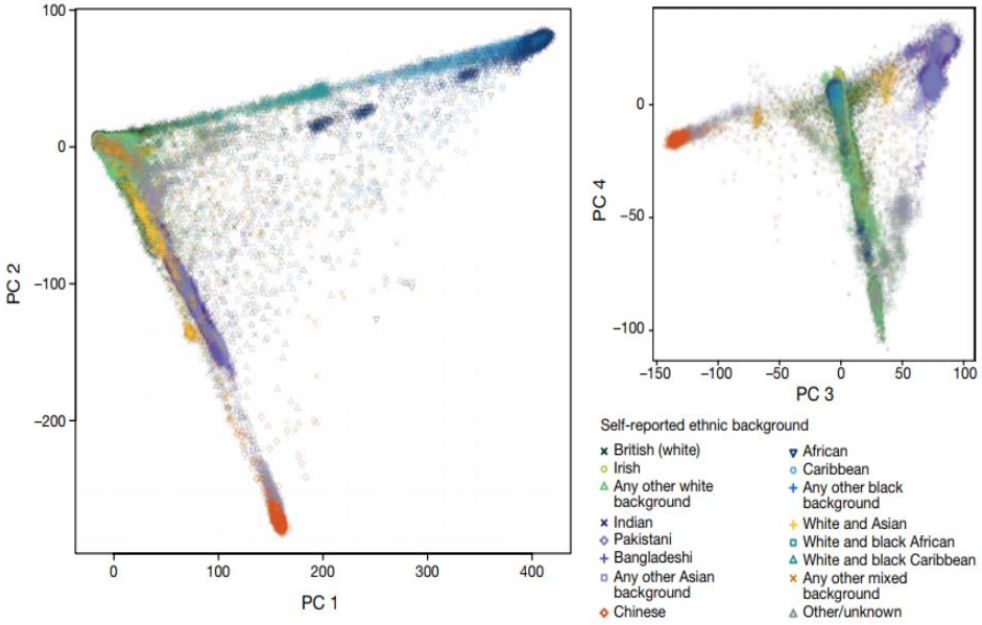


Fig. 2. Taken from the UK Biobank analysis of [Bycroft 2018]. Each point represents a UK Biobank participant (n=488,377 samples) and is placed according to their principal component (PC) scores in each of the top four principal components. Colours and shapes indicate the self-reported ethnic background of each individual.

As a second illustration of dimensional reduction we look at results of applying SVD to a gray image, whose pixel entries can be regarded as a data matrix $\boldsymbol{M}$. In the example of Fig. 3a, we have a 200x320 pixel representation, thus $\boldsymbol{M}$ has rank 200. Approximating it with 10 eigenvalues one gets a recognizable image. With 50 eigenvalues the match seems perfect (see https://math.byu.edu/~schow/clown_svd.htm).

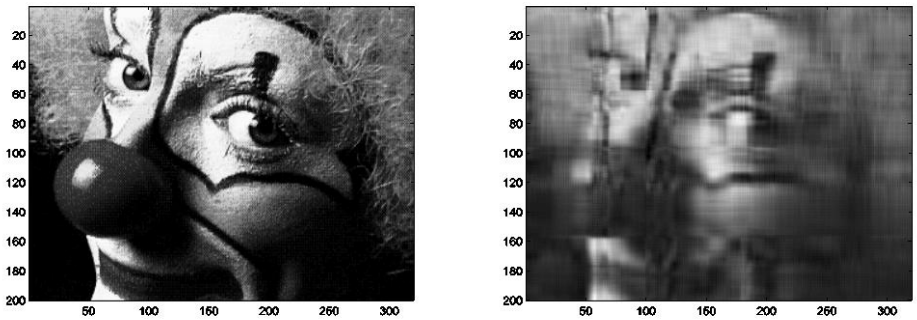


Fig. 3 A. Image leads to a matrix with rank=200. B. Image reconstructed with a reduced matrix having just the 10 leading eigenvalues. Source: https://math.byu.edu/~schow/clown_svd.htm

Feature Filtering

In the example of Fig. 3, all entries of the matrix have the same meaning, i.e. gray scale image values. This is not the case for general data matrices. In order to have a measure of the different importance of different features let us introduce the concept of SVD-Entropy [Alter 2000, Wall 2003] defined in term of all eigenvalues S_k appearing on the diagonal of S

$$E = -\frac{1}{\log N}\sum_{k=1}^{N} P_k \log P_k \qquad \text{where} \qquad P_k = S_k^2 \ / \sum_l S_l^2.$$

This allows us to introduce the concept of CE, defining the contribution of some feature *j* (out of the *m* features) by comparing $E(M)$ with $E(M')$, the entropy of an *nx(m*-1) matrix which lacks this feature *j*: $CE(j) = E(M) - E(M')$. The results lead to an understanding which features should be kept for an SVD analysis in the problem at hand. This is exemplified in Fig. 4, reproduced from [Varshavsky 2006]. The problem analyzed here is clustering of cells extracted from 72 Leukemia patients, classified into four groups, whose measured features are expression values of 7129 genes [Golub 1999]. Evaluating CEs of the features, they are presented here in a ranked order. One clearly observes three general groups of features: high CE values, which turn out to be the most relevant contributions for our discussion, many neutral CEs and a few negative CEs which are due to noisy features. [Varshavsky 2006] have demonstrated that clustering the data on the basis of only 100-200 leading CE features, leads to much better results than employing all features.

The reasoning behind this selection is the following: SVD entropy is low when one or a few features are dominant. In this case it suffices to use these features to label the data and no further SVD decomposition is needed. On the other hand, large SVD entropy indicates that retaining the relevant features we expect the instances (data points) to be more evenly spread in the truncated SVD space. It is then expected that clustering algorithms will be useful to catch the non-linear grouping which exists in such data. Hence the general recommendation is to apply feature filtering before reducing dimensionality through SVD.

For other methods of feature selection see, e.g., [Guyon and Elisseeff, 2003].

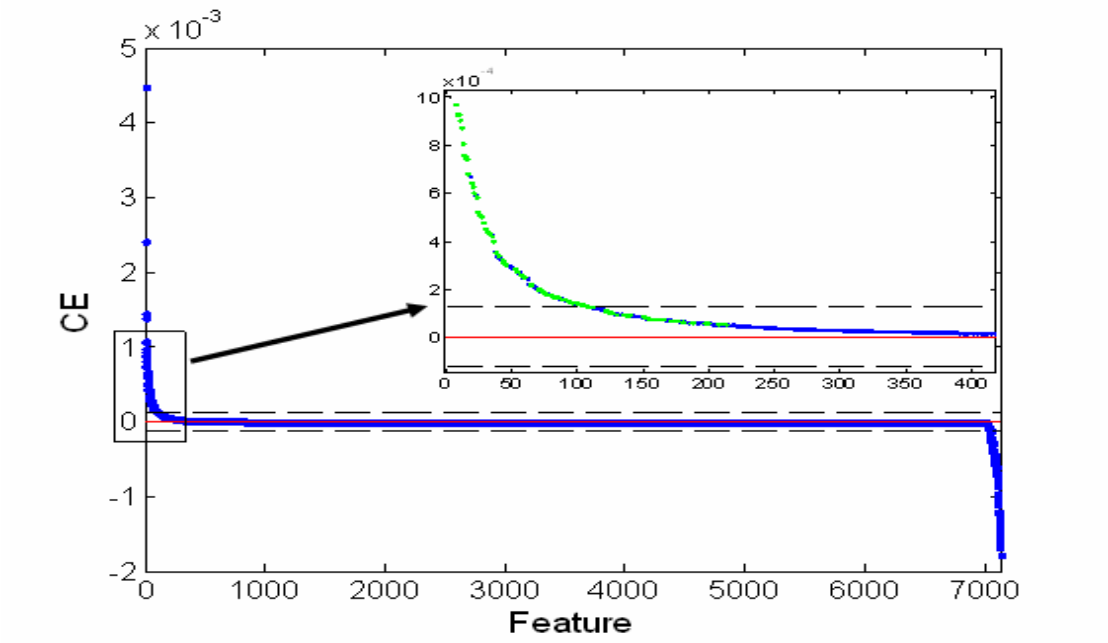


Fig. 4. Unsupervised Feature Filtering (UFF) evaluates CE for each feature. Results [Varshavsky 2006] apply to the data set of [Golub 1999]. The green points in the inset correspond to an alternative feature filtering which proceeds with gradual removal of the highest feature when selecting the next one.

References

Alter, O., Brown, P.O. and Botstein, D. (2000) Singular value decomposition for genome-wide expression data processing and modeling, *PNAS*, 97, 10101-10106.

Wall, M., Rechtsteiner, A. and Rocha, L. (2003) Singular Value Decomposition and Principal Component Analysis. In Berrar, D., Dubitzky, W. and Granzow, M. (eds), *A Practical Approach to Microarray Data Analysis*. Kluwer, 91–109.

Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A., Bloomfield, C.D. and Lander, E.S. (1999) *Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring*, Science, 286, 531-537.

Guyon, I. and Elisseeff, A. (2003) An Introduction to Variable and Feature Selection, *Journal of Machine Learning Research*, 3, 1157--1182.

Roy Varshavsky, Assaf Gottlieb, Michal Linial and David Horn. "Novel Unsupervised Feature Filtering of Biological Data" *Bioinformatics* 2006, 22(14):e507-513.

Roy Varshavsky, PhD thesis 2007.

Clare Bycroft et al. 2018. The UK Biobank resource with deep phenotyping and genomic data. *Nature* 562, 203-209.

Footnote 1: When the different entries have very different numerical ranges, one should normalize them to similar scales so that dissimilarities will not adversely affect the resulting analysis

Chapter 4: The Bias-Variance Problem, and averaging over networks.

One of the things you learn when you deal with data is that you have to deal with errors, handle them correctly, and accept their existence as part of the game you play. A physics student learns in the lab to distinguish between systematic and statistic errors in the measurements which s/he performs. The analogs of these concepts in machine learning were introduced by [Geman 1992] as bias and variance, where the first refers to the computational tools one employs and the second to inherent properties of the data. The mathematical definition which they have suggested, can be written in the language of statistical inference as

$$E_D[(f_D(x) - E[y|x])^2] = (E_D[f_D(x)] - E[y|x])^2 + E_D[(f_D(x) - E_D[f_D(x)])^2]$$

Here E_D represents expectation with respect to all possible training sets D. $f_D(x)$ is the function which is the desired output of data x within the set D, with $E_D[f_D(x)]$ representing the "ground-truth" which should be derived from all data-sets D. We try to estimate it by using a computational statistical tool (e.g. a neural network), predicting the distribution of output y given input x: y|x. The first term on the right-hand side is the bias of our estimator, and the second is the variance of the data. Together they contribute to the error of our result.

A bias/variance tradeoff appears in training neural networks for many epochs. Conventionally one divides data sets into training sets and test sets. As training of the network continues for increasing number of epochs, the training error decreases, but the generalization error, as measured on the test set, reaches a minimum and then increases. The decrease of the training error corresponds to decrease of bias, but the increase of the generalization error reflects an increase due to variance. Stopping at the minimum of the generalization error strikes a balance between the two. In the absence of a declared test set, the developer should reserve some fraction of the training set to serve as a cross-validation set, to extract from it when the desired balance is reached.

