CHAPTER 19

# Contrastive explanations

Contrastive explanations aim to generate local explanations for neural networks of a given data input, in which, besides highlighting what is minimally sufficient (e.g., tall and long hair) in an input to justify its classification, we also want to identify contrastive characteristics or features that should be minimally and critically *absent* (e.g., glasses), so as to maintain the current classification and to distinguish it from another input that is "closest" to it but would be classified differently (e.g., Bob). We thus want to generate explanations of the form *An input $x$ is classified in class $y$ because features $f_i, \ldots, f_k$ are present and because features $f_m, \ldots, f_p$ are absent.*

There is a strong motivation to have such a form of explanations due to their presence in certain human-critical domains. In medicine and criminology, there is the notion of pertinent positives and pertinent negatives (Herman, 2016), which together constitute a complete explanation. *A pertinent positive (PP) is a factor whose presence is minimally sufficient in justifying the final classification.* On the other hand, *a pertinent negative (PN) is a factor whose absence is necessary in asserting the final classification.* For example, in medicine a patient showing symptoms of cough, cold and fever, but no sputum or chills, will most likely be diagnosed as having flu rather than having pneumonia. Cough, cold, and fever could imply both flu or pneumonia; however, the absence of sputum and chills leads to the diagnosis of flu. Thus sputum and chills are pertinent negatives, which along with the pertinent positives are critical and in some sense sufficient for an accurate diagnosis. The methods to identify such PPs and PNs can benefit from the study of generating and designing adversarial examples.

## 19.1 Contrastive explanations method

In this section, we detail the contrastive explanations method (CEM) proposed by Dhurandhar et al. (2018). Let $\mathcal{X}$ denote the feasible data space, and let $(\boldsymbol{x}_0, t_0)$ denote an example $\boldsymbol{x}_0 \in \mathcal{X}$ and its inferred class label $t_0$ obtained from a neural network model. The modified example $\boldsymbol{x} \in \mathcal{X}$ based on $\boldsymbol{x}_0$ is defined as $\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is a perturbation applied to $\boldsymbol{x}_0$. The method of finding pertinent positives/negatives is formulated as an optimization problem over the perturbation variable $\boldsymbol{\delta}$ used to explain the model prediction results. We denote the prediction of the model on the example $\boldsymbol{x}$ by $\text{Pred}(\boldsymbol{x})$, where Pred is any function that outputs a vector of prediction scores for all classes, such as prediction probabilities and logits (unnormalized probabilities) widely used in neural networks, among others.

To ensure that the modified example $\boldsymbol{x}$ is still close to the data manifold of natural examples, Dhurandhar et al. (2018) proposed to use an autoencoder to evaluate the closeness of $\boldsymbol{x}$ to the data manifold. We denote by $\text{AE}(\boldsymbol{x})$ the reconstructed example of $\boldsymbol{x}$ using the autoencoder AE.

*Finding pertinent negatives (PNs).* For pertinent negative analysis, we are interested in what is missing in the model prediction. For any natural example $\boldsymbol{x}_0$, we use the notation $\mathcal{X}/\boldsymbol{x}_0$ to denote the space of missing parts with respect to $\boldsymbol{x}_0$. We aim to find an interpretable perturbation $\boldsymbol{\delta} \in \mathcal{X}/\boldsymbol{x}_0$ to study the difference between the most probable class predictions in $\arg\max_i[\text{Pred}(\boldsymbol{x}_0)]_i$ and $\arg\max_i[\text{Pred}(\boldsymbol{x}_0 + \boldsymbol{\delta})]_i$. Given $(\boldsymbol{x}_0, t_0)$, our method finds a pertinent negative by solving the following optimization problem:

$$\min_{\boldsymbol{\delta} \in \mathcal{X}/\boldsymbol{x}_0} \; c \cdot f_\kappa^{\text{neg}}(\boldsymbol{x}_0, \boldsymbol{\delta}) + \beta \|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\delta}\|_2^2 + \gamma \|\boldsymbol{x}_0 + \boldsymbol{\delta} - \text{AE}(\boldsymbol{x}_0 + \boldsymbol{\delta})\|_2^2. \quad (19.1)$$

