1 short summary (08.04.20)

made by Johannes Schmidt

1.1 topic

When one tries to predict properties of molecules like **ionisation power** (IP) or **electron affinity** (EA) one might manage it by calculating the property of each individual atom in the molecule and than construct the property by merging them together, for example via a deep neural network. However, only using the atomic number as input feature is not enough. This becomes more plausible with figure 1.

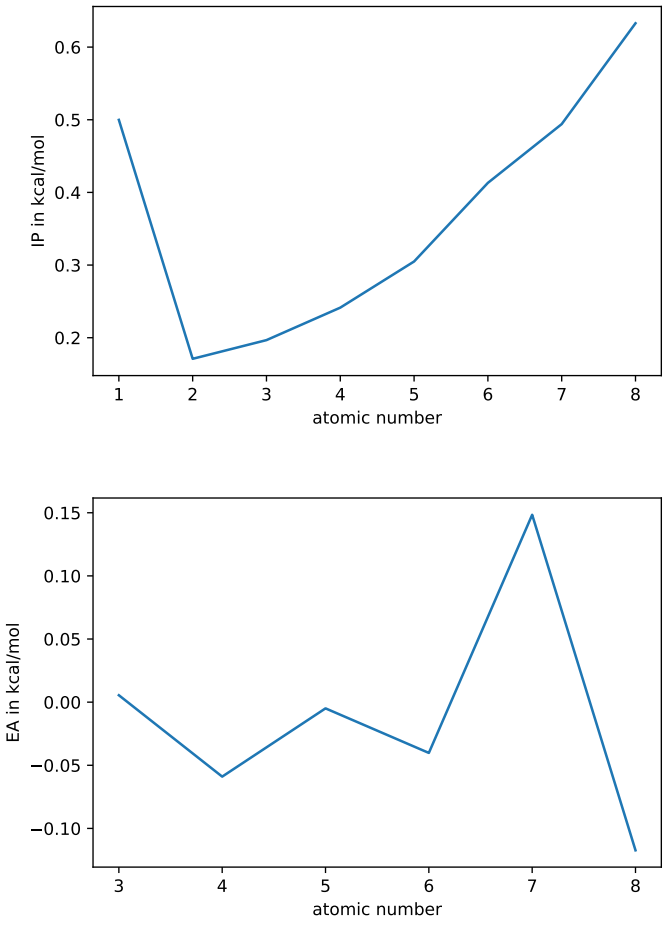

Figure 1

One can see directly that this problem is not easily described by any analytic function. One way to address this problem is to add more input features with the hope that the problem becomes more linear the more dimensions one adds. The Question now is which additional features one might use.

In principle this could be everything one knows about the atom which is related to the interested label: Atom position, charges, the type of bonding, orbital structure, or even coefficients that parameterize the orbitals.
Here we go for an approach that combines multiple of the given examples. But before we investigate in molecules, we first concentrate on single atoms.

1.2 Fitting atoms

1.2.1 The labels (y)

Here we firstly only investigate in IP. It was calculated via the Python framework **PySCF** [1]. It provides an easy way to calculate properties like IP for a given constellation of atoms. This is also true if there is only one atom, however, the property has to make sense for one atom (Atomisation energy is an example for this).
When calculating the IP for a single atom there might appear two output values, depending on the way the electrons might be in superposition with each other. In this case the smaller value was chosen.
As a basis for the calculation in PySCF the basis functions called **STO-3G** was used.

Roughly, it can be understand as three Gaussians approximating the orbital of the electron. Each of these Gaussians can be expressed as $c_i \exp(-\zeta_i x^2)$ where the c_i are called *linear* and the ζ_i are colled *non-linear* coefficients. For each element we get three of these (indicated by STO-**3**G).
For the H atom it looks like this[2]:

```
1 #BASIS SET: (3s) -> [1s]
2 H      S
3         0.3425250914E+01          0.1543289673E+00
4         0.6239137298E+00          0.5353281423E+00
5         0.1688554040E+00          0.4446345422E+00
```

One the right side the linear and one the left side the non-linear coefficients. There is also a special case if p-electrons are involved. For more details on chemical basis sets, see [1].

1.2.2 The data (x)

First of all, note that here we only investigate in the first 10 elements of the periodic table, so from H to Ne: $x\_data = [x_1, x_2, ..., x_{10}]$

[1] Documentation of PySCF: https://sunqm.github.io/pyscf
[2] Source: https://www.basissetexchange.org

For the data set the following features were chosen:

- atomic number Z_i
- orbital structure
- non-linear coefficients Ω_i (contains three numbers!)

Each data point starts with the atomic number followed by the coefficients ordered by the orbital structure.
Therefore, the i-th element looks like this:

$$x_i = [Z_i, \Omega_i([1s]_1), \Omega_i([1s]_2), \Omega_i([2s]_1), \Omega_i([2s]_2), \Omega_i([2p]_1), ..., \Omega_i([2p]_6)]$$

where $\Omega_i(x_k)$ can be understand as a tuple to three numbers, like that

$$\Omega_i(x_k) = \left(c_i^{(1)}, c_i^{(2)}, c_i^{(3)}\right)\Big|_{c=c(x_k)}$$

Here x stands for the orbital and contains the quantum numbers n and l and k numerates the electrons in that orbital.