For one given data set, one may apply the same formalism to the set of all initial conditions of the neural network, using as predictor the average outputs of these sets. One may then argue [Naftaly 1997] that this average reduces the variance of the predictor. This was applied to a study of the time series of sunspot activities. This popular time series served as a challenge to neural networks initiated by [Weigend 1990]. The purpose was predicting the lowest average relative variance (ARV) of the test set

$$ARV_{train,test} = \frac{\sum_{t \in train,test}(y_t - p_t)^2}{\sum_{t \in train}(y_t - E\{y\})^2}$$

where y_t is the yearly sunspot activity, and p_t is the result of the predictor at time t. The denominator is the observed variance of the data in the training set. The training set contained the period between 1701 and 1920, for which one minimizes ARV_{train}, and the test set contained the measurements during the years 1921 to 1955 on which one tests the prediction ARV_{test}. The yearly sunspot activity is demonstrated in Fig. 1.

[Naftaly 1997] employed a set of Recurrent Neural Networks (RNNs), with an architecture of 12 inputs, a hidden layer of 4 sigmoidal neurons, and one output. The 12 inputs contained the measured sunspot activities at years t-1, t-2, t-3 , t-4 , t-9 and t-10, and also the activities of the four neurons of the hidden layer at time t-1. Employing Q networks, the predictor was defined as the average of all Q outputs.

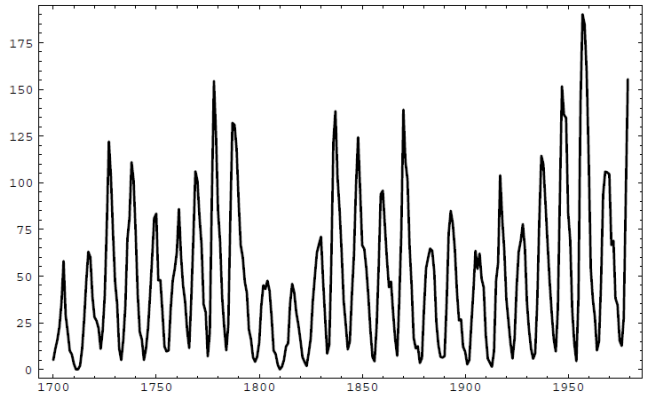


Fig. 1. The yearly sunspot activity recorded between 1701 and 1955. Reproduced from [Naftaly 1997].

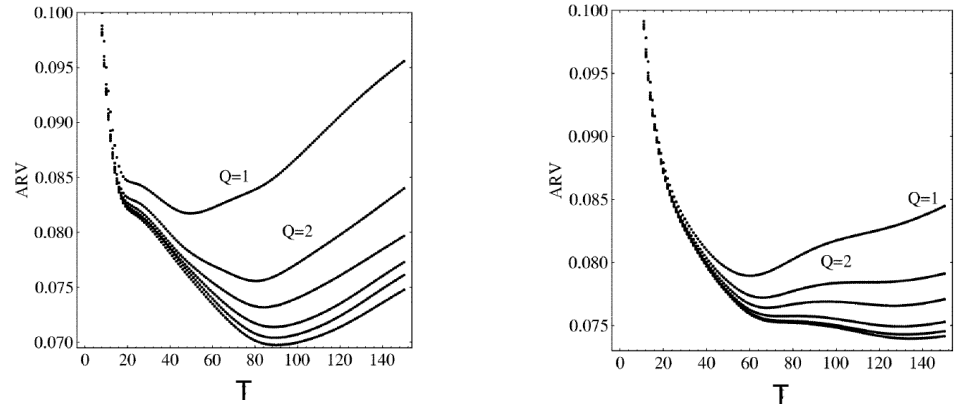


Fig.2. Plots of the predictors of Q networks for (left) the training set and (right) the test set *vs* T, the number of training periods. The curves are shown for different choices of group sizes: Q = 1*;* 2*;* 4*;* 10*;* 20 from top to bottom. The lowest curve is the extrapolation to $Q \rightarrow \infty$. Reproduced from [Naftaly 1997].

The results are demonstrated in Fig. 2 for both the training and the test set. As expected one sees a decrease of both errors as function of the number of training periods t, due to decrease of the bias, as expected. But then one observes increase due to variance. The latter is however almost completely wiped out for large Q values. Thus one can continue to large epochs, further decreasing the bias and decreasing the ARV since the variance due to initial conditions has been eliminated. Note that one finds in this example an increase of variance for low Q values also in the training set, which happens because the networks employed here have recurrent elements in the hidden layer.

References

Geman S, Bienenstock E and Doursat R. 1992 Neural networks and the bias/variance dilemma *Neural Comput.* **4** 1–58

Naftaly U, Intrator N and Horn D. 1997. Optimal ensemble averaging of neural networks. *Network: Comput. Neural Syst.* **8** 283–296.

Weigend A S, Huberman B A and Rumelhart D 1990. Predicting the future: a connectionist approach *Int. J. Neural Syst.* **1** 193–209

Chapter 5. Support Vector Machines.

The perceptron, or single mathematical neuron, is a device which implements linear separation of data into two groups. In Fig. 1 we demonstrate this situation where the separating plane is defined in terms of its perpendicular vector **w** (defined by the perceptron's weights) and having bias b. Clearly many separating planes can be defined, but the one chosen here, together with the two end points associated with the blue group and the one edge point of the green group, is such that the margin separating these two groups is the largest, m=2/|**w**|.

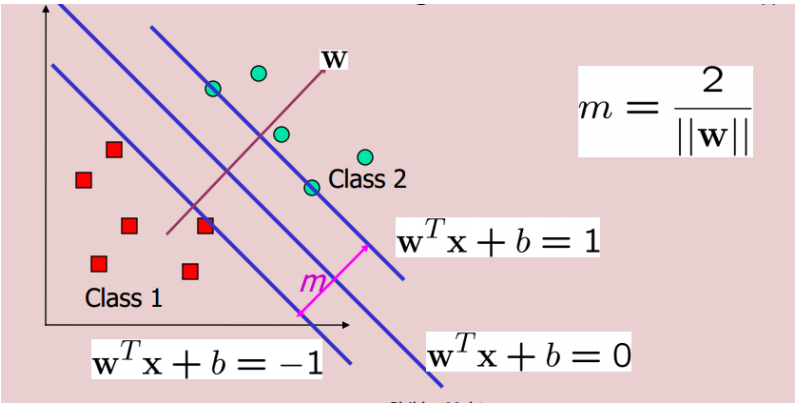


Fig. 1. Marginal decision boundaries in linear SVM. The widest margin between two separable classes of data points is obtained by the configuration exemplified here. Reproduced from datascience.stackexchange.com.

Obviously a perceptron cannot solve a non-linear separation task, such as presented in Fig. 2. This can however be provided by a Support Vector Machine (SVM) using the kernel trick [Boser, Guyon, Vapnik 1992]. This formalism [Vapnik 1995] has led to many applications in the following decade, allowing for the development of Machine Learning (ML) into a new direction.

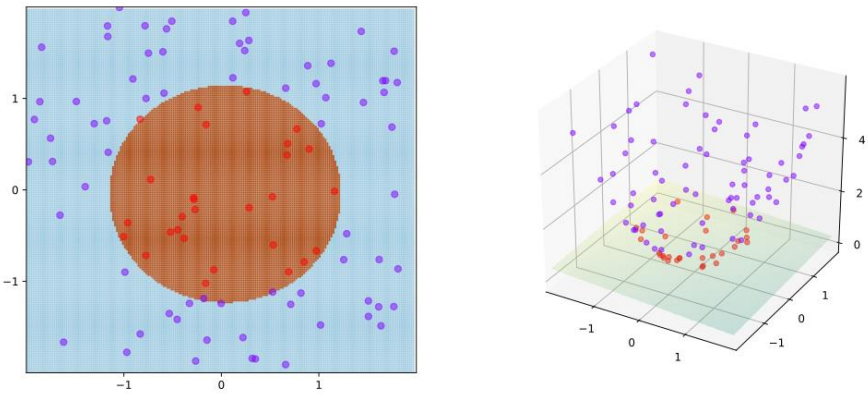


Fig. 2. A non-linear separation task (left), can be solved by an SVM employing a quadratic kernel projecting the data onto a 2-dim manifold in a 3-dim space. Reproduced from Wikipedia. Origin: By Shiyu Ji - Own work, CC B4.0, https://commons.wikimedia.org/w/index.php?curid=60458994

A linear-SVM is identical to the perceptron. Using the configuration drawn in Fig. 1, also known as a hard-margin implementing linear separability, SVM refers to the extreme data points obeying $|\boldsymbol{w}\cdot\ \boldsymbol{x}_i\ + b| = 1$ as support-vectors, since they support the two border hyper-planes of the different sets of data points. In the hard-margin problem all data-points can be categorized by $y_i\ (\boldsymbol{w}\cdot\ \boldsymbol{x}_i\ + b) \geq 1$ with $y_i\ = -1$ for points in class 1 and 1 for points in class 2 (in Fig. 1). Equality (rather than inequality) holds only for SVs. Expressing these conditions with Lagrange multipliers α_i one can seek the maximal margin through minimization of the Lagrangian

$$L = \frac{1}{2}\,||\boldsymbol{w}||^2 - \sum_{i=1}^{m} \alpha_i\,(\,y_i\,(\boldsymbol{w}\cdot\ \boldsymbol{x}_i\ + b) - 1) \qquad (1)$$

Taking the derivatives of L with respect to b and to **w** one finds that the minima should obey

$$\sum_{i=1}^{m} \alpha_i\ y_i\ = 0 \quad \text{and} \quad \boldsymbol{w} = \sum_{i=1}^{m} \alpha_i\ y_i\ \boldsymbol{x}_i\,. \qquad (2)$$

To solve for the support vectors one inserts these equalities back into L and searches for the points for which the dual Lagrangian *W*

$$W = \sum_{i=1}^{m} \alpha_i\ - \frac{1}{2} \sum_{i=1,j=1}^{m} \alpha_i\ \alpha_j\ y_i\ y_j\ \left(\boldsymbol{x}_i\ \cdot\ \boldsymbol{x}_j\right) \qquad (3)$$

reaches its maximum. All support vectors obey $\alpha_i\ > 0$ while for all other points $\alpha_i\ = 0$.

The kernel trick allows one to move from data space where all points $\boldsymbol{x}_i$ located to a Hilbert space reached by a feature map $\Phi(\boldsymbol{x}_i)$ where distances can be expressed in terms of kernels:

$$||\Phi(\boldsymbol{x}_1)\text{-}\ \Phi(\boldsymbol{x}_2)||^2 = k(\boldsymbol{x}_1\,,\boldsymbol{x}_1) + \ k(\boldsymbol{x}_2\,,\boldsymbol{x}_2) - 2k(\boldsymbol{x}_1\,,\boldsymbol{x}_2) \qquad (4)$$

Polynomial, sigmoidal and Gaussian maps allow for such kernel representations. Thus, for a Gaussian one may choose $k(\boldsymbol{x}_1\,,\boldsymbol{x}_2) = \exp(-q(\boldsymbol{x}_1\ -\ \boldsymbol{x}_2\ )^2)$. Thus solving an SVM after mapping to such a Hilbert space, amounts to replacing the dot product of data points in Eq. (3) by their kernel function

$$W = \sum_{i=1}^{m} \alpha_i\ - \frac{1}{2} \sum_{i=1,j=1}^{m} \alpha_i\ \alpha_j\ y_i\ y_j\ k\left(\boldsymbol{x}_i\,,\boldsymbol{x}_j\right). \qquad (5)$$

Thus the problem described in Fig. 2 may be resolved by projecting the 2-dim plane onto a 3-dim feature space, such as

$$z_1 = x_1^2 \qquad z_2 = x_2^2 \qquad z_3 = \sqrt{2}\, x_1\, x_2$$

where the indices refer to axis labels. This performs a projection of the 2-dim $\boldsymbol{x}$ space onto a 2-dim manifold in the 3-dim $\boldsymbol{z}$ space. A 2-dim plane in $\boldsymbol{z}$ can cut the manifold along a circle providing the desired separation of the two classes.

