Density functionals from deep learning

Jeffrey M. McMahon

Department of Physics & Astronomy

March 15, 2016

Kohn–Sham Density-functional Theory (KS-DFT)

The energy functional of Hohenberg and Kohn[1]:

$$E[n] = \int d\mathbf{r}\, v(\mathbf{r}) n(\mathbf{r}) + F[n]$$

Kohn–Sham (KS) density-functional theory (KS-DFT)[2]:

$$F[n] = \boldsymbol{T_s[n]} + \boldsymbol{E_{\mathrm{xc}}[n]} + E_{\mathrm{H}}[n]$$

$$T_s[n] = \sum_{i=1}^{N} -\frac{1}{2} \int d\mathbf{r}\, \phi_i^*(\mathbf{r}) \nabla^2 \phi_i(\mathbf{r})$$

$$n(\mathbf{r}) = \sum_{i=1}^{N} |\phi_i(\mathbf{r})|^2 \quad , \quad n = \int d\mathbf{r}\, n(\mathbf{r})$$

$T_s[n]$: Noninteracting kinetic energy

$E_{\mathrm{xc}}[n]$: Exchange and correlation energy

$E_{\mathrm{H}}[n]$: Hartree energy

The computational time of KS-DFT is limited by the evaluation of $T_s[n]$, while the accuracy by the approximation of $E_{\mathrm{xc}}[n]$.

[1] P. Hohenberg and W. Kohn, *Phys. Rev.* **136**, B864 (1964)

[2] W. Kohn and L. J. Sham, *Phys. Rev.* **140**, A1133 (1965)

Density Functionals

There has been considerable effort towards the development of better approximations to $E_{\mathrm{xc}}[n]$, as well as an orbital-free approximation to $T_s[n]$.

Consider $E_{\mathrm{xc}}[n]$:

In the original work of KS[1], a local density approximation was made.

Improvements have traditionally been based either on:

- Nonempirical approximations derived from quantum mechanics
- Empirical approximations containing parameters fit to improve the accuracy on particular chemical systems

While these approximations work surprisingly well, they are unable to consistently provide the high accuracy needed for many problems.

Recently, a novel approach to density-functional approximation was proposed[2], based on (conventional) machine learning.