So for the H atom it's

$$x_1 = [Z_1 = 1, \Omega_1([1s]_1), 0, 0, ...0]$$

with

$$\begin{aligned}\Omega_1([1s]_1) &= \left(c_1^{(1)}, c_1^{(2)}, c_1^{(3)}\right)\Big|_{c=c([1s]_1)} \\ &= (0.1543289673E+00,\ 0.5353281423E+00,\ 0.4446345422E+00)\end{aligned}$$

So in the end, the actual data looks like this

```
x_1 = [1,
       0.1543289673E+00, 0.5353281423E+00, 0.4446345422E+00,
       0, 0, 0,
       0, 0, 0,
       0, 0, 0,
       0, 0, 0,
       0, 0, 0,
       0, 0, 0,
       0, 0, 0,
       0, 0, 0]
```

This is done for all of the 10 elements.

1.2.3 numerical methods

To fit the modes we used the Python framework *scikit-learn* and performed a kernel ridge method. We tried some kernels but Gaussians and linear performed the best on the first view. To test the accuracy we used a leave one out test, separates one data point trains on the rest and tests on the separated point. This is done for all 10 points.

There is one parameter to tune for the training, the *alpha* parameter. As a loss for this optimization we defined a "Güte", which is defined as the average difference of the prediction and the real value in all the leave one out tests. In this way there is no prioritization of any atom, but the "Güte" is the lowest when **all Atoms** perform best on average. This value was found for alpha = 10 with linear kernels (see figure 2).

1.3 appendix

1.3.1 chemistry basics

- Here, ionisation energy is the minimum amount of energy required to remove the most loosely bound electron.
- Here, electron affinity is defined as the amount of energy released when an electron is attached to a neutral atom

References

[1] C. David Sherrill. Basis sets in quantum chemistry. http://vergil.chemistry.gatech.edu/courses/chem6485/pdf/basis-sets.pdf (called at 10.03.20).

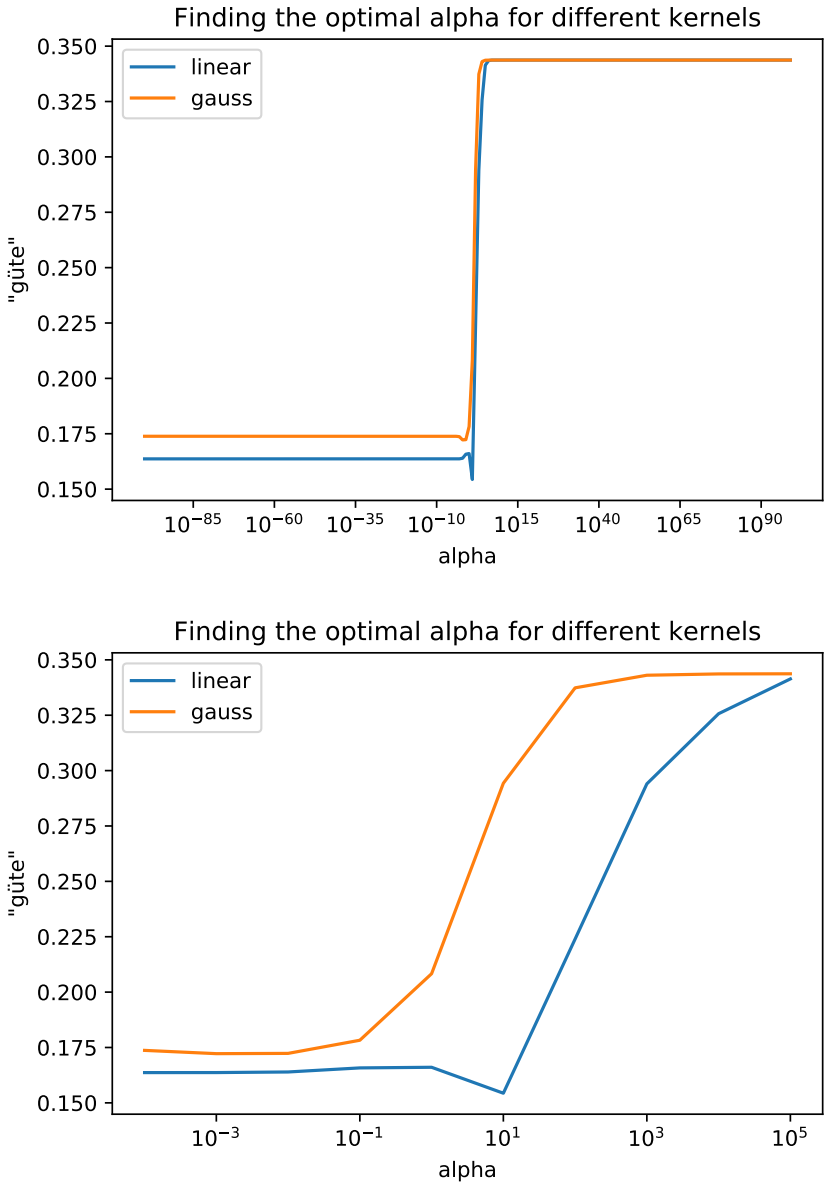
Finding the optimal alpha for different kernels
linear
gauss
"güte"
0.350
0.325
0.300
0.275
0.250
0.225
0.200
0.175
0.150
10^{-85}
10^{-60}
10^{-35}
10^{-10}
10^{15}
10^{40}
10^{65}
10^{90}
alpha
Finding the optimal alpha for different kernels
linear
gauss
"güte"
0.350
0.325
0.300
0.275
0.250
0.225
0.200
0.175
0.150
10^{-3}
10^{-1}
10^{1}
10^{3}
10^{5}
alpha

Figure 2