A **soft margin** can be introduced when the data cannot be strictly separated into two classes. This amounts to replacing the boundary conditions with $y_i\,(\boldsymbol{w}\cdot\,\boldsymbol{x}_i\, + b) \geq 1 - \xi_i$ where $\xi_i \geq 0,$ while adding to the Lagrangian of Eq. (1) a term $L \rightarrow\ C\sum_{i=1}^{m}\xi_i$. This, in turn, leads to a new constraint on the Lagrange parameters $0 \leq \alpha_i\ \leq C.$ In the next chapter we will see how this soft margin allows us to declare points as outliers in a clustering problem.

References

Vapnik, Vladimir N.; *The Nature of Statistical Learning Theory*, Springer-Verlag, 1995. ISBN 0-387-98780-0

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992, July). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (pp. 144-152). ACM.

Corinna Cortes, Vladimir Vapnik. Support-Vector networks. Machine Leaming, 20, 273-297 (1995)

Fig. 1 reproduced from Wikipedia. Origin: By Larhmam - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=73710028

Fig. 2 reproduced from Wikipedia. Origin: By Shiyu Ji - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=60458994

Chapter 6. Support Vector Clustering (SVC)

So far we have dealt with classification methods of data. This falls into the realm of **supervised learning**, i.e. machine learning based on labelled data. The labels are being learned by the machine, and should generalize well when new data are being tested on it.

Another branch of ML is **unsupervised learning**, where unlabeled data are provided to the machine, and it succeeds to put some order into it. This is often referred to as **clustering**, i.e. separating data into different clusters; when such clusters coincide with known classes, i.e. all data-points share some known property which has not been known to the clustering algorithm, this is considered as a worthy achievement.

These two different learning categories are sometimes described by referring to how children learn languages. Teaching words to babies coincides with the paradigm of supervised learning. Hence the **semantics** of a language, i.e. the meaning of it, may be viewed as acquired by supervision. On the other hand, the **syntax** of a language, i.e. the grammar rules to be used for proper construction of sentences, are often assumed to be acquired by unsupervised learning, i.e. by listening to many examples of speech and absorbing implicitly the correct rules.[1] These statements may be subject to debate among linguistic experts, and are just intended to exemplify the two ways of learning without claiming that this short paragraph describes the full truth on this topic.

Following our demonstration of the SVM, we discuss here Support Vector Clustering (SVC) [Ben-Hur 2001]. Presented with data points $\mathbf{x}_j$ in a Euclidean space, we seek a transformation into Hilbert space such that for every data point its projection fits into the inside of a sphere

$$||\Phi(\boldsymbol{x}_j)-\boldsymbol{a}||^2 \leq R^2$$

where R is the radius of the sphere whose center is $\boldsymbol{a}$. Requiring the minimization of R subject to the existence of the constraints that all projections of data points are enclosed by this sphere, can be performed through applying the SVM formalism to the Lagrangian

$$L = R^2 - \sum_{j=1}^{m}(R^2 - ||\Phi(\boldsymbol{x}_j) - \boldsymbol{a}||^2)\beta_j$$

where $1 \geq \beta_j \geq 0$ are Lagrange multipliers. Minimizing L with respect to R and to $\boldsymbol{a}$ one finds the conditions

$$\sum_{i=1}^{m}\beta_i = 1 \text{ and } \boldsymbol{a} = \sum_{j=1}^{m}\beta_j\,\Phi(\boldsymbol{x}_j)\ .$$

Introducing them into L one is lead to the dual Lagrangian

$$W = \sum_{j=1}^{m}\beta_j\ \Phi(\boldsymbol{x}_j)^2 - \sum_{i=1,j=1}^{m}\beta_i\beta_j\ \Phi(\boldsymbol{x}_j)\cdot\Phi(\boldsymbol{x}_j)$$

$$= \sum_{j=1}^{m}\beta_j\ K(\boldsymbol{x}_j,\boldsymbol{x}_j) - \sum_{i=1,j=1}^{m}\beta_i\beta_j\ K(\boldsymbol{x}_i,\boldsymbol{x}_j),$$

where we have employed the kernel trick.

Applying this analysis to an artificial data set one gets the results shown in Fig. 1 for different values of the parameter q in a Gaussian kernel

$$K(\boldsymbol{x}_i,\boldsymbol{x}_j) = e^{-q||\boldsymbol{x}_i-\boldsymbol{x}_j||^2}\ .$$

The different contours in data space delineate the inverse projection from the surface of the sphere in Hilbert space. As q is being increased, the surface area decreases, and this is reflected in data-space by tighter binding of data points within the contours. Thus clustering is provided by SVC through boundaries which group the data-points.

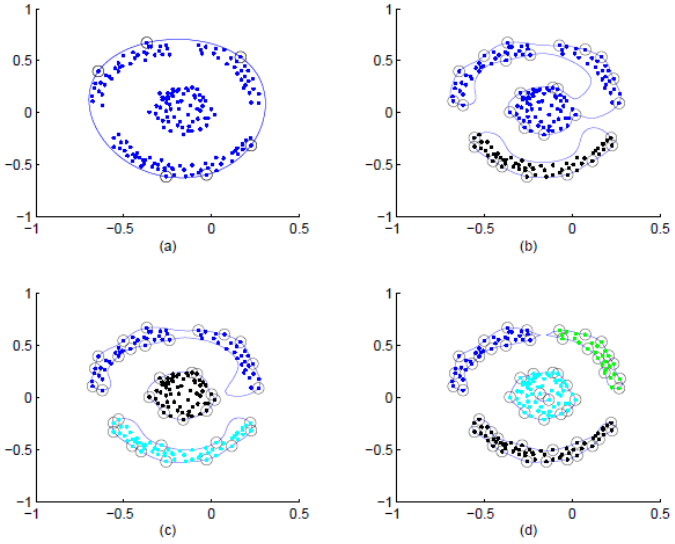


Fig. 1. Clustering of an artificial set of data points by SVC. Small circles are SVs. Different clusters are differently colored. The four different clustering choices shown here are obtained by choosing q = 1, 20, 24, 48 for sub-figures a, b, c, d, accordingly. Reproduced from [Ben-Hur, 2001].

Moving from strict to soft boundary conditions is tantamount in the clustering formalism to allowing for outliers, i.e. points which lie outside cluster boundaries. This can be implemented through declaring an upper bound on the Lagrange multipliers

$1 > C \geq \beta_j \geq 0$.

For large number of points N, the number of outliers turns out to be $1/C$. It is useful to consider the fraction of outliers $p = 1/NC$, as a parameter which we can have at our disposal in this analysis. This is particular useful in problems where the clusters of data points are not well separated. By declaring some of the points to be outliers, the formalism will allow for clustering the remaining points.

An interesting example is provided in Fig. 2, where one attempts the clustering of a data-set of crabs (4 classes of 50 subjects each, whose 5-dim physical measurements are analyzed with PCA and projected on the 2d plane of PC2 and PC3) which is one of the examples in the text book of [Ripley 1996].

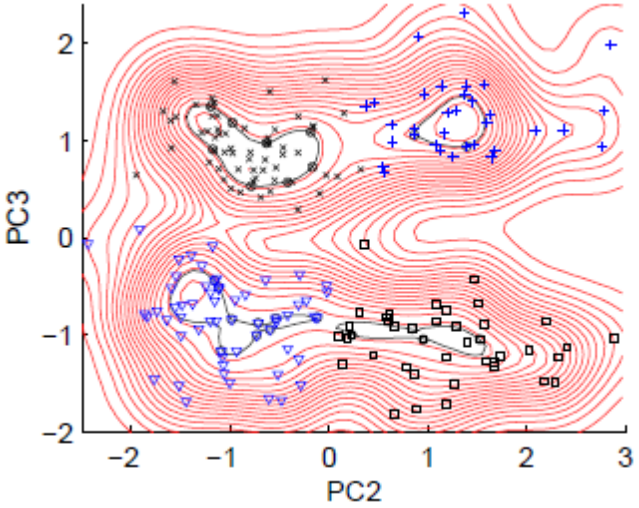


Fig. 2. SVC of the crab data for *q=4.8, p=0.7*. This provides 4 core-clusters with many (70%) outliers, which may be associated with their nearest cores by geometric inspection. Reproduced from [Ben-Hur 2001].

The topographic map in Fig. 2 is that of the Parzen window estimator of the density function (in *d* dimensions)

$$P_q(\boldsymbol{x}) = \frac{1}{N}\left(\frac{q}{\pi}\right)^{d/2} \sum_{i=1}^{N} K_q(\boldsymbol{x}_i, \boldsymbol{x})$$

demonstrating that the clusters lie at the peaks of *P* for this choice of *q=4.8*.

References

B.D. Ripley. Pattern recognition and neural networks. Cambridge University Press, Cambridge, 1996.

B. Schőlkopf, R.C. Williamson, A.J. Smola, J. Shawe-Taylor, and J. Platt. Support vector method for novelty detection. in Advances in Neural Information Processing Systems 12: Proceedings of the 1999 Conference, Sara A. Solla, Todd K. Leen and Klaus-Robert Muller eds., 2000.

A.Ben-Hur, D. Horn, H.T. Siegelmann, and V. Vapnik. Support Vector Clustering. Journal of Machine Learning Research 2 (2001) 125-137.

A. Ben-Hur, D. Horn, H.T. Siegelmann, and V. Vapnik. A support vector clustering method. in Advances in Neural Information Processing Systems 13: Proceedings of the 2000 C ference, Todd K. Leen, Thomas G. Dietterich and Volker Tresp eds., 2001.

Footnote 1. Chomsky has claimed that the language ability of human beings indicates that the human brain contains a "language organ" providing a "universal grammar" capability.

Chomsky, N. (1986) *Knowledge of Language: Its Nature, Origin, and Use* (Praeger, New York).

Pinker, S. (2013) *Language, Cognition, and Human Nature.* Print ISBN-13: 97801993287 Published to Oxford Scholarship Online: January 2014 DOI:10.1093/acprof:oso/9780199328741.001.0001

Pinker, S. and Jackendoff, R. (2005). The faculty of language: what's special about it? *Cognition* 95 (2005) 201–236

Chapter 7. Novel Formulation of Parzen Data Analysis

This chapter is based on my papers https://arxiv.org/ftp/arxiv/papers/1808/1808.08776.pdf 2018, and https://www.intechopen.com/online-first/novel-formulation-of-parzen-data-analysis 2019.

The Parzen window density is a well-known technique, associating Gaussian kernels with data points. It is a very useful tool in data exploration, with particular importance for clustering schemes and image analysis. The Parzen-window distribution [1], has been introduced in 1965, and still serves the goal of pattern recognition [2]. Given a set case of data points within some Euclidean data - or feature - space of dimension d, with possible positive attributes (e.g. intensities) I_i, described by an experimental distribution

$$Q(\mathbf{y}) = \frac{1}{N}\sum_i I_i \delta(\mathbf{y} - \boldsymbol{\mu}_i) \qquad (1)$$

where $N = \sum_i I_i$. In doing so, we assume that the measurement errors are small by comparison to the distances between the points and to the resolution with which we wish to study the data (otherwise replace the delta-functions by Gaussians with appropriate widths). The Parzen-window distribution can be represented in the following (not normalized) fashion

$$\psi_q(\mathbf{x}) = \int d\mathbf{y}\, Q(\mathbf{y})\, K_q(\mathbf{x} - \mathbf{y}) \qquad (2)$$

where we employ the Gaussian kernel

$$K_q(\mathbf{x} - \mathbf{y}) = \exp\left(-q\ (\mathbf{x} - \mathbf{y})^2\right) \qquad (3)$$

where $q = \frac{1}{2\sigma^2}$ is the resolution, with σ being the Gaussian width, or standard-deviation.