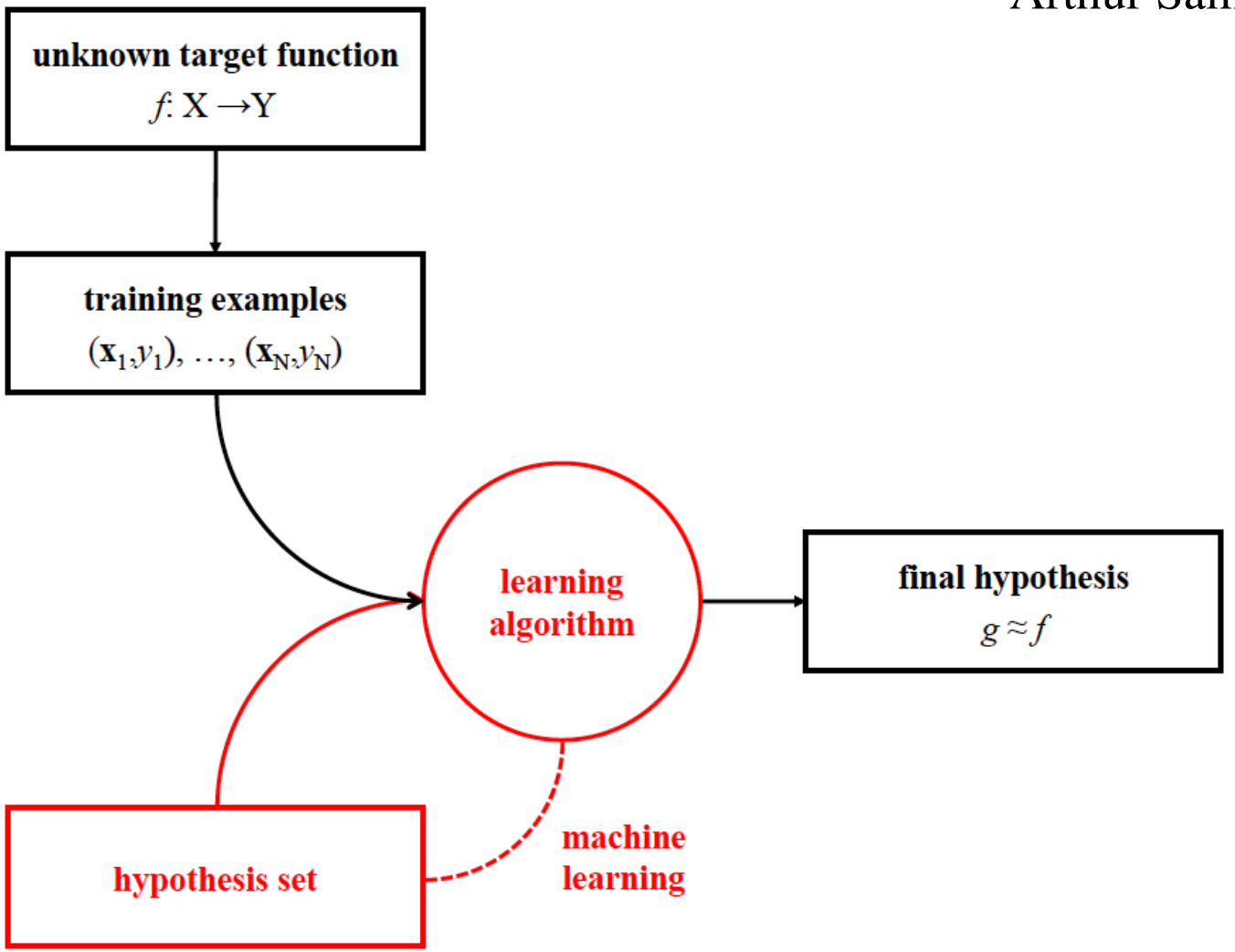
[1] W. Kohn and L. J. Sham, *Phys. Rev.* **140**, A1133 (1965)

[2] J. C. Snyder, M. Rupp, K. Hansen, K.-R. Müller, and K. Burke, *Phys. Rev. Lett.* **108**, 253002 (2012)

Machine Learning

"*[**Machine learning** is the] field of study that gives computers the ability to learn without being explicitly programmed.*"

— Arthur Samuel, 1959



Conventional Machine Learning

Conventional machine learning algorithms are very limited in their ability to process data in their natural form.

Example: They perform poorly on problems where the input–output function must be *invariant* to irrelevant variations in the input, while at the same time be very *sensitive* to others.

Invariance can be incorporated by preprocessing the data using good feature extractors. However, this requires domain expertise.

Sensitivity can be improved using nonlinear features, such as kernel methods. However, algorithms that rely solely on the smoothness prior, with a similarity between examples expressed with a local kernel, are sensitive to the variability of the target[1].

[1] Y. Bengio, O. Delalleau, and N. L. Roux,, *Advances in Neural Information Processing Systems 18* (MIT Press) (2006)

Deep Learning

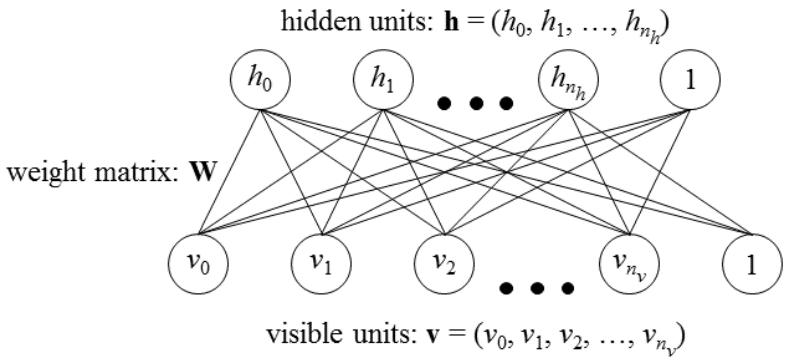
Deep learning allows computational models that are capable of discovering intricate structure in large datasets and high-dimensional data, with multiple levels of abstraction.