We elaborate on the role of each term in the objective function (19.1) as follows. The first term $f_\kappa^{\text{neg}}(\boldsymbol{x}_0, \boldsymbol{\delta})$ is a designed loss function that encourages the modified example $\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{\delta}$ to be predicted as a class different from $t_0 = \arg\max_i[\text{Pred}(\boldsymbol{x}_0)]_i$. The loss function is defined as

$$f_\kappa^{\text{neg}}(\boldsymbol{x}_0, \boldsymbol{\delta}) = \max\{[\text{Pred}(\boldsymbol{x}_0 + \boldsymbol{\delta})]_{t_0} - \max_{i \neq t_0}[\text{Pred}(\boldsymbol{x}_0 + \boldsymbol{\delta})]_i, -\kappa\}, \quad (19.2)$$

where $[\text{Pred}(\boldsymbol{x}_0 + \boldsymbol{\delta})]_i$ is the $i$th class prediction score of $\boldsymbol{x}_0 + \boldsymbol{\delta}$. The hinge-like loss function favors the modified example $\boldsymbol{x}$ to have a top-1 prediction class different from that of the original example $\boldsymbol{x}_0$. The parameter $\kappa \geq 0$ is a confidence parameter that controls the separation between $[\text{Pred}(\boldsymbol{x}_0 + \boldsymbol{\delta})]_{t_0}$ and $\max_{i \neq t_0}[\text{Pred}(\boldsymbol{x}_0 + \boldsymbol{\delta})]_i$. The second and third terms $\beta\|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\delta}\|_2^2$ in (19.1) are jointly called the elastic net regularizer, which is used for efficient feature selection in high-dimensional learning problems (Zou and Hastie, 2005). The last term $\|\boldsymbol{x}_0 + \boldsymbol{\delta} - \text{AE}(\boldsymbol{x}_0 + \boldsymbol{\delta})\|_2^2$ is an $L_2$ reconstruction error of $\boldsymbol{x}$ evaluated by the autoencoder. This is relevant provided that a well-trained autoencoder for the domain is obtainable. The parameters $c, \beta, \gamma \geq 0$ are the associated regularization coefficients.

*Finding pertinent positives (PPs).* For pertinent positive analysis, we are interested in the critical features readily present in the input. Given a natural example $\boldsymbol{x}_0$, we denote the space of its existing components by $\mathcal{X} \cap \boldsymbol{x}_0$. Here

**Algorithm 6** Contrastive Explanations Method (CEM).

```
Input: example $(x_0, t_0)$, neural network model $\mathcal{N}$, and (optionally ($\gamma > 0$)) an autoencoder $AE$
1) Solve (19.1) and obtain
$\boldsymbol{\delta}^{\text{neg}} \leftarrow \text{argmin}_{\boldsymbol{\delta} \in \mathcal{X}/\boldsymbol{x}_0}\; c \cdot f_\kappa^{\text{neg}}(\boldsymbol{x}_0, \boldsymbol{\delta}) + \beta\|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\delta}\|_2^2 + \gamma\|\boldsymbol{x}_0 + \boldsymbol{\delta} - \text{AE}(\boldsymbol{x}_0 + \boldsymbol{\delta})\|_2^2$.
2) Solve (19.3) and obtain
$\boldsymbol{\delta}^{\text{pos}} \leftarrow \text{argmin}_{\boldsymbol{\delta} \in \mathcal{X} \cap \boldsymbol{x}_0}\; c \cdot f_\kappa^{\text{pos}}(\boldsymbol{x}_0, \boldsymbol{\delta}) + \beta\|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\delta}\|_2^2 + \gamma\|\boldsymbol{\delta} - \text{AE}(\boldsymbol{\delta})\|_2^2$.
return $\boldsymbol{\delta}^{\text{pos}}$ and $\boldsymbol{\delta}^{\text{neg}}$.
## Contrastive Explanation: Input $x_0$ is classified as class $t_0$ because features $\boldsymbol{\delta}^{\text{pos}}$ are present and because features $\boldsymbol{\delta}^{\text{neg}}$ are absent.
```