Following [3] we define a relative probability weight

$$p_q(\mathbf{x}|\mathbf{y}) = \frac{K_q(\mathbf{x}-\mathbf{y})}{\psi_q(\mathbf{x})} \qquad (4)$$

which represents the influence of point y in Q(y) on point x in ψ(x), and is normalized at any point x through

$$\int d\mathbf{y}\, Q(\mathbf{y})\, p_q(\mathbf{x}|\mathbf{y}) = 1\ . \qquad (5)$$

This formulation allows us to generalize the conventional Parzen-window formulation and introduce new concepts. We start by defining the potential function $V_q(\mathbf{x})$ through

$$V_q(\mathbf{x}) = -q\frac{\partial}{\partial q}\log\psi_q(\mathbf{x}) = q\int d\mathbf{y}\, Q(\mathbf{y})(\mathbf{x}-\mathbf{y})^2 p_q(\mathbf{x}|\mathbf{y}) \equiv q < (\mathbf{x}-\mathbf{y})^2 > \qquad (6)$$

where the last equality serves as the definition of an expectation value under the probability defined by Eqs. (4) and (5). In the mathematical appendix we discuss several examples of manipulations of such expectation values. It is now quite straightforward to derive the relation

$$-\frac{1}{4q}\nabla^2\psi_q + V_q\psi_q = \frac{\mathrm{d}}{2}\psi_q \qquad (7)$$

which is a Schrődinger equation obeyed by ψ with the potential $V_q(\mathbf{x})$ in a d-dimensional Euclidean space of $\mathbf{x}$. This coincides with the formalism of Quantum Clustering (QC) [4] to which we will return later.

Another interesting concept one may define is the entropy function

$$H_q(\mathbf{x}) = -\int d\mathbf{y}\, Q(\mathbf{y})\ p_q(\mathbf{x}|\mathbf{y}) \log\ p_q(\mathbf{x}|\mathbf{y}) = -<\log p_q(\mathbf{x}|\mathbf{y})> \qquad (8)$$

which may also be rewritten as

$$H_q(\mathbf{x}) = \log \psi_q(\mathbf{x}) + V_q(\mathbf{x}) \,. \qquad (9)$$

Eq. (9) has an analog in statistical mechanics [5], where H is the entropy, V is the average energy of a canonical ensemble, and ψ is its partition function. Note, however, that the scalar fields ψ, V and H are functions of space, rather than constants of a physical system.

Examples of the behavior of $\log \psi$ and of V are demonstrated in Fig. 1 for a data set of 9000 observed galaxies (with red-shift in the domain 0.47 ± 0.005) regarded as points in spherical angles θ and φ within some limited range. The set of galaxies downloaded from the Sloan Digital Sky Server DR12.

Whereas for $\sigma = \frac{1}{\sqrt{2q}} = 2$ (in angle degree units) the two fields exhibit many extrema, there exist clear differences for larger sigma, e.g. σ=10, where $\log \psi$ has one maximum whereas V maintains several minima. The different structures can be employed to formulate different clustering methods to which we return in the next chapter.

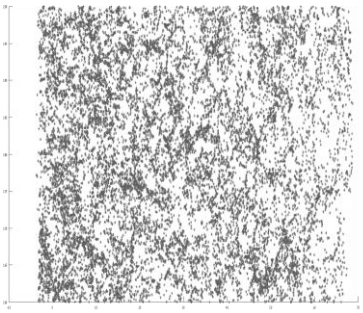


Fig. 1a. Loci of 9000 Galaxies within some range of θ and φ

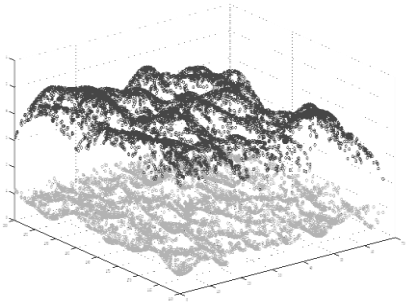


Fig. 1b. $\log \psi$ (top) and V (bottom) are displayed over the data plane for σ=2.

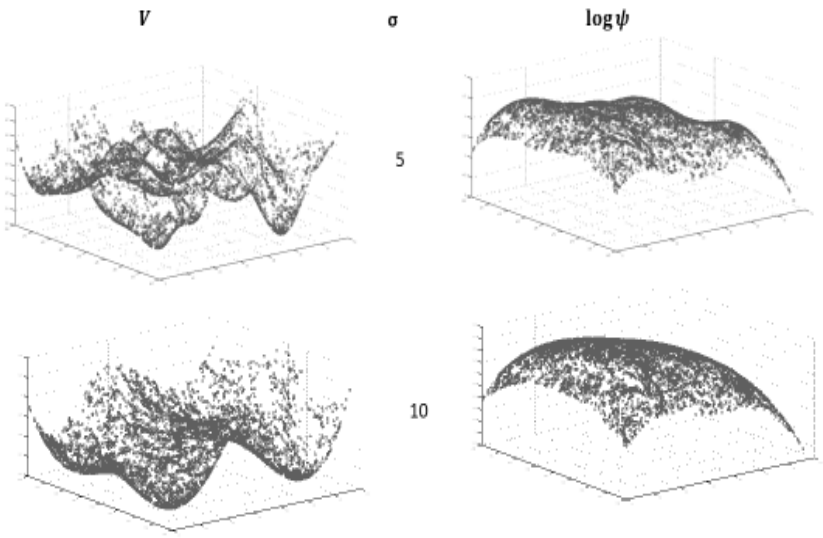


Fig. 1c. Surfaces of V and $\log\psi$ for increased values of the Gaussian width.

Extrema of the different scalar field occur when their derivatives vanish. This calls for the definition of appropriate vector fields

$$\mathbf{D} = 2q < \mathbf{x} - \mathbf{y} > \qquad (10)$$

$$\mathbf{E} = 2q^2 < (\mathbf{x} - \mathbf{y})^3 > \qquad (11)$$

obeying the relations

$$\nabla \log \psi_q(\mathbf{x}) = -\mathbf{D} \qquad (12)$$

$$\nabla V_q(\mathbf{x}) = \mathbf{D} + \mathrm{V}\mathbf{D} - \mathbf{E} \qquad (13)$$

$$\nabla H_q(\mathbf{x}) = \nabla V_q(\mathbf{x}) + \nabla \log \psi_q(\mathbf{x}) = \mathrm{V}\mathbf{D} - \mathbf{E}. \qquad (14)$$

Local maxima (and minima) of the probability function occur at **D**=0, maxima (and minima) of entropy occur where $\mathbf{E} = \mathrm{V}\mathbf{D}$ and minima (and maxima) of the potential occur where $\mathbf{E} = \mathrm{V}\mathbf{D} + \mathbf{D}$. All extrema coincide if both **D**=0 and **E**=0 at the same value of **x**. Otherwise, one finds at maxima of the Parzen probability that $\nabla V_q(\mathrm{x}) = -\mathbf{E}$.

An interesting conclusion from these equations is that

$$q\frac{\partial}{\partial q}\mathbf{D} = \mathbf{D} + V\mathbf{D} - \mathbf{E} = \nabla V \,. \qquad (15)$$

This implies that maxima of ψ_q , where $\mathbf{D} = 0$, change with q in a direction determined by $-\nabla V$, i.e. toward close-by minima of *V*. Eq. (15) leads to a new interpretation of QC: clustering based on minimization of *V* may be interpreted as clustering based on *stationarity of **D** with respect to changes in the scale-parameter q.*

Information regarding the different extrema of ψ and V can be united into one equation by considering another scalar function

$$U_q(\mathbf{x}) \equiv \mathbf{D}^2 = (2q < \mathbf{x} - \mathbf{y} >)^{\mathbf{2}} . \qquad (16)$$

Using the relations spelled out above, it follows that the condition

$$q \frac{\partial}{\partial q} U_q(\mathbf{x}) = -2\nabla \log \psi_q(\mathbf{x}) \cdot \nabla V_q(\mathbf{x}) = 0 \qquad (17)$$

implies that either ψ or V (or both) reach their extrema at the **x** values for which Eq. (17) holds. *U* is non-negative, therefore *U*=0 is a minimum of *q*. It corresponds to extrema of ψ which are associated with $\mathbf{D} = 0$. Other values of *U* which obey eq. (17) are associated with extrema of V which occur whenever $\frac{\partial}{\partial q} \mathbf{D}_q = 0$.

Eq. (17) is a concise statement regarding the points of interest in our data-analysis method. In analogy with statistics, one may view this equation as an inference method finding the correct parameter *q* which leads to points of interest at given values of **x**.

An interesting comparison can be made with the score function [6] in statistics, defined by

$$u(\theta) = \frac{\partial}{\partial \theta} \log f(\mathbf{x}|\theta) \qquad (18)$$

for a probability function *f(***x***)* which depends on the parameter θ. The average of *u* vanishes and its variance is known as Fisher's information [7]. The analogy of f with ψ and *u* with V is incomplete, because the integral of ψ is not normalized to 1. As a result, whereas the expectation value of *u* vanishes, the analogous statement in our analysis becomes

$$\left(\frac{q}{\pi}\right)^{\frac{\mathrm{d}}{2}} \int d\mathbf{x}\, \psi_q(\mathbf{x}) V_q(\mathbf{x}) = \frac{\mathrm{d}}{2} . \qquad (19)$$

Although all extrema may be regarded as points of interest, some are of more interest than others: extrema that remain fixed in **x** for a range of *q* values which is large compared with that of other points of interest. This criterion, introduced by Roberts [8], allows for searching for scales which correspond to important properties of the data. Thus it sub-serves the search for good clustering of the data [3,4,10].

Appendix: Mathematical Relations

From the various definitions we can derive the following relations

$$q \frac{\partial}{\partial q} \frac{1}{\psi} = -\frac{1}{\psi} q \frac{\partial}{\partial q} \log \psi = \frac{V}{\psi} \qquad \nabla \frac{1}{\psi} = \frac{\mathbf{D}}{\psi}$$

$$\nabla < f > = \mathbf{D} < f > + < \nabla f > - 2q < (\mathbf{x} - \mathbf{y}) f >$$

$$q \frac{\partial}{\partial q} < f > = V < f > + < q \frac{\partial}{\partial q} f > - q < (\mathbf{x} - \mathbf{y})^2 f >$$

Thus

$$\nabla V_q(\mathbf{x}) = 2q^2 < (\mathbf{x} - \mathbf{y})^{\mathbf{2}} > < \mathbf{x} - \mathbf{y} > + 2q < \mathbf{x} - \mathbf{y} > - 2q^2 < (\mathbf{x} - \mathbf{y})^3 >$$

which leads to Eq. (13).

Low resolution limits.