High-order abstractions make it easier to separate, and even extract, underlying explanatory factors in data. Such disentanglement leads to features in higher layers that are more *invariant* to some factors of variation, while at the same time being more *sensitive* to others.

Our model is based on a generative deep architecture that makes use of hidden (latent) variables (high-order features) to describe the probability distribution over (visible) data values, $p(\mathbf{v})$.

Restricted Boltzmann Machine (RBM)

Consider a **restricted Boltzmann machine (RBM)**:



Training an RBM:

$$\arg\max_{\mathbf{W}} \prod_{\mathbf{v} \in \mathbf{V}_{\text{ul}}} P(\mathbf{v})$$

$P(\mathbf{v})$: marginal probabilities of $\mathbf{v}$

$\mathbf{V}_{\text{ul}}$: set of unlabeled input data

where:

$$P(\mathbf{v}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$$

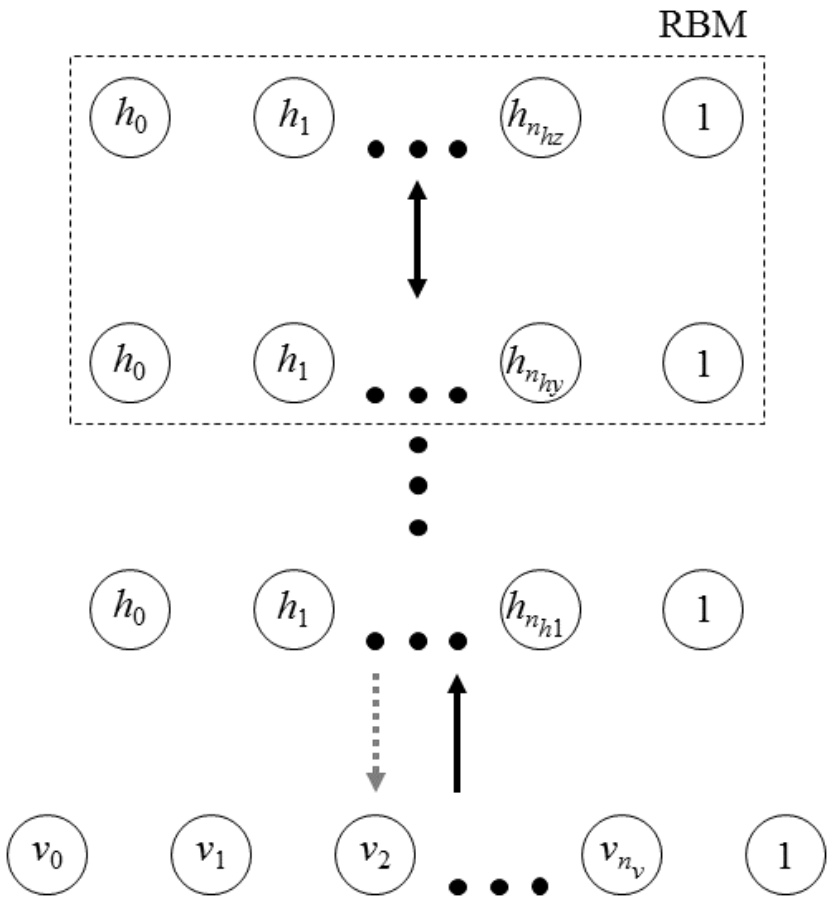
$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{a}^{\mathrm{T}}\mathbf{v} - \mathbf{b}^{\mathrm{T}}\mathbf{h} - \mathbf{v}^{\mathrm{T}}\mathbf{W}\mathbf{h}$$

$$Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$$

After training, an RBM provides a closed-form representation of $p(\mathbf{v})$.