we aim at finding an interpretable perturbation $\boldsymbol{\delta} \in \mathcal{X} \cap \boldsymbol{x}_0$ such that after removing it from $\boldsymbol{x}_0$, $\arg\max_i[\text{Pred}(\boldsymbol{x}_0)]_i = \arg\max_i[\text{Pred}(\boldsymbol{\delta})]_i$; that is, $\boldsymbol{x}_0$ and $\boldsymbol{\delta}$ will have the same top-1 prediction class $t_0$, indicating that the removed perturbation $\boldsymbol{\delta}$ is representative of the model prediction on $\boldsymbol{x}_0$. Similarly to finding pertinent negatives, we formulate finding pertinent positives as the following optimization problem:

$$\min_{\boldsymbol{\delta} \in \mathcal{X} \cap \boldsymbol{x}_0} \; c \cdot f_\kappa^{\text{pos}}(\boldsymbol{x}_0, \boldsymbol{\delta}) + \beta\|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\delta}\|_2^2 + \gamma\|\boldsymbol{\delta} - \text{AE}(\boldsymbol{\delta})\|_2^2, \tag{19.3}$$

where the loss function $f_\kappa^{\text{pos}}(\boldsymbol{x}_0, \boldsymbol{\delta})$ is defined as

$$f_\kappa^{\text{pos}}(\boldsymbol{x}_0, \boldsymbol{\delta}) = \max\{\max_{i \neq t_0}[\text{Pred}(\boldsymbol{\delta})]_i - [\text{Pred}(\boldsymbol{\delta})]_{t_0}, -\kappa\}. \tag{19.4}$$

In other words, for any given confidence $\kappa \geq 0$, the loss function $f_\kappa^{\text{pos}}$ is minimized when $[\text{Pred}(\boldsymbol{\delta})]_{t_0}$ is greater than $\max_{i \neq t_0}[\text{Pred}(\boldsymbol{\delta})]_i$ by at least $\kappa$.

*Algorithmic details.* A projected fast iterative shrinkage-thresholding algorithm (FISTA) (Beck and Teboulle, 2009) is applied to solve problems (19.1) and (19.3). FISTA is an efficient solver for optimization problems involving $\ell_1$ regularization. Take pertinent negative as an example, assume that $\mathcal{X} = [-1, 1]^p$ and $\mathcal{X}/\boldsymbol{x}_0 = [0, 1]^p$, and let $g(\boldsymbol{\delta}) = f_\kappa^{\text{neg}}(\boldsymbol{x}_0, \boldsymbol{\delta}) + \|\boldsymbol{\delta}\|_2^2 + \gamma\|\boldsymbol{x}_0 + \boldsymbol{\delta} - \text{AE}(\boldsymbol{x}_0 + \boldsymbol{\delta})\|_2^2$ denote the objective function of (19.1) without the $\ell_1$ regularization term. Given the initial iterate $\boldsymbol{\delta}^{(0)} = \mathbf{0}$, projected FISTA iteratively updates the perturbation $I$ times by

$$\boldsymbol{\delta}^{(k+1)} = \Pi_{[0,1]^p}\{S_\beta(\boldsymbol{y}^{(k)} - \alpha_k \nabla g(\boldsymbol{y}^{(k)}))\}, \tag{19.5}$$

$$\boldsymbol{y}^{(k+1)} = \Pi_{[0,1]^p}\{\boldsymbol{\delta}^{(k+1)} + \frac{k}{k+3}(\boldsymbol{\delta}^{(k+1)} - \boldsymbol{\delta}^{(k)})\}, \tag{19.6}$$