Expanding the Gaussian kernel in q, one may express $\psi_q(\mathbf{x})$ in terms of a Taylor series of weighted moments of the data,

$$\psi_q(\mathbf{x}) = \sum_n \frac{(-q)^n}{n!} \int d\mathbf{y}\, Q(\mathbf{y})\, (\mathbf{x}-\mathbf{y})^{\mathbf{2n}},$$

which, for the explicit data representation of Eq. (1) becomes

$$\psi_q(\mathbf{x}) = \frac{1}{N} \sum_n \frac{(-q)^n}{n!} \sum_i I_i \;\; (\mathbf{x}-\boldsymbol{\mu}_i)^{2n} \quad .$$

In particular, in the asymptotic limit of low q one finds

$$\psi_q(\mathbf{x}) \approx 1 - \frac{q}{N} \sum_i \; I_i \;\; (\mathbf{x}-\boldsymbol{\mu}_i)^2$$

References

[1] E. Parzen, On estimation of a probability density function and mode, The annals of mathematical statistics 33 (3) (1962) 1065-1076.

[2] D. Xu, Y. Tian, A comprehensive survey of clustering algorithms, Annals of Data Science 2 (2) (2015) 165-193.

[3] L. Deutsch and D. Horn: The Weight-Shape Decomposition of Density Estimates: A Framework for Clustering and Image Analysis Algorithms. *Pattern Recognit.* 81 (2018) 190-199.

[4] D. Horn and A. Gottlieb. Algorithm for data clustering in pattern recognition problems based on quantum mechanics. *Phys. Rev. Lett.* 88(1) (2001) 018702.

[5] E. T. Jaynes. Information theory and statistical mechanics. *Phys. Rev.* 1957, 106(4):620.

[6] D. R. Cox, and D. V. Hinkley (1974). Theoretical Statistics. Chapman & Hall.

[7] R. A. Fisher. Theory of Statistical Estimation. Math. Proc. Philos. Society 22 (1925) 700-725.

[8] S. J. Roberts, Parametric and non-parametric unsupervised cluster analysis, Pattern Recognition 30 (2) (1997) 261-27

Chapter 8. Weight Shape Decomposition (WSD) and Quantum Clustering (QC)

In the previous chapter we have introduced the concepts of entropy and potential fields, obeying

$$H_q(\mathbf{x}) = \log \psi_q(\mathbf{x}) + V_q(\mathbf{x})$$

where ψ is the un-normalized Parzen probability density. This relation can be rewritten as

$$\psi(\mathbf{x}) = W(\mathbf{x})S(\mathbf{x})$$

using [1] the concepts of Weight $W(\mathbf{x}) = e^{H(\mathbf{x})}$ and Shape $S(\mathbf{x}) = e^{-V(\mathbf{x})}$. Since $V(\mathbf{x}) \geq 0$, it follows that $S(\mathbf{x}) \leq 1$. Moreover, both W and S are non-negative. S is integrable over $\mathbf{x}$ and, as such, can also serve as a distribution.

To demonstrate the meaning of these functions we plot in Fig. 1 the results of an exercise in which 200 data points were generated around x= +3 and 100 around x=-3 from two random Gaussian distributions with σ=1. To these were added 25 random data points around x=0 with σ=0.25. The W function captures correctly the 200:100 ratio, but only the S distribution captures the three Gaussian features. This is made possible by factoring out W, with its 2:1 bias, from the probability function. This exercise demonstrates the potential power of S in **anomaly detection**.

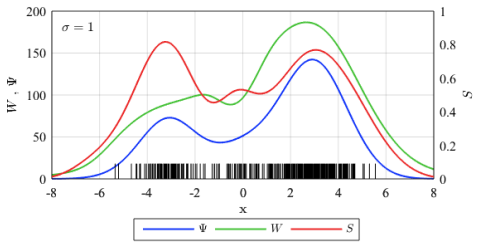


Fig. 1. Artifically generated random points, represented by short bars, lead to the Parzen probability function (blue) and its Weight and Shape components, represented by the green and red curves correpondingly. The W curve reproduces correctly the bias of about 100 points to the left and 200 points to the right, but only the S curve brings out the third Gaussian component of additional 25 points around x=0. I thank Lior Deutsch for generating this figure.

Deutsch and Horn [1] have used the WSD in order to establish three clustering mechanisms, which amount to MPC, MWC and MSC. These abbreviations stand for Maximal Probability (Weight and Shape) Clustering correspondingly. The algorithms are based on letting replicas of the data points move according to gradient-ascent up to the correponding maxima of the three different fields:

$$\mathbf{x}_i' \leftarrow \mathbf{x}_i' + \eta \nabla \chi(\mathbf{x}_i')$$

where $\chi \epsilon \{\psi, W, S\}$ for X= P, W or S in MXC. $\mathbf{x}_i'$ start out at the loci of the data $\mathbf{x}_i$ and end up at the field maxima which are their closest attractors. η is a small parameter to be chosen by the user. For more technical details, and different variants of these algorithms, see [1].

MPC is identical with the Mean Shift (MS) algorithm [2,3] and MSC coincides with the Quantum Clustering (QC) algorithm [4], where points were allowed to follow gradient descent to the attractors formed by minima of the potential V.

It is quite illuminating to test these three clusering methods on the crab data investigated in Chapter 6 with SVC. The results [1] are displayed in Fig. 2. The x-axis represents the common parameter σ. The y-axis of Fig. 2a displays the quality of the data as represented by the Jaccard score $0 \leq J = \frac{n_{11}}{n_{11}+n_{10}+n_{01}} \leq 1$ referring to the number of pairs of points in two classification schemes (expert classfication and clustering method) which belong together in both classifications, n_{11}, and which belong together in one but not in the other, $n_{10} + n_{01}$. Note that pairs of points which never belong in either the same class or cluster, n_{00}, are not included in this score. MSC, which coincides with QC, does best according to this criterion. Fig. 2b displays the numbers of clusters as function of σ. Also here QC is singled out as indicating the true number, 4, for a large range of σ. Moreover, QC displays stable consistency, with #clusters=4 being a stable result for a large range of σ. This is the criterion for choosing a clustering solution proposed by Roberts [5].

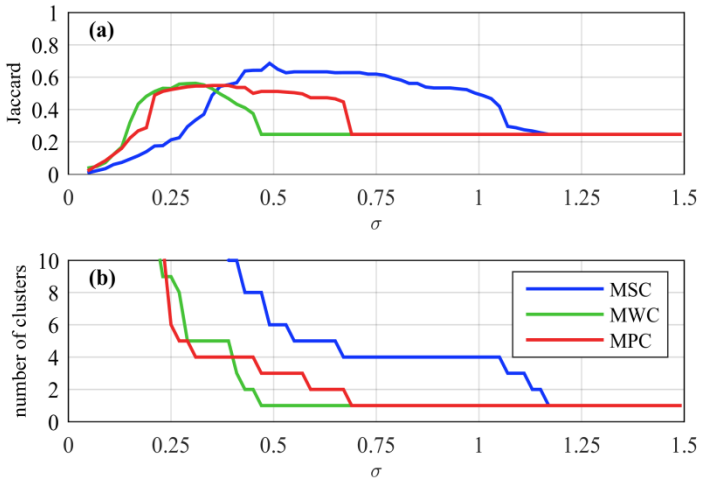


Fig. 2. (a) Comparison of the three clustering methods with expert classification. (b) Predicted numbers of clusters according to the three schemes. Reproduced from [1]. MSC coincides with QC, and MPC coincides with the MS algorithm.

In order to have a better understanding of these results we compare the data (colored according to expert classification) with topographic maps corresponging to the three different methods, for both σ=0.3 and 0.7, in Fig. 3. The first value correponds to a candidate correct solution for MPC (or MS), and may be compared with the SVC solution in Fig. 2 of Chapter 6, with MS supplying the gradient-ascent mechanism allowing for the outliers of SVC to join their cluster-cores. The value σ= 0.7 leads to topological structures which end up in a single cluster for MPC and MWC, whereas MSC=QC has the structure leading to the correct result.

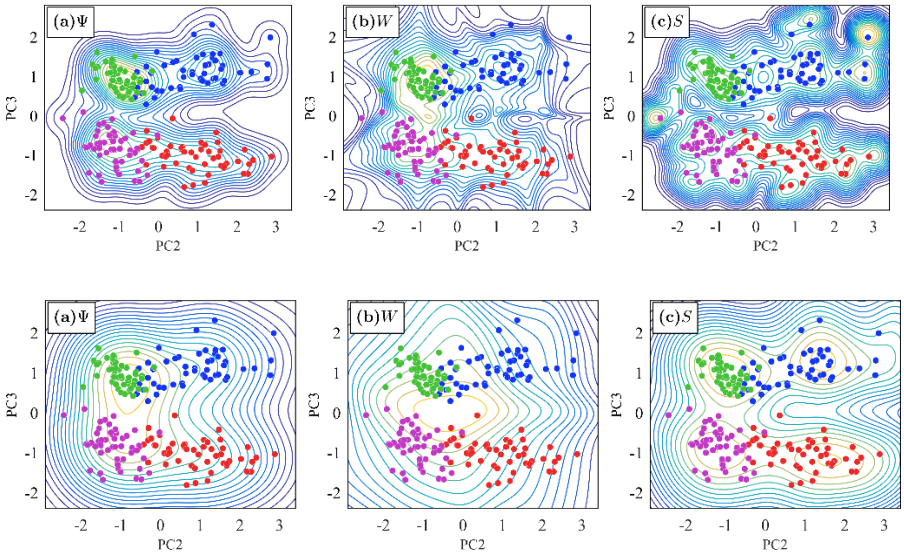


Fig. 3. Topological maps for the three methods in the cases of (top) σ=0.3 and (bottom) σ=0.7. Compare with Fig. 2 to understand the results of the former figure. Reproduced from [1].

References

[1] L. Deutsch and D. Horn. The Weight-Shape Decomposition of Density Estimates: A Framework for Clustering and Image Analysis Algorithms. *Pattern Recognit.* 81 (2018) 190-199.

[2] Y. Cheng. Mean shift, mode seeking, and clustering, IEEE transactions on pattern analysis and machine intelligence 17 (8) (1995) 790-799.

[3] D. Comaniciu, P. Meer. Mean shift: A robust approach toward feature space analysis, IEEE Transactions on pattern analysis and machine intelligence 24 (5) (2002) 603-619.

[4] D. Horn and A. Gottlieb. Algorithm for data clustering in pattern recognition problems based on quantum mechanics. *Phys. Rev. Lett.* 88(1) (2001) 018702.

[5] S. J. Roberts. Parametric and non-parametric unsupervised cluster analysis, Pattern Recognition 30 (2) (1997) 261-27

Chapter 9. Shape Analysis of Images

This chapter is based on my paper https://www.intechopen.com/online-first/novel-formulation-of-parzen-data-analysis 2019 and on ref [3].

In the late 90s it has become fashionable to use the Parzen-window approach for image analysis. The framework of these studies was also known as "scale-space" [1], referring to the parameter q or σ used for the underlying Gaussian kernels. Varying the scale to discover different features has been introduced in the 80s [2]. It seems therefore reasonable to ask if the WSD can contribute novel insights to the scale-space approach of image analysis.

The generalized formalism of the Parzen approach, as described in Chapter 6, allows for image analysis by viewing each pixel of an image as a data point with $\boldsymbol{i}$ on a grid in two dimensions, and assigning it a weight I_i which corresponds to the gray scale of the pixel.

