Machine learning for iron oxide identification from oxygen K edge in EELS spectra

Author: Marc Roset Tomàs.
Advisor: Sònia Estradé and Daniel Del Pozo Bueno
Facultat de Física, Universitat de Barcelona, Diagonal 645, 08028 Barcelona, Spain.

Abstract: In this work we test machine learning tools such as the Support Vector Machine algorithm and neural network models on the task of Electron Energy-Loss Spectroscopy (EELS) spectra classification. Given many sample spectra of EELS applied on wüstite and magnetite nanocubes, we train both models to determine the oxidation state of iron. We show that SMV exhibits a good performance on classifying clean data, and we demonstrate the capability of neural networks of producing robust results given shifted data.

I. INTRODUCTION

Electron Energy Loss Spectroscopy (EELS) [1] is a spectroscopic technique based on measuring the energy lost by the electrons that pass through a given nanomaterial. It is a very useful technique to study a whole array of properties of a material at a nanometric scale, such as its thickness, optical response, band structure and interband transitions, elemental composition, bonding and oxidation state and distribution of near neighboring atoms.

The features that are present in EELS spectra are caused by inelastic scattering events of the sample nanomaterial with incoming electrons. An EELS spectrum can be divided into a *low-loss* region (low values of energy loss, from 0 eV to about 50-100 eV) and *core-loss* region (up to several keV of energy loss). In the low-loss region we find high intensity peaks such as the *zero loss peak* and the *plasmon peak*, from which we can extract the specimen thickness and the characteristic resonance frequency of the conduction band electrons, respectively. In the core-loss region we find lower intensity features, showing the characteristic atomic transition energy, the energy-loss near edge structure (ELNES), which is the fine structure that appears in a region of a few eVs near the absorption edges of the studied material and that are related to local bonding effects; and the extended energy loss fine structure (ExELFS) for a given edge, from which we can extract the composition, the bonding and oxidation state and the distribution of near neighboring atoms, respectively. These absorption edges arise from the required energy for an inner atomic electron to jump to an excited level.

A typically characterized material in EELS experiments is transition metal oxides. Transition metals are defined as elements having a partially filled *d* sub-shell, or which can give rise to cations with an incomplete *d* sub-shell. They are located on the d-block of the periodic table. A characteristic of transition metals is that they exhibit two or more oxidation states, such as the oxidation states +2 and +3 for iron (Fe). In the case of transition metals oxides, we find in the core-loss region of its EELS spectrum the oxygen K-edge and L_2 and L_3 lines absorption edges (also known as *white lines*). As a part of the O K edge ELNES, we find a narrower intensity edge, the oxygen pre-peak.

It is useful to find the oxidation states of the transition metals in order to identify the different type of oxides, which will present different material properties. There are many approaches to the extraction of oxidation states from the EELS spectrum, such as, the calculation of the ratio of the L_3 to L_2 white lines, the measurement of the energy separation between the oxygen pre-peak and the main peak in the oxygen K-edge or the detailed analysis of the ELNES of any of the absorption edges [1, 2, 3].

Furthermore, it is important to note that EELS experiments generate large amounts of spectra, so it has become crucial to find new methods to analyse the resulting information quickly and precisely. For this reason, and to solve the problem of the determination of oxidation states, several Machine Learning (ML) tools have been proposed. One of these tools is Support Vector Machines (SVM), which were used by D. del-Pozo-Bueno et al. [4] to accurately predict the oxidation states of iron and manganese oxides from the ELNES of the white lines of the transition metals. Another ML strategy is to use neural networks as done by M. Chatzidakis et al. [5], where different neural network architectures were tested on the task of manganese oxide classification, also focusing on the energy regions of the EELS spectrum where the white lines are found.

In this work we will study the reliability and robustness of both SVMs and neural networks in the determination of the oxidation state of iron through the analysis of the ELNES of the oxygen K-edge and the oxygen pre-peak in iron oxides. We will be classifying iron oxide data corresponding to wüstite (FeO, oxidation state Fe^{+2}) and magnetite (Fe_3O_4, oxidation state $Fe^{+2}Fe_2^{+3}$).

II. METHODS

Machine learning models are mathematical models that can predict the label or class of a given object. This prediction takes place after the model has been exposed to many examples of related objects (known as the training dataset) and has extracted useful features for future predictions. ML models can be classified into supervised (the training dataset has been previously labelled either by a human or another model) and unsupervised (the training dataset is unlabelled). In this work we use two supervised learning models: SVMs and neural networks.