Deep Belief Network (DBN)

RBMs can be stacked, learning successive layers of abstractions[1]. The resulting model is called a **deep belief network (DBN)**:



[1] G. E. Hinton, S. Osindero, and Y.-W. Teh, *Neural Comput.* **18**, 1527 (2006)

Mapping the Input to an Output

Following training, the DBN is used to initialize a nonlinear mapping F:

$$F : \mathbf{V} \mapsto \mathbf{Z}$$

parameterized by the weights $\mathbf{W}$ of the DBN, which maps the input vector space $\mathbf{V}$ to its feature space $\mathbf{Z}$.

Note that F is initialized in an entirely unsupervised way.

A supervised learning method is used to find a mapping from $\mathbf{Z}$ to an output y.

We considered the following probabilistic linear regression model with Gaussian noise:

$$y = f(\mathbf{z}) + \varepsilon \ , \ \ \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

where the function(al) $f(\mathbf{z})$ is distributed according to a Gaussian process (GP).

Note that this choice should be considered as one without a loss of generality.

Model System

The model system considered: N noninteracting, spinless electrons confined to a 1D box with a continuous potential. Our goal is to approximate $T_s[n]$.

Continuous potentials $v(x)$ were generated from:

$$v(x) = -\sum_{i=1}^{3} a_i \exp[-(x - b_i)^2/(2c_i^2)]$$

where a_i, b_i, and c_i were randomly selected.

The Schrödinger equation was solved numerically for $\{\phi_i\}_{i=1}^{N}$ and their corresponding energies $\{\epsilon_i\}_{i=1}^{N}$, by discretizing the domain using n_x grid points, and using Numerov's method in matrix form[2]. From these $\boldsymbol{n} = (n(x_1), n(x_2), \ldots, n(x_{n_x}))$ and $T_s[n]$ were calculated.

A dataset containing thousands of $(\boldsymbol{n}, T_s[n])$ data points was constructed.

[1] J. C. Snyder, M. Rupp, K. Hansen, K.-R. Müller, and K. Burke, *Phys. Rev. Lett.* **108**, 253002 (2012)

[2] M. Pillai, J. Goglio, and T. G. Walker, *Am. J. Phys.* **80**, 1017 (2012)

Performance Evaluation

Following training, performance was assessed by testing it on unseen data points.

Performance statistics were selected so as to give a comprehensive assessment of a given model, as well as allow a direct comparison between different ones:

- Normalized mean squared error (NMSE)[1]: amount of relative scatter, and tends not to be biased towards models that under- or overpredict:

$$\mathrm{NMSE} = \overline{(y - y_*)^2}/(\bar{y}\,\overline{y_*})$$

- Normalized mean bias factor (NMBF)[2]: amount of bias present:

$$\mathrm{NMBF} = \begin{cases} \overline{y_*}/\bar{y} - 1 & \overline{y_*} \geq \bar{y} \\ 1 - \bar{y}/\overline{y_*} & \overline{y_*} < \bar{y} \end{cases}$$

- Square of the sample correlation coefficient (r^2)[3]: proportion of variance in the input data that is accounted for:

$$r^2 = \mathrm{ss}_{yy_*}^2/(\mathrm{ss}_{yy}\mathrm{ss}_{y_*y_*})$$

where $y = T_s[n]$, y_* is the corresponding prediction, and ss are (co)variances.