This allows for writing the Parzen probability function as a 2d convolution:

$$\psi[\boldsymbol{i}] = -\sum_j I[\boldsymbol{j}]K[\boldsymbol{j}-\boldsymbol{i}] \equiv (I * K)[\boldsymbol{i}] \qquad (1)$$

Employing the formalism of WSD [3] one can then show that the potential function becomes

$$V[\boldsymbol{i}] = \frac{(I*L)[\boldsymbol{i}]}{(I*K)[\boldsymbol{i}]} \quad \text{where} \quad L = -K\ log\ K. \qquad (2)$$

The Weight and Shape functions are defined by $W(\boldsymbol{i}) = e^{H(\boldsymbol{i})}$ and $S(\boldsymbol{i}) = e^{-V(\boldsymbol{i})}$ as before.

As an example we reproduce in Fig. 1 the matrices K and L on a 7x7 grid employing σ=1.1 in pixel (or grid) units.

(a)

0	0	0.02	0.02	0.02	0	0
0	0.04	0.13	0.19	0.13	0.04	0
0.02	0.13	0.44	0.66	0.44	0.13	0.02
0.02	0.19	0.66	1	0.66	0.19	0.02
0.02	0.13	0.44	0.66	0.44	0.13	0.02
0	0.04	0.13	0.19	0.13	0.04	0
0	0	0.02	0.02	0.02	0	0

(b)

0	0.02	0.07	0.09	0.07	0.02	0
0.02	0.12	0.26	0.32	0.26	0.12	0.02
0.07	0.26	0.36	0.27	0.36	0.26	0.07
0.09	0.32	0.27	0	0.27	0.32	0.09
0.07	0.26	0.36	0.27	0.36	0.26	0.07
0.02	0.12	0.26	0.32	0.26	0.12	0.02
0	0.02	0.07	0.09	0.07	0.02	0

Fig. 1 The matrices (a) K and (b) $L = -K\ log\ K$ on a 7x7 grid using σ=1.1. Reproduced from [3]. Convolution with the data is being used to evaluate the potential, as defined in Eq. (2).

We have previously seen that V is a normalized second derivative of the probability function. Hence it can also serve as an edge detector. This property leads to the fact that the large components of Shape can serve as producing line-caricatures of images, as demonstrated in Fig.2. In these figures we have used a truncated Gaussian filter with σ = 1.1, such as in Fig. 1a. Before applying the filter, the original image was rescaled to the range [0.1, 0.9]. For display purposes, the shape image pixel values were truncated at the image's 1% and 99% percentiles, and the image is displayed inversely (high values are dark, low values are bright) [3].

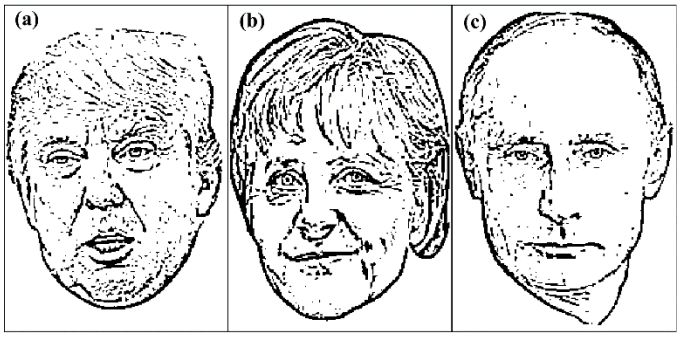


Fig. 2. Line caricatures produced by plotting the high Shape-values derived from three gray-scale figures. All images were thresholded at the same large shape value. (a) Image size: 254 × 198. (b) Image size: 267 × 189. (c) Image size: 268 × 188. Reproduced from [3].

This methodology can be applied to 3d images, in imaging data which are of interest in bio-med research and in technology. To demonstrate the third point we display in Figure 3 the results of an ansalysis of a T2 MRI of the brain of a Macaque monkey [Fisher 2018]. Deriving both P and S, rescaling them, and limiting ourselves to large relative values (thresholded components) of Probability and Shape, we find that the latter peaks in cortical regions, whereas the former peaks in internal regions of the brain. This is demonstrated in Figure 3, exhibiting the occurrence of large Shape values in the outer brain structures, and large Probability values in the inner brain structures. Thus, a simple thresholding procedure allows one to easily segment the MR image, for the purpose of further analysis of the cortex by applying QC to the MRI input in the large *S* domain. In Figure 4 we follow these conclusions with a display of QC clusters projected onto the surface of the brain, leading to its parcellation into cortical omponents which are derived by just computational image analysis.

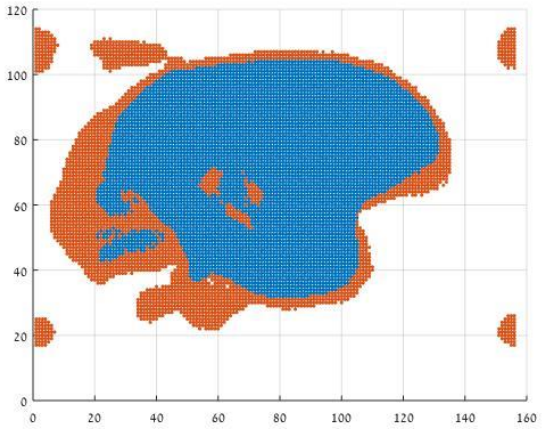


Figure 3. Thresholded Shape (red) and thresholded Probability (blue) dominate different regions within the same MR image of a brain, projected onto its (side-view) y-z plane. This analysis used σ=3 in voxel units. Data outside the brain are due to artefacts and noise in the MR image. This result of a T2 MRI brain image of a macaque monkey indicates that large Shape components dominate external (cortical) regions of the brain, whereas large Probability compnents dominate internal brain regions. Reproduced from [5].

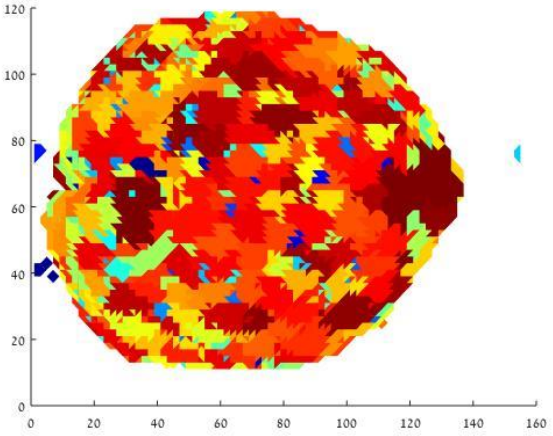


Figure 4. Characteristic results of QC cortical clusters as mapped onto the surface of the brain, and projected on the (top view) x-y plane. This figure displays a map of the largest clusters of Shape, each described by a different color. Reproduced from [5].

References

[1] T. Lindeberg. "Scale-space theory: A basic tool for analyzing structures at different scales". Journal of applied statistics 1994, 21(1-2):225–270.

[2] A. Witkin. "Scale-space filtering: A new approach to multi-scale description" . Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'1984, Vol. 9, pp. 150–153.

[3] Deutsch L, Horn D. The Weight-Shape Decomposition of Density Estimates: A Framework for Clustering and Image Analysis Algorithms. Pattern Recognit. 81 (2018) 190-199.

[4] Fisher I. Parcellation of Brain Images using Shape analysis [M.Sc. thesis]. Tel-Aviv University, 2018. http://horn.tau. ac.il/publications/FisherThesis.pdf

[5] D. Horn (2019). Novel Formulation of Parzen Data Analysis. https://www.intechopen.com/online-first/novel-formulation-of-parzen-data-analysis

Chapter 10. From Shallow to Deep Networks

Classification of handwritten numerals was one of the important technical problems which has been investigated by neural networks. LeCun et al 1998 have investigated various architectures, achieving test error rates between 2.5% to 4.7%. In its modern incarnation, the relevant dataset is called MNIST, including a training set of 60000 examples and a test set of 10000 examples. Modern deep networks achieve error rates of a few percent, as we shall see below. Another related data set is that of FASHION, which was constructed to serve the same task, by including the same numbers of train and test examples, and being also classified into 10 classes, in analogy to the ten digits.

Following LeCun 1998, many of the deep networks are constructed as Convolutional Neural Networks (CNN). The architecture is demonstrated in Fig. 1, taken from LeCun 1995. The first convolutional operation was inspired by known structures in the human brain, where neurons are connected by a limited "visual field" to other neurons. The important step is therefore implementing a limited connectivity between layers, as well as allowing for subsampling, thus reducing layer sizes while increasing their numbers. The idea was that the different feature maps will lead to increasing feature hierarchies, thus understanding that the network leads from primitives (e.g. edge detection) to increasing conceptual understanding leading eventually to the required learning.

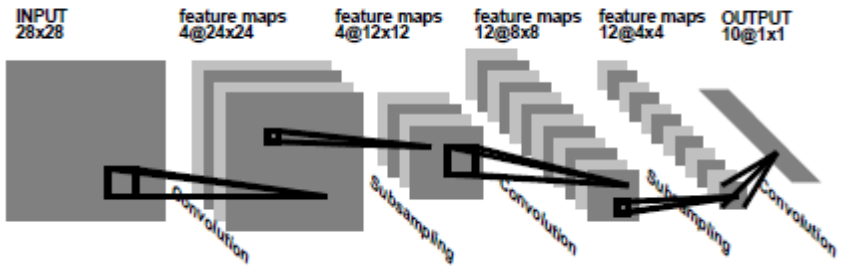


Figure 1. The architecture of LeNet1 (LeCun 1995), which has led to the concept of CNN.

ILSVRC has organized a competition of image classification in 2012, which was based on ImageNet, a dataset containing 15 million labeled images. ILSVRC used a subset of 1.2 million images, containing 1000 images in each of 1000 categories. The winner was AlexNet (Krizhevsky, A., Sutskever, I., Hinton, G), which contains 5 convolutional layers, some of which are followed by max-pooling layers, ending with 3 fully connected layers.

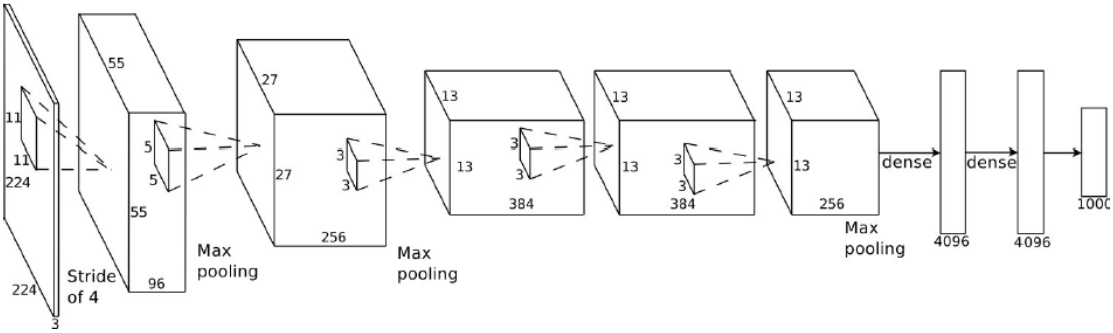


Figure 2. The architecture of AlexNet. The input-layer has a width of 3, corresponding to RGB colors. "Max-pooling" propagates the highest value in its visual field. "Dense" means fully connected coupling. Reproduced from Krizhevsky (2012).

Containing altogether 650 K neurons, some 60M parameters and 630 M connections, AlexNet was trained with stochastic gradient descent (SGT) on two Nvidia GPUs for about a week, leading to winning the 2012 competition. The following figure describes the low-level filters of the first layer. AlexNet has also introduced the ReLU transfer function,

ReLU(x)=0 if x<0,

ReLU(x)=x if x>0,

replacing the sigmoid to become the most popular transfer function in deep networks. AlexNet heralded the age of Deep Learning. It became evident that one can apply the CNN methodology to Big Data, which is a rising demand in modern science and technology.