Our training dataset consists of 11691 iron oxide EELS spectra. Before implementing any model though, the dataset must be pre-processed so as to facilitate classification. Originally, we have a (2D) image with atomic resolution that spans across the whole nanomaterial, with an EELS spectrum associated to every pixel. First of all, the bundle of spectra is all de-noised through the PCA algorithm [6], which decomposes an input vector into many components and weeds out the low-variance ones (related to noise signal). Then 2-class K-means clustering is performed onto the image to

extract the background (in other words, the data corresponding to an electron beam that has missed or only partially interacted with the material is discarded). Given our data comes from different sample material and the calibration of the tooling is not reliable, the next step is to align the spectra with a reference peak (in our case the oxygen pre-peak located at 530 eV). We then crop the part of the spectra that we will be using (ELNES of oxygen K-edge and oxygen pre-peak) and remove a signal component derived from electrons that follow a power-law. Spikes, which are abrupt peaks caused by misfirings of the sensor's pixels, are eliminated through interpolation. The last step is normalization, and it is very important since absolute intensity is related to the sample thickness, and this is not a feature we want our model to pick on. The resulting spectra is 30 eV wide and it is fitted to a 300 channel histogram (presenting a dispersion of 0.1 eV per channel) resulting on 300-dimensional vectors. An example spectrum present on our dataset can be seen in Fig. (1).

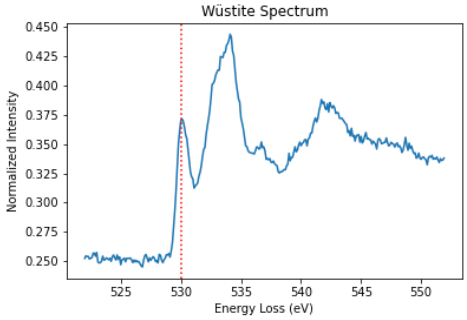

FIG 1. EELS spectrum for wüstite, cropped to the O K edge region. The red dotted line is the energy at which the spectra are aligned and where the oxygen pre-peak is found.

A Support-Vector Machine, SVM for short, [7] classifies n-dimensional datapoints through the use of an n-dimensional hyperplane that divides the feature space in halves. Therefore, it is well suited for linearly separable, binary labelled data (only 2 classes of data). The hyperplane is defined by its normal vector *w* and its bias *b*, and it must satisfy the following condition for all N x_i vectors of the training dataset x (we denote y_i as the label of the x_i datapoint).

$$\begin{cases} w \cdot x_i + b > 0 & if\ y_i = +1 \\ w \cdot x_i + b < 0 & if\ y_i = -1 \end{cases}$$

Thus, the hyperplane must accurately classify all examples from the training dataset. While there are a set of values for *w* and *b* that satisfy this condition, the idea behind SVM is to choose those values that achieve the greatest *margin*, which is defined to be the smallest distance from the hyperplane to any of the training datapoints. From an intuitive point of view, the maximization of the margin gives the smallest generalization error (in other words, it ensures accurate future predictions). It can be shown that the maximization of the margin corresponds to the minimization of ||w|| (we usually minimize $\frac{\|w\|}{2}$, which derives in the same values of w but offers a simpler expression of the solution to the optimization problem). Although it will not be shown here, the path to solving this problem is through the use of Lagrange multipliers.

Once the optimal hyperplane has been solved, the way in which we classify new data points (spectra not seen in the example dataset) is by evaluating the sign of the following expression:

$$y(x) = \sum_{i=1}^{N} a_i y_i x_i \cdot x + b,$$

where a_i are the lagrangian multipliers. Those x_i for which the corresponding a_i are non-zero will influence the outcome of the prediction and are referred to as *support vectors*.

We have worked with the SVM implementation based on the *libsvm* library [8] that can be found on the *scikit-learn* Python package [9]. It technically is an implementation of the soft-margin SVM algorithm [7], which includes a parameter *C* that is set by the user and that enables the misclassification of training datapoints in order to achieve a bigger margin.

We don't know beforehand if our data is going to be linearly separable, so it is important to introduce the concept of a kernel. Kernels are functions that map the original dataset into another feature space where the data might be linearly separable. In this work we will be using the radial basis function (RBF) and sigmoid kernels. We also refer to the linear kernel as that in which no transformation is applied. These kernels are also included on the SVM algorithm of the scikit-learn package.

Neural networks, NN for short, [7] are a variety of methods that perform well given non-linearly separable data. They can be understood as a sequence of complex operations that is built in a modular fashion using *layers*. The whole NN architecture is parametrized by a large set of *weights* but, as opposed to SVMs, the set of weights that define our NN is not found in an analytical manner.