where $\Pi_{[0,1]^p}$ denotes the vector projection onto the set $\mathcal{X}/\boldsymbol{x}_0 = [0,1]^p$, $\alpha_k$ is the step size, $\boldsymbol{y}^{(k)}$ is a slack variable accounting for momentum acceleration with $\boldsymbol{y}^{(0)} = \boldsymbol{\delta}^{(0)}$, and $S_\beta : \mathbb{R}^p \mapsto \mathbb{R}^p$ is an elementwise shrinkage-thresholding function defined as

$$[S_\beta(\boldsymbol{z})]_i = \begin{cases} \boldsymbol{z}_i - \beta & \text{if } \boldsymbol{z}_i > \beta, \\ 0 & \text{if } |\boldsymbol{z}_i| \le \beta, \\ \boldsymbol{z}_i + \beta & \text{if } \boldsymbol{z}_i < -\beta \end{cases} \tag{19.7}$$

for $i \in \{1, \ldots, p\}$. The final perturbation $\boldsymbol{\delta}^{(k^*)}$ for pertinent negative analysis is selected from the set $\{\boldsymbol{\delta}^{(k)}\}_{k=1}^{I}$ such that $f_\kappa^{\text{neg}}(\boldsymbol{x}_0, \boldsymbol{\delta}^{(k^*)}) = 0$ and $k^* = \arg\min_{k \in \{1,\ldots,I\}} \beta \|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\delta}\|_2^2$. A similar projected FISTA optimization approach is applied to pertinent positive analysis.

Eventually, as seen in Algorithm 6, both the pertinent negative $\boldsymbol{\delta}^{\text{neg}}$ and the pertinent positive $\boldsymbol{\delta}^{\text{pos}}$ obtained from the optimization methods are used to explain the model prediction. The last term in both (19.1) and (19.3) will be included only when an accurate autoencoder is available; otherwise, $\gamma$ is set to zero.

*Model-agnostic CEM.* It is worth noting that just like how white-box adversarial attacks can be extended to black-box adversarial attacks through the use of zeroth-order optimization techniques (Chen et al., 2017a; Liu et al., 2020a), contrastive explanations can be generated in a model-agnostic fashion by finding PPs/PNs through iterative queries with a black-box model, as proposed by Dhurandhar et al. (2019).

## 19.2 Contrastive explanations with monotonic attribute functions

Luss et al. (2021) address some limitations of the original CEM proposed by Dhurandhar et al. (2018). To identify PNs, addition is easy to define for grayscale images, where a pixel with zero value indicates no information, and so increasing its value toward 1 indicates addition. However, for color images with rich structure, it is not clear what is a "no-information" value for a pixel and consequently what does we mean by addition. By rich structure we mean that there exists an interpretable latent representation for the data; all faces have a particular shape, hair has a color, and even noses can be described by their shape (e.g., pointy or not). Defining addition in a naive way such as simply increasing the pixel or red-green-blue (RGB) channel intensities can lead to uninterpretable images as the relative structure may not be maintained with the added portion not necessarily being interpretable. Moreover, even for grayscaled images, just increasing values of pixels may not lead to humanly interpretable images, nor is there a guarantee that the added portion can be interpreted even if the overall image is realistic and lies on the data manifold.

To overcome these limitations, Luss et al. (2021) define "addition" in a novel way, which leads to realistic images with the additions also being interpretable and is called contrastive explanations method using monotonic attribute functions (CEM-MAF).

Given $k$ (available or learned) interpretable features (latent or otherwise), which represent meaningful concepts (viz. moustache, glasses, smile), let $g_i$, $i \in \{1, \ldots, k\}$, be the corresponding functions acting on these features with higher values indicating the presence of a certain visual concept, whereas lower values indicating its absence. For example, CelebA (Liu et al., 2015) has different high-level (interpretable) features for each image such as whether the person has black hair or high cheekbones. In this case, we can build binary classifiers for each of the features where 1 indicates the presence of black hair or high cheekbones, whereas 0 indicates its absence. These classifiers would be the functions $g_i$. On the other hand, for datasets with no high-level features, we can find latent features by learning disentangled representations and choose those latent features that are interpretable. Here the functions $g_i$ would be the identity (or negative identity) map depending on which direction adds a certain concept (viz. light to dark colored lesion).