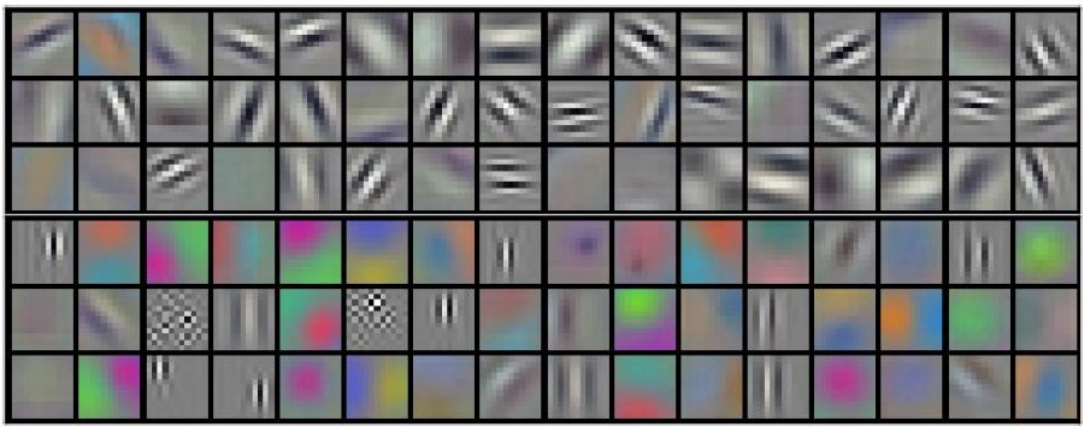


Figure 3. 96 learned low-level filters. Reproduced from Krizhevsky (2012).

To explain the working of CNN, Zeiler and Fergus (2013) developed a deconvolutional algorithm, allowing for the demonstration of hierarchical feature abstraction. Given sets of activations in a given layer, it generates sets of input-pixels which can be regarded as the most favorable images which could lead to the observed patterns. Their results, for their own CNN trained on ImageNet, are displayed in Figure 4. Shown, for each layer, are the top 9 activations in a random subset of feature maps, together with their corresponding generated pixel-space images by the deconvolutional network. Note the strong grouping of pixel-images, which increases as the hierarchy of the layer increases. The aim of Zeiler and Fergus was put forward in their title as "Visualizing and Understanding Convolutional Networks". Clearly we, the readers, regard visualization as an important part of our understanding. Hence we may conclude that they achieved their aim.

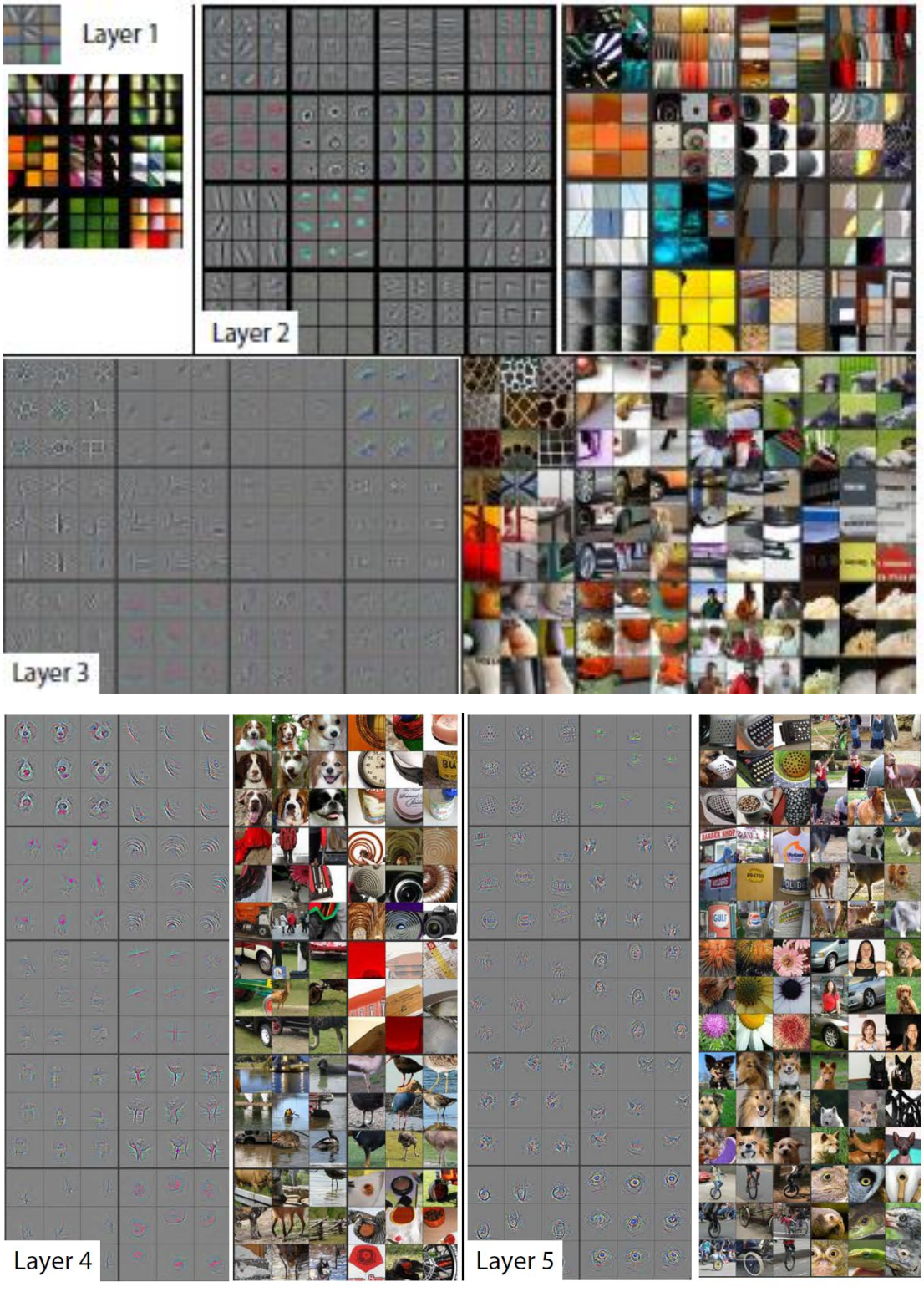


Figure 4, reproduced from Zeiler and Fergus (2013)

.

References

The MNIST data-base of handwritten digits is described in http://yann.lecun.com/exdb/mnist/

Fashion MNIST (Zolando Research) https://www.kaggle.com/zalando-research/fashionmnist

Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Comput. 1(4), 541–551 (1989)

Matthew D. Zeiler and Rob Fergus Visualizing and Understanding Convolutional Networks ECCV 2014. https://cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf

Y. LeCun, L. D. Jackel, L. Bottou, A. Brunot, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, P. Simard and V. Vapnik: Comparison of learning algorithms for handwritten digit recognition, in Fogelman, F. and Gallinari, P. (Eds), International Conference on Artificial Neural Networks, 53-60, EC2 & Cie, Paris, 1995

Introducing Deep Learning with MATLAB. Mathworks

Y LeCun: Deep Networks. Lecture at CERN, 2016.
https://indico.cern.ch/event/510372/attachments/1245509/1840815/lecun-20160324-cern.pdf

Chapter 11. The DL Paradigm and its Supremacy

With the obvious successes of Deep Learning (2015) it has turned into a technology that may be easily adapted to many applications. In order to popularize DL, Google has developed its open-system tool TensorFlow (TF), based on Python, within which all you have to do is to design the architecture of your model, and it will perform everything else, including sending the data to your system in batches for training and testing, adapting it to the hardware which best suits your computational needs (e.g., CPU or GPU), etc. The first example available on the website of TF tutorials is the MNIST problem. The TF code they propose is as follows:

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

This program defines the data-set mnist by loading the data (images=x and assignments =y) from Keras, which is TensorFlow's high-level API for building and training deep learning models. The model is a sequential NN model, starting with a "Flatten" layer which flattens the input of 28X28 pixel gray-scale (between 0 and 255) images into a vector of 512 entries. They are connected to a Dense layer with all-to-all connectivity using the ReLU transfer function. This is followed by a dropout operation at a rate of 0.2, and densely connected to the output layer through a softmax activation function, leading to an output of probabilities of the expected 10 digits. The compiler is told which optimization and loss functions to use, and what metrics to evaluate. Each epoch is automatically divided into many batches, running altogether over 60K data during each epoch. When completed, it runs over 10K test data to test its performance. This program achieves an accuracy of 0.98 within less than 1 minute.

This short and successful implementation is a tribute to the long way Neural Networks have come since the early multi-perceptron days. Note that this program is not a convolutional network, hence the intuitive understanding of how AlexNet solves image classification does not apply here. Nonetheless it works. The sheer power of training on multitudes of data in a high-parameter space of NNs does it: it finds a manifold within the large parameter-space which embeds correct minima of the training set, and generalizes well to the test set.

I recommend the Kadenze lecture referenced below as a general high-level course for interested beginners with programming experience in Python.

To demonstrate the incredible power of the DL technology, I recommend watching the YouTube lecture of Demis Hassabis (2019) who describes the achievements of the DeepMind team in general reinforcement learning that masters chess and Go games through self-play (Silver et al 2018). This team became famous when its ZeroGo algorithm has succeeded to beat the human Go champion in 2015. Whereas the first algorithm started out by learning from human players, it then proceeded to learn from its own games. Eventually the DeepMind team developed AlphaZeroGo which, given the basic rules of the game, learns from scratch using an evolutionary development of players, all based on the same learning algorithm, playing against each other. Finally, this team developed the AlphaZero algorithm which can learn from scratch every board-game and master it. This is demonstrated in Silver (2018), based on matches between AlphaZero and the leading competitors in Chess, Shogi and Go.

The AI index team publishes each year statistics of various kinds regarding AI activities and achievements. The 2018 issue contains a list of human performance milestones, including the following: 2016, object detection of ImageNet, with error of 3% while humans perform at the level of 5%. 2016, GO, beating human experts. 2017, Skin cancer classification, outperforming expert dermatologists. 2018, Chinese to English translation. 2018, Prostate Cancer Grading from prostatectomy specimens.

Evolution of Intelligence

It is interesting to note that the successes of DeepMind follow an evolutionary methodology. It behooves an analogy with molecular biology. The basic carriers of DNA, the chromosomes, are known to contain genes, as well as the needed programs to activate them into the structural proteins and RNA molecules which compose the functioning living cell. Yet a full comprehension of why and how the linear code of nucleotides in a chromosome does it, is not really available. Nonetheless there is one principle which is the acceptable paradigm, Darwin's Natural Selection. This is how Biology plays its evolutionary game, leading to selection of species which survive in a complex and changing world.

The DL evolutionary methodology is being used to design complex architectures which can cope with complex problems. Google defines "intelligence" as "the ability to acquire and apply knowledge and skills". Clearly novel DL applications justify the statement that machine intelligence can outperform human's intelligence on various tasks. Naturally we expect the list of such tasks to grow with the present intensive activity of DL research. While many may strive for the classic AI goal, with machines replacing humans, Michael Jordan (2018) argues that this misses the real revolution which is now in its infancy: developing technologies which lead to societal world-wide intelligent infrastructures, which may hopefully empower mankind.