A fully connected neural network (FCNN) [7] is that in which the main operation that is performed is matrix multiplication. In this work we have used a slightly modified version of the FCNN proposed by M. Chatzidakis et al. [5], shown in Fig. (2).

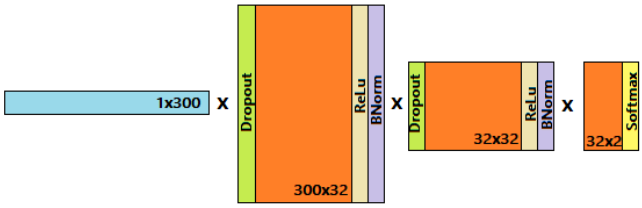

FIG 2. Representation of the FCNN architecture. The sequence of operations is from left to right, starting with a 300-dimensional spectrum and ending in class probabilities after Softmax.

The orange layers are the weight matrices, and there are some extra layers that represent different operations. The activation function ReLu is a function defined as $f(x) = \max(0, x)$ and is performed elementwise on the resulting matrix. Dropout sets random elements (with a probability set by the user) on the matrix to zero as to prevent overfitting of

the model (that is, to ensure the model doesn't overperform on the training set and underperforms on real test data). Batch normalization normalizes the layer outputs by re-centering and re-scaling. The Softmax function is the last operation performed and transforms a numerical output that can take any real value into a class probability (we would obtain a probability corresponding to Fe^{+2} oxidation state and another probability for $Fe^{+2}Fe_2^{+3}$).

On the other hand, we also have worked with a convolutional neural network (CNN) proposed by M. Chatzidakis et al. [5] and shown in Fig. (3). CNNs are a more complex form of NNs that perform well on image classification tasks. The idea behind CNNs is that weights are not arranged in a simple matrix but rather in convolutional filters (also known as kernels, not to be confused with SVM kernels), which are tensors that slide through the input performing weighted multiplications.

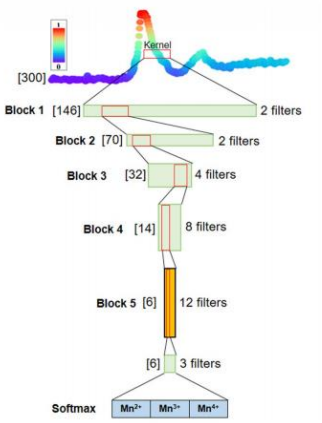

FIG. 3 The CNN architecture of M. Chatzidakis et al. [5]. Only the convolutional operations and the corresponding kernel sizes are shown, but other operations are present. In our case, the last step is modified such that only 2 class probabilities are outputted.

The ReLu, dropout, Softmax and batch normalization operations are also present in the CNN architecture, with the addition of average pooling, which takes the average value over a window with arbitrary size. This has the effect of reducing the input tensor dimension.

It is important to note that a NN architecture is arbitrary and a great deal of trial-and-error is required in order to find a good one.

While operating on the training dataset, class probabilities will be computed and compared to the correct label. We can then calculate an error or loss known as the categorical cross-entropy loss function, which gives an idea on how accurate the prediction for each class was:

$$Loss = -y_1 \cdot \log(\hat{y}_1) - y_2 \cdot \log(\hat{y}_2)$$

where y is the one-hot encoded target value (eg. y = [0, 1] if the target value is {1}, or the second possible oxidation state) and ŷ is the predicted probability for each oxidation state. Through the use of error backpropagation procedure, which enables us to calculate how each individual weight influences the final loss, we can calculate the direction of maximum error decrease in the weight space. With a certain weight update policy (also known as optimizer) the value of the weights will be changed. This will be repeated many times, and given sufficient training data, we will be able to approach a local minimum of the loss function and its corresponding weights. This weight-setting procedure is identical in both NN, and results in what we call a *trained neural network*. To implement neural networks, we have worked with the *keras* library [10].

III. RESULTS AND DISCUSSION

Given a random partition of our training dataset as 65% destined to training the model and 35% to test it, the results that we have obtained on the test dataset using an SVM can be found on Table I. As previously stated, we have worked with Lineal, RBF and Sigmoid kernels. We also have worked with a slightly different procedure of solving an SVM model, which is through Stochastic Gradient Descent (SGD). It is an optimizer used in NN training, but it can also be used to solve SVMs in a non-analytical manner.

Kernel	Best parameters	Accuracy	Training time (s)	Translation accuracy	Noise accuracy
Lineal	C = 1e5	0.94	~5	0.55	0.52
RBF	C = 2714 γ = 10	0.97	~8	0.61	0.60
Sigmoid	C = 1e5 γ = 0.1 Coef0 = 1.0	0.82	~4	0.52	0.57
SGD	-	0.8-0.9	<1	0.58	0.76

TABLE I: The best performing parameters for each kernel, the accuracies on the test dataset, the training time and the accuracies on translated and noisy data. While the SGD model is included in the table along other kernels for ease of presentation, it is important to note that it is not a kernel. The hyperplane solution for the SGD case is found in a numerical procedure similar to a NN.

