Implementation project
Lab Machine Learning for Data Science
Summer Semester 2024

G. Montavon
Institute of Computer Science
Department of Mathematics and Computer Science
Freie Universität Berlin

Lab ML for Data Science: Part III
Getting Insights into Images and their Metadata

The goal of Part III of the project is to gain insight into an image dataset and its meta-data (e.g. label annotations). A first aspect that can be of interest is whether this meta-data is preditable from the actual images, i.e. if there exists patterns in the image that can be used to determine the value of the meta-data (e.g. class membership). A second aspect of interest is to be able to identify what exact structures in the data supports these predictions, either to verify that the obtained prediction accuracy is based on meaningful features (and thus reproducible), or out of scientific interest, to acquire further knowledge on the relation between pixels and metadata. These questions will be addressed in Sections 3 and 4 respectively.

1 The Dataset

We will consider a Plant Disease dataset which contains images of leaves along with their metadata (plant type, presence of disease, and type of disease). Some examples of images taken from the dataset are given below:

The first two images are coming from healthy plants, and the last two images are from a diseased plant. The whole dataset is available for download at this page:

```
https://data.mendeley.com/datasets/tywbtsjrjv/1
```

To keep the task simple, we will limit ourselves to two classes: “apple-healthy” and “apple-black-rot”. Furthermore, to further reduce computations, you may decide to randomly sample a limited number of instances from each class, for example, 100 images per class.

2 Pretrained Models for Image Recognition

Images can be seen abstractly as a vectors in d dimensions (where d corresponds to the number of pixels times the number of color channels). On these vectors, classical methods such as linear discriminants could be in principle applied. However, there are limitations to such a direct approach. In particular, meaningful differences between images generally can not be expressed linearly in pixel space. For this reason, one rarely works directly on the pixel representation.

Instead, nonlinear feature extraction pipelines are common. Recent ones are learned on some large generic image recognition task (e.g. ImageNet classification) and take the form of pretrained neural networks. A number of pretrained networks for image classification can be found here:

```
https://pytorch.org/vision/stable/models.html
```

Among these networks, one with a rather simple structure and that has satisfactory prediction capabilities is the the VGG-16 network. We will choose this network for the subsequent experiments, specifically, the one without batch normalization. The VGG-16 network can be abstracted as a sequence of layers, where first layer receives pixel values as input and the last layer outputs scores for the classes on which the model has been trained. Top layers are task-specific and do not transfer to new tasks. However, lower layers are composed of more general features. When transferring one such network to a new task, it is therefore common to remove the top layers.

The VGG-16 network available in PyTorch, which one can store in a variable `model`, readily comes decomposed in two parts. The first part can be found in the variable `model.features`, and the top part in the variable `model.classifier`. In the following, we refer to the mapping performed by the first part of the network (i.e. `model.features`) as the function $\Phi(\boldsymbol{x})$, and our subsequent analysis will be performed on the representation at the output of that function.

3 Predicting Classes from Images

Let us now come back to our initial task, which is to determine whether meta-data (specifically the class associated to each instance) can be predicted from the corresponding image. Specifically, we consider the task of discriminating between our two classes (the black-rot-diseased and healthy apple trees).

A common model for discriminating between two classes is the *difference of means*. Let $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ be the images from our dataset, and $\mathcal{C}_1, \mathcal{C}_2$ be the set of indices corresponding to images of each class. The mean for the two classes (in feature space) can be computed as:

$$\boldsymbol{\mu}_1 = \frac{1}{|\mathcal{C}_1|} \sum_{i \in \mathcal{C}_1} \Phi(\boldsymbol{x}_i)$$

$$\boldsymbol{\mu}_2 = \frac{1}{|\mathcal{C}_2|} \sum_{i \in \mathcal{C}_2} \Phi(\boldsymbol{x}_i)$$

Based on these two mean vectors, one can identify the direction of the difference of means:

$$\boldsymbol{w} = \frac{\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1}{\|\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1\|}$$

and build a simple the discriminant function which projects the data on this direction. This allows us to score any new instance w.r.t. its predicted membership to the first or second class:

$$g(\boldsymbol{x}) = \boldsymbol{w}^\top \Phi(\boldsymbol{x})$$

When the number of dimensions is high, this discriminant is generally more robust than the equally common Fisher discriminant. The ability of our 'difference-of-means' discriminant to resolve the classes can be measured using metrics such as the "area under the ROC curve" (or AUC). The latter can be found readily implemented in scikit-learn. Note that such performance measurement should be performed on a test set disjoint from the training set (i.e. disjoint from the set of images we have used for computing the vectors $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$). Also, the number of instances that have been used for each class should be mentioned alongside the AUC score.