References:

Yann LeCun, Yoshua Bengio & Geoffrey Hinton: Deep learning. Nature 521 (2015) 436. doi:10.1038/nature14539

Kadenze course: Creative Applications with TensorFlow
https://www.kadenze.com/courses/creative-applications-of-deep-learning-with-tensorflow/info

AI index. https://aiindex.org

Demis Hassabis: The Power of Self-Learning Systems. Lecture at MIT, 2019.
https://youtu.be/cEOAerVz3UU

D. Silver et al.: A general reinforcement learning algorithm that masters chess, shogi and Go through self-play. Science 362, 1140–1144 (2018)

M. Jordan: Artificial Intelligence—The Revolution Hasn't Happened Yet. Medium 2018.

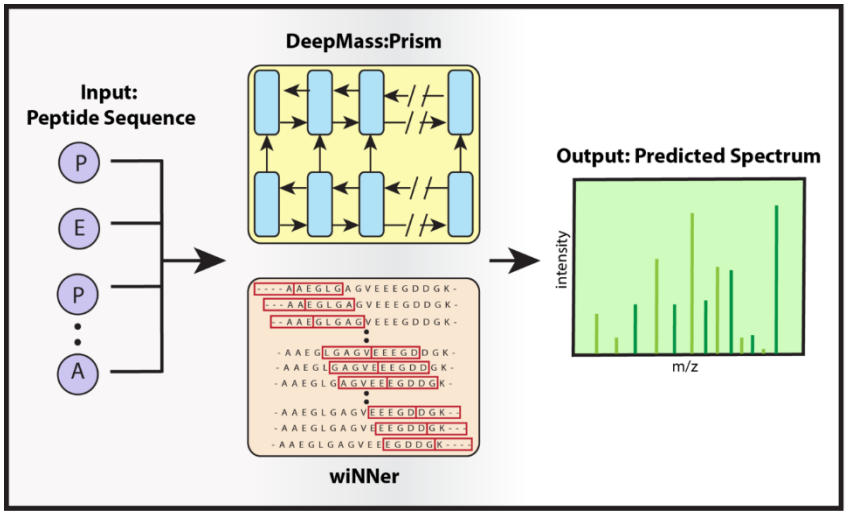
Chapter 12. Some Applications

Protheomics

Working independently, a team led by researchers at the Max Planck Institute of Biochemistry and Verily and a team led by researchers at the Technical University of Munich (TUM) developed deep learning tools for predicting patterns of ion fragmentation in mass spec-based proteomics.

According to their developers, the tools could help boost the number of proteins confidently identified in proteomic experiments and could also streamline data-independent acquisition mass spec work by allowing researchers to run such experiments without first generating sample-specific spectral libraries.

The software packages, called DeepMass:Prism by the Max Planck team and Prosit by the TUM team, are the latest of several efforts to apply deep learning methods to the analysis of mass spec proteomic data.



DeepMass:Prism discovered new chemical rules that determine how the peptides break into smaller fragments. Quoting Jurgen Cox in the MPG newsroom (2019): "This is a very exciting finding, it's like a junk yard employee who understands where a certain part of the car is installed, even though he has never seen this type of car before. The predictions by *DeepMass:Prism* have led to the identification of a new kind of interaction within proteins. We believe that this discovery is only the beginning of what deep learning can do for research in life sciences."

ProteinNet

ProteinNet is a standardized data set for ML of protein structure, i.e. relating the knowledge of the amino-acid sequence of the protein to its 3d spatial structure. It utilizes sensitive evolution-based distance metrics to segregate distantly related proteins, and additionally created validation sets distinct from the official CASP (Critical Assessment of protein Structure Prediction) sets. As stated in the paper, availability of such a data set can spur new algorithmic developments in protein bioinformatics and lower the barrier to entry for researchers from the broader machine learning community.

DL models based on ProteinNet, show continuously improving results in the biannual CASP competitions.

Neural Anatomy

Schubert et al (2019) have developed a method based on convolutional neural networks that reconstructs entire nerve cells with all their elements and connections almost error-free from image stacks. The data are obtained by a Serial Block-Face Scanning Electron Microscope (SBEM). The following figure, taken from Schubert (2019), shows the reconstructed labeling of neuron's dendritic tree by their cellular morphology neural network (CMN).

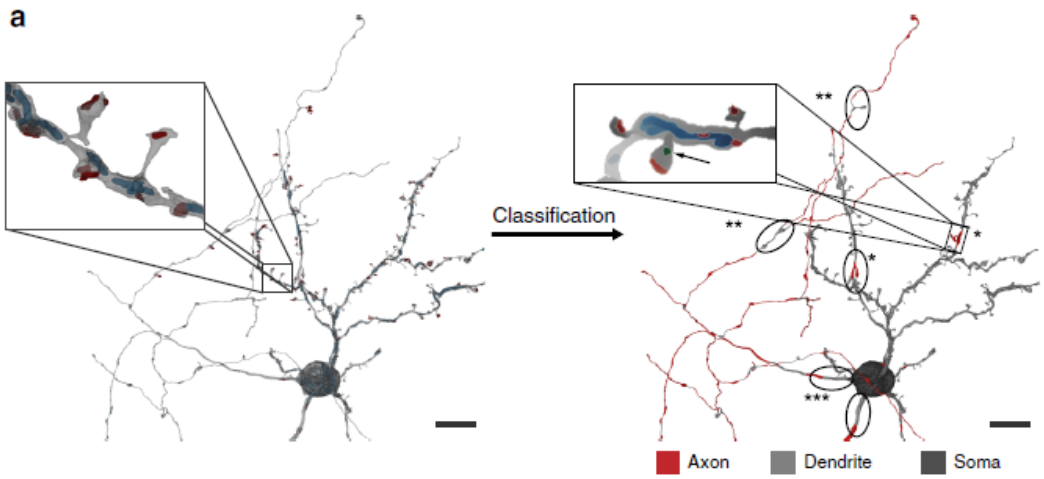


Fig. 5a from Schubert (2019). Cellular morphology neural network (CMN) prediction of subcellular compartments. The 3909 rendering locations of the reconstruction were predicted as axon, dendrite, or soma. Local errors are indicated by asterisks.

References

S. Tiwary, R. Levy, P. Gutenbrunner, F.S. Soto, K. Palaniappan, L. Deming, M. Berndl, A. Brant, P. Cimermancic and J. Cox, 2019. High-quality MS/MS spectrum prediction for data-dependent and data-independent acquisition data analysis. Nature Methods, May 2019

Learning from spectral experience: Deep learning algorithms facilitate the analysis of mass spectrometry data. MPG newsroom MAY 27, 2019.

Mohammed AlQuraishi. ProteinNet: a standardized data set for machine learning of protein structure. BMC Bioinformatics (2019) 20:311 https://doi.org/10.1186/s12859-019-2932-0

Winfried Denk & Heinz Horstmann: Serial Block-Face Scanning Electron Microscopy to Reconstruct Three-Dimensional Tissue Nanostructure PLOS Biology, October 2004:

Schubert P J, Dorkenwald S, Januszewski M, Jain V, Kornfeld J, 2019. Learning cellular morphology with neural netwroks. Nature communications 10, 2736 (2019)

Chapter 13. ML of ML Architectures

We have learned in chapter 2 how a neural network adjusts its parameters trough the backprop algorithm. But which network should one use to solve a given problem? This choice is conventionally done by experts who try various models in order to generate the best one. It has recently been proposed that the network architecture can also be learned through a DL algorithm. Zoph and Le (2017) propose an iterative scheme, pictured in Fig. 1, where one generates randomly several networks, trains them for a while, and uses the loss of these models in a reinforcement learning procedure

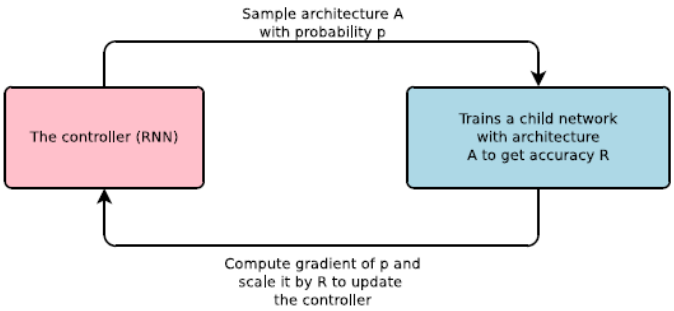


Figure 1: An overview of Neural Architecture Search.

The next figure, reproduced from Zoph (2018) demonstrates how a controller can be used to construct a basic element of the neural network.

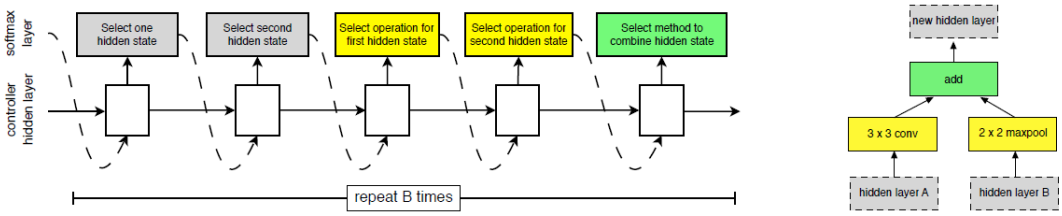
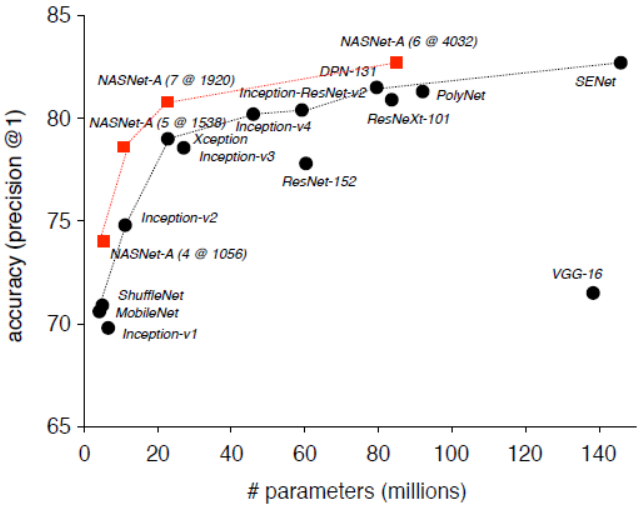


Figure 3. Controller model architecture for recursively constructing one block of a convolutional cell. Each block requires selecting 5 discrete parameters, each of which corresponds to the output of a softmax layer. Example constructed block shown on right. A convolutional cell contains B blocks, hence the controller contains $5B$ softmax layers for predicting the architecture of a convolutional cell. In our experiments, the number of blocks B is 5.

Testing this methodology on the ImageNet ILSVRC challenge of 2012, it outperforms all previous handcrafted models. The red squares in the next figure are the results of Zoph (2018) and are currently known as AuotoML. Further information is given in the you tube lecture by Jeff Dean.



Comparison of top performing CNN efforts on the ImageNet 2012 challenge. Reproduced from Fig. 5 of Zoph (2018).

These novel results of the Google Brain research team, demonstrate that automatic generation of DL models is a reachable goal. Together with the evolutionary schemes mentioned in chapter 11, the future may still surprise us, discovering what else DL models can achieve.

References

Jeff Dean, Google Senior Fellow (20.09.2018, ETH Zurich): Deep Learning to Solve Challenging Problems. YouTube lecture. https://youtu.be/ljBOzdKuX7A

Zoph B, Le Q V: Neural architecture search for neural reinforcement learning. ICLR 2017.

Zoph B, Vasudevan V, Schlens J, Le Q V. Learning transferrable architecture for scalable image recognition. CVPR 2018.