The “Best Parameters” column refers to those parameters (the soft-margin SVM’s C and other parameters that are kernel-specific) that perform the best. They are found through a grid-search procedure, in which the user specifies a parameter space to be explored, and all of their combinations are used to build and test a model. An example of this is seen in Fig. (4).

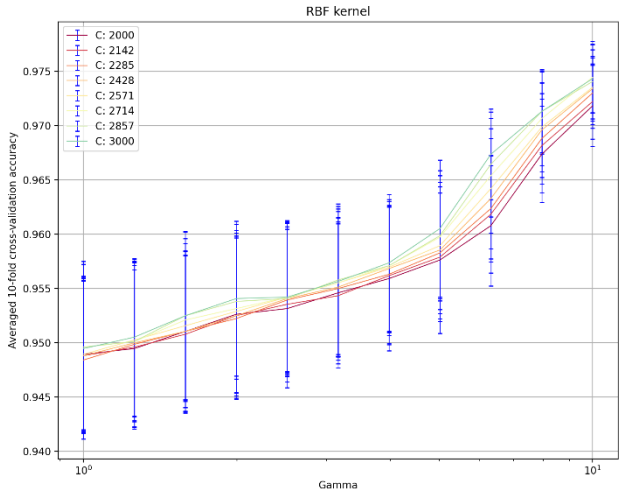

FIG 4. Grid-search for the RBF kernel. We plot the average 10 fold cross-validation accuracy, which is the average accuracy tested on 10 different train-test splitting of the original dataset. The search space consists of 8 *C* values and 11 γ values equally spaced in logarithmic space. From this graph the optimal parameters can be extracted, and are shown in Table I.

Apart from the test data accuracies, we also have tracked the training time and the accuracies given randomly shifted spectra and spectra with added gaussian noise, in order to study the models robustness. Fig. (5) illustrates a comparison between a normal spectrum and the same spectrum once we apply a translation and gaussian noise.

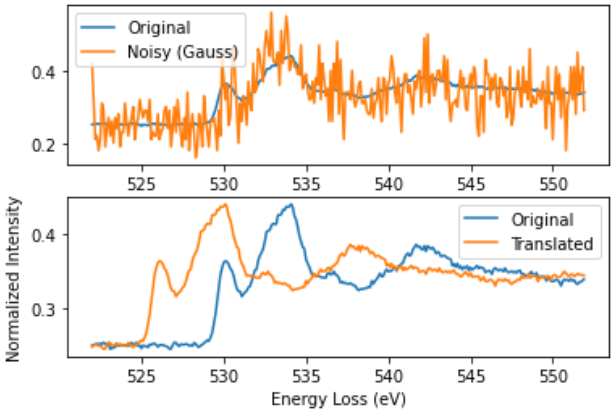

FIG 5. From top to bottom: Comparison between the original spectrum (in blue) and the noisy one (in orange). Comparison between the original spectrum (in blue) and the shifted one (in orange).

The results obtained with NN are found on Table II. FCNN (1) and CNN (1) denote the architectures as proposed by M. Chatzidakis et al. [5]. CNN (2) is the same architecture as CNN (1) but with an increased probability of neuron dropout from 0.1 to 0.5. The reasoning behind this change was the observation of a mismatch of the accuracy between training set and test set. This and other useful insights as to how to tweak the model can be extracted from looking at the evolution of training and test accuracy with respect to the epochs (number of iterations of the error minimization NN algorithm). We see an example of this in Fig. (6).

As in the SVM implementation, we have tested the accuracy in the normal test set, translated data and noisy data. We have worked with both clean, fully pre-processed data (‘Clean’) and data where PCA has not been applied (‘Raw’).