4 Understanding the Image-Class Relation Pixel-Wise

In practice, it is often of limited value to assess how well classes can be predicted from the images. Instead, one might ask what exact input features that are relevant for carrying these accurate predictions. There are various reasons why we would like to do this:

First, identifying relevant features enables us to validate that the high prediction accuracy is genuine and not attributable to some artefact in the image. This enables to verify that the outcome of our analysis remains reproducible on data without artefacts.

Second, and most importantly, this may provide further insights into the image-class relations. Indeed, identifying relevant input features or pixels enables us to understand what exactly in the image relates to one class or another.

4.1 Sensitivity Analysis

For generating these pixel-wise explanations, various techniques exist. The simplest of them looks at the derivative of the model w.r.t. the input pixels for a given image. The resulting derivatives can be converted to importance scores by observing that each pixel receives a 3-dimensional vector as a gradient (the RGB channels), and then computing the square norm of that vector:

$$S_i = \left\| \frac{\partial g}{\partial x_i} \right\|^2$$

The collections of pixel-wise scores $(S_i)_{i=1}^d$ can be rendered as a heatmap of same size as the original image (cf. below for an example), where important features are commonly highlighted in red. Interestingly, this analysis would reduce to analyzing the difference of mean vectors when choosing the simple feature representation $\Phi(\boldsymbol{x}) = \boldsymbol{x}$.

While the gradient in a neural network is difficult to calculate manually, machine learning frameworks such as PyTorch support automatic differentiation, thereby making the analysis very easy to implement. However, while sensitivity analysis is simple to implement, it tends to produce explanations that are noisy (cf. below) and not fully satisfactory for the user.

4.2 More Robust Explanations

There are various ways of robustifying gradient-based explanations. One of them consists of biasing the gradient in a way that prioritizes excitatory effect over inhibitory effects in the network. A simple way of generating this asymetry is to rewrite specific layers in a way that the forward function remains the same locally but the gradient is modified to implement the asymetry. For example, for any layer that can be

written generically as:

$$z_k = \sum_j a_j w_{jk} + b_k$$

(this include linear and convolutional layers), we can generate the locally functionally identical layer:

$$z_k = \left(\sum_j a_j w_{jk}^{\uparrow} + b_k^{\uparrow}\right) \cdot \left[\frac{\sum_j a_j w_{jk} + b_k}{\sum_j a_j w_{jk}^{\uparrow} + b_k^{\uparrow}}\right]_{\text{cst.}}$$

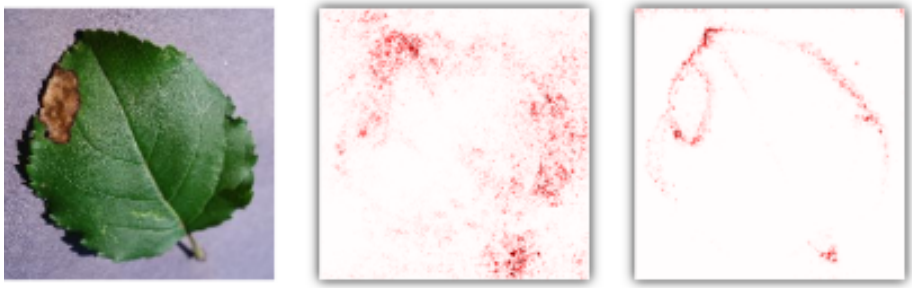
where e.g.

$$w_{jk}^{\uparrow} = w_{jk} + 0.25 \max(0, w_{jk})$$
$$b_k^{\uparrow} = b_k + 0.25 \max(0, b_k).$$

The value 0.25 is here hardcoded, but can be replaced in principle by any positive value. The notation $[\cdot]_{\text{cst.}}$ indicates to Pytorch that this part of the computation should be treated as constant (i.e. detached from the gradient), and this has the effect of modifying the gradient.

In practice, this construction can be implemented in PyTorch by defining a new class of layers (call it `BiasedLayer`) that encapsulates the original layer and a cloned version of the layer with parameters modified in the appropriate way. In the VGG-16 network, this layer-rewrite strategy can be applied to any linear/convolution layer except for the first and last one.

The effect of such modification on the outcome of sensitivity analysis is shown in the images below, where the second heatmap (corresponding to the robustified sensitivity analysis) is less noisy and more strongly focused on the actual features in the image.



4.3 Discussion

In the example above, however, we observe that pixels highlighted to be relevant do not accurately overlap with the region of the leaf where the disease is visible. As a last step, one would therefore want to explain what can be the sources of such a mismatch. For example, one may ask whether this negative result is due to an insufficiently good pretrained neural network, to a improper method for extracting relevant features, to problems with data quality, or to a flawed understanding by the human of the plant disease. After identifying what are the possible problems, suggestions should be made for overcoming the identified problems.