[1] S. R. Hanna and D. W. Heinold, Tech. Rep. API Publication No. 4409 (American Petroleum Institute, Washington, DC) (1985)

[2] S. Yu, B. Eder, R. Dennis, S.-H. Chu, and S. E. Schwartz,, *Atmos. Sci. Lett.* **7**, 26 (2006)

[3] K. Pearson, *Proc. R. Soc. London.* **58**, 240 (1895)

Kinetic-energy Density Functional

Performance for $N = 2$ to 8:

N	NMSE ($\times 10^{-6}$)	NMBF ($\times 10^{-4}$)	r^2
2	3.1(7)	−1.6(6)	0.977(4)
3	0.34(7)	−1.0(2)	0.93(1)
4	0.035(5)	−0.06(6)	0.960(5)
5	0.0076(8)	0.15(3)	0.951(5)
6	0.0017(3)	−0.07(1)	0.959(5)
7	0.0007(1)	0.002(8)	0.948(7)
8	0.00015(2)	−0.015(4)	0.970(3)

Performance for $N = 4$, using self-consistent densities:

NMSE ($\times 10^{-6}$)	NMBF ($\times 10^{-4}$)	r^2
0.46(3)	-4.0(2)	0.81(1)

The Mapping F

The model is systematically improvable. Consider the mapping F:

Improvement in performance as the *representational power* of F is increased:

n_{h1}-n_{h2}	NMSE ($\times 10^{-6}$)	NMBF ($\times 10^{-4}$)	r^2
25–10	0.13(2)	−0.3(2)	0.87(2)
25–25	0.059(7)	−0.4(1)	0.932(8)
50–25	0.034(3)	−0.2(1)	0.962(3)
125–50	0.020(3)	−0.17(5)	0.976(3)

Improvement in performance as the *resolution* of F is increased:

M_{ul}	NMSE ($\times 10^{-6}$)	NMBF ($\times 10^{-4}$)	r^2
100	0.046(4)	−0.37(6)	0.948(4)
200	0.043(5)	−0.23(7)	0.950(6)
500	0.034(3)	−0.2(1)	0.962(3)
1000	0.028(3)	−0.24(7)	0.970(3)

The Function(al) f

The performance of the model also depends on the mapping of the high-level features to some output.

Improvement in performance as the accuracy of f is improved:

M_1	NMSE ($\times 10^{-6}$)	NMBF ($\times 10^{-4}$)	r^2
20	0.044(3)	$-0.5(1)$	0.951(3)
50	0.034(3)	$-0.2(1)$	0.962(3)
100	0.020(2)	$-0.16(4)$	0.975(3)
200	0.014(1)	$-0.10(2)$	0.983(2)

Note that remarkable accuracy can be obtained (e.g., $r^2 > 0.95$) using as few as 20 (labeled) data points.

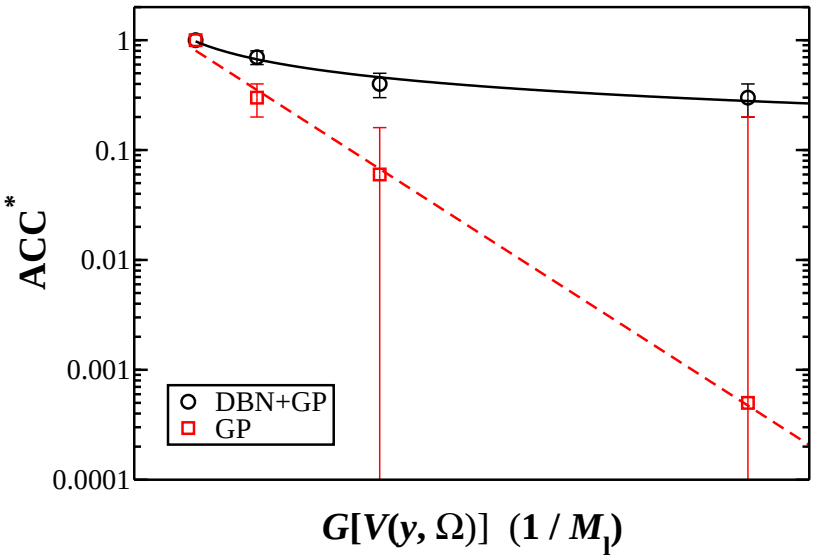
Model Efficiency

Insight into the model and its performance can be obtained by looking at its efficiency η to map its high-level features to a desired output.

$$\eta = \left(\frac{\text{ACC} \cdot G[V(y, \Omega)]}{M_1} \right) \Big/ \eta_0$$

ACC: accuracy of the model
$G[V(y, \Omega)]$: a functional of $V(y, \Omega)$, the total variation of y
M_1: amount of labeled training data
η_0: normalization factor.