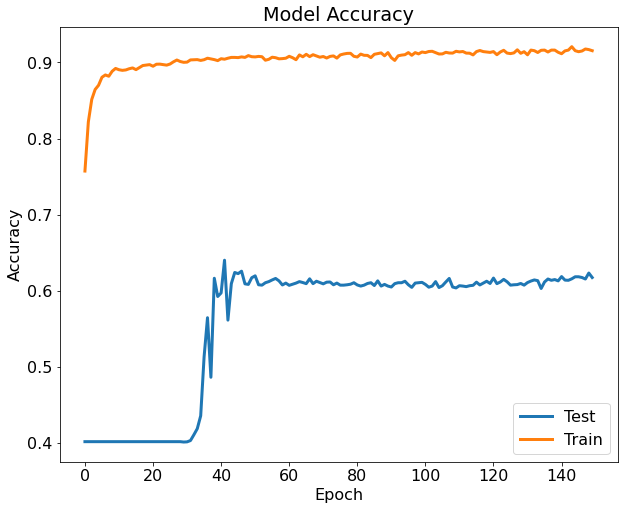

FIG 6. Train and test data accuracy with respect to epochs for the CNN (1) model. We see a rapid convergence of the accuracies, implying the optimizer (Adam, in our case) has worked properly on finding a minimum. On the other hand, while a difference on performance between the sets is to be expected, this big gap is a signal that the model might be overfitting.

We see that SVMs (in particular with the application of the RBF kernel) perform quite well on normal test data. We also notice how quickly SGD can train our model, giving a good accuracy. Nevertheless, accuracy drops considerably when we manipulate the data, as seen in the translated and noisy dataset (except for the noisy accuracy in the SGD case, which is surprisingly high). The best SVM overall is therefore the RBF kernel one.

Architecture	Training dataset	Accuracy	Translation accuracy	Noise accuracy
CNN (1)	Clean	0.62	0.77	0.49
	Raw	0.73	0.71	x
FCNN (1)	Clean	0.81	0.65	0.68
	Raw	0.70	0.59	x
CNN (2)	Clean	0.86	0.85	0.61
	Raw	0.72	0.70	x

TABLE II: The accuracy for normal test data, translated data and noisy data for the 3 models, trained in both raw and clean data. We do not believe there is any value on applying gauss noise on already noisy, raw data, so we have not tested the accuracy for this particular case.

NNs perform worse on the normal test set, but considerably outperform SVMs in the shifted dataset. This is because NNs are capable of representing much more complex functions and tend to pick on features (such as the one that gives it translation invariancy) that SVMs are not as capable of. On the other hand, NNs are harder to train, and simpler classification tasks as the one corresponding to the normal test data is better suited to simpler models. Regarding the noisy dataset, we find it underperforms in certain models, but achieves similar results in others. The best NN model overall is the CNN (2), with the FCNN (1) not too far behind, both trained on clean data (as we would expect).

IV. CONCLUSIONS

In this work we have demonstrated the capability of soft-margin support vector machines of accurately determining the oxidation state of iron from O-K edge and oxygen pre-peak data. The best performance was achieved using the radial basis function kernel. We have shown how neural networks perform great when the data is manipulated through random shifts of the spectra. The best neural network on this regard was a convolutional neural network. Finally, we have seen some difficulty on classifying noisy data from both machine learning models. In conclusion, both the soft-margin SVM and NN have proven to be powerful techniques on the task of extracting useful features from the oxygen absorption edge region of EELS spectra.

Acknowledgements

I would like to express my gratitude towards my advisors Sònia Estradé and Daniel Del Pozo Bueno for the guidance and continuous feedback during the development of this TFG. I would also like to thank my parents for the support they have always shown.

[1] R. F. Egerton, Electron energy-loss spectroscopy in the electron microscope. *Springer* 2011.cit

[2] Colliex, C.; Manoubi, T.; Ortiz, C. Electron-energy-loss-spectroscopy near-edge fine structures in the iron-oxygen system. *Phys. Rev. B*, 44, 11402–11411 (1991).

[3] Varela, M. et al. Atomic-resolution imaging of oxidation states in manganites. *Phys. Rev. B*, 79, 085117 (2009).

[4] del-Pozo-Bueno, D.; Peiró, F.; Estradé, S. Support vector machine for EELS oxidation state determination. *Ultramicroscopy*, 221, 113190 (2021).

[5] Chatzidakis, M.; Botton, G.A. Towards calibration-invariant spectroscopy using deep learning. *Sci. Rep.*, 9, 2126 (2019).

[6] Cueva, P.; Hovden, R.; Mundy, J.A.; Xin, H.L., Muller, D.A. Data processing for atomic resolution electron energy loss spectroscopy, Microsc. Microanal., 18 (4), 667–675 (2012).

[7] Christopher M. Bishop Pattern Recognition and Machine Learning. *Springer* 2006.

[8] Chang, C. C.; Lin, C. J. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2, 1-27, (2011). http://www.csie.ntu.edu.tw/~cjlin/libsvm

[9] Pedregosa et al. Scikit-learn: Machine Learning in Python. *JMLR*, 12, p2825-2830, (2011). https://scikit-learn.org/stable/

[10] Chollet, F.; et al. Keras (2015). https://keras.io