Example: Normalized accuracies (ACC^*) of the (DBN+GP) model in comparison to a GP, as a function of target variability.



Advantages

The developed model offers several advantages (over conventional machine learning):

- It overcomes the invariance–sensitivity problem ...
- ... and can be initialized in an entirely unsupervised way
- It is very computationally efficient: RBM/DBN training scales linearly in both time and (storage) space with M_{ul}
- ... this means that it can make efficient use of very large data sets of unlabeled data to learn its high-level features ....
- ... while only requiring a small amount of labeled data to map them to a desired output
- Qualitative interpretations can be obtained of learned features and/or invariances

Summary and Open Questions

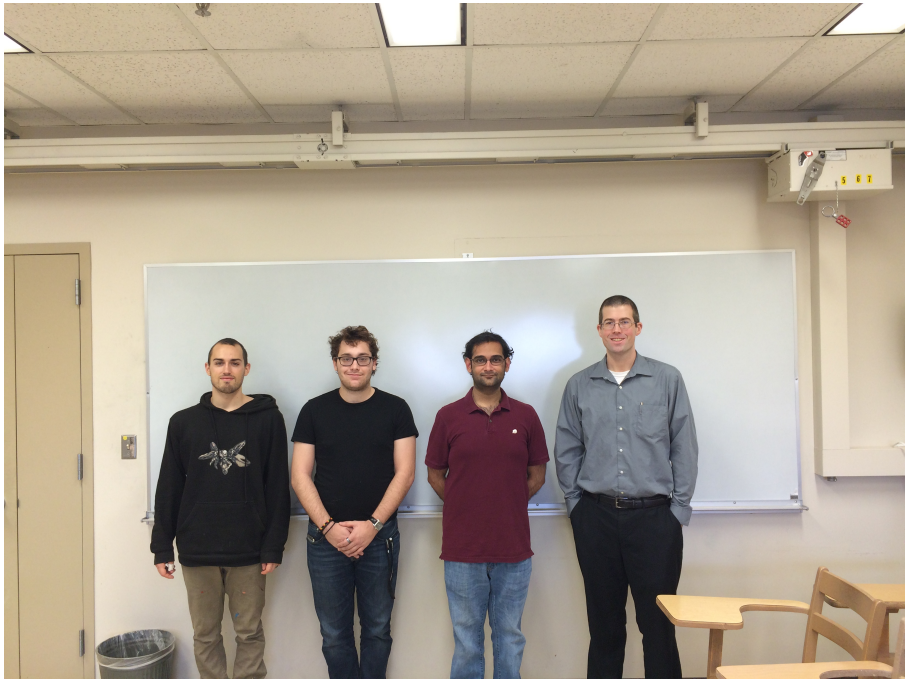
Summary

- A model based on deep learning was developed, and applied to the problem of density-functional prediction
- The model was shown to perform well on approximating $T_s[n]$ for noninteracting electrons in a 1D box
- Several advantages (over conventional machine learning) were discussed

Open Questions

- Can this approach be used in actual KS-DFT calculations? Perhaps in a self-consistent way?
- Can this approach be used in other problems for which invariance and sensitivity are needed — e.g., approximating potential-energy surfaces?

Acknowledgments



Members (left to right):

- Thomas Badman
- Nikolas Steckley
- Jeevake Attapattu
- Jeffrey M. McMahon

Start-up support:

WASHINGTON STATE UNIVERSITY

Department of Physics & Astronomy