Let $\mathcal{X}$ denote the feasible input space with $(\boldsymbol{x}_0, t_0)$ being an example such that $\boldsymbol{x}_0 \in \mathcal{X}$ and $t_0$ is the predicted label obtained from a classifier $f$, where $f$ is any function that outputs a vector of prediction scores for all classes. To make the final image realistic, CEM-MAF first learns a data manifold, denoted $\mathcal{D}$, using a generative adversarial network (GAN) or a variational autoencoder (VAE) on which we can perturb the image, so that the final image also lies on it after the necessary additions. Let $z$ denote the latent representation with $z_x$ denoting the latent representation corresponding to input $\boldsymbol{x}$ such that $\boldsymbol{x} = \mathcal{D}(z_x)$. This gives rise to the following optimization problem for finding PNs in CEM-MAF:

$$\begin{aligned}\min_{\delta \in \mathcal{X}} \ & \gamma \sum_i \max\{g_i(\boldsymbol{x}_0) - g_i(\mathcal{D}(z_\delta)), 0\} + \beta \|g(\mathcal{D}(z_\delta))\|_1 \\ & - c \cdot \min\{\max_{i \neq t_0} [f(\boldsymbol{\delta})]_i - [f(\boldsymbol{\delta})]_{t_0}, \kappa\} + \eta \|\boldsymbol{x}_0 - \mathcal{D}(z_\delta)\|_2^2 + \nu \|z_{\boldsymbol{x}_0} - z_\delta\|_2^2. \end{aligned} \tag{19.8}$$

The first two terms in the objective function here are the novelty for PNs. The first term encourages the addition of attributes where we wants the $g_i$ for the final image to be no less than their original values. The second term encourages minimal addition of interpretable attributes. The third term is the PN loss from (Dhurandhar et al., 2018) and encourages the modified example $\boldsymbol{\delta}$ to be predicted as a class different from $t_0 = \arg\max_i [f(\boldsymbol{x}_0)]_i$, where $[f(\boldsymbol{\delta})]_i$ is the $i$th class prediction score of $\boldsymbol{\delta}$. The hinge-like loss function pushes the modified example $\boldsymbol{\delta}$ to lie in a class different from $\boldsymbol{x}_0$. The parameter $\kappa \geq 0$ is a confidence parameter that controls the separation between $[f(\boldsymbol{\delta})]_{t_0}$ and $\max_{i \neq t_0} [f(\boldsymbol{\delta})]_i$. The fourth ($\eta > 0$) and fifth ($\nu > 0$) terms encourage the final image to be close to the original image in the input and latent spaces, respectively. In practice, we could have a threshold for each of the $g_i$, where only an increase in values beyond that threshold would imply a meaningful addition. The advantage of defining addition in this manner is that not only are the final images interpretable, but so are the additions, and we can clearly elucidate which (concepts) should be necessarily absent to maintain the original classification. Finally, we note that formulation (19.1) could be equivalently written as an optimization problem over the latent space since $\delta = \mathcal{D}(z_\delta)$.

To find PPs for CEM-MAF, we want to highlight a minimal set of important pixels or superpixels (from a segmentation of input $\boldsymbol{x}_0$), which by themselves are sufficient for the classifier to output the same class as the original example. Let $\mathcal{M}$ denote a set of binary masks, which when applied to $\boldsymbol{x}_0$ produce images $\mathcal{M}(x_0)$ by selecting the corresponding superpixels from the segmentation of $\boldsymbol{x}_0$. Let $M_x$ denote the mask corresponding to the image $\boldsymbol{x} = M_x(\boldsymbol{x}_0)$ when applied on input $\boldsymbol{x}_0$. The goal for example image $\boldsymbol{x}_0$ is to find an image $\boldsymbol{\delta} \in \mathcal{M}(x_0)$ such that $\text{argmax}_i[\text{Pred}(\boldsymbol{x}_0)]_i = \text{argmax}_i[\text{Pred}(\boldsymbol{\delta})]_i$ (i.e., the same prediction) with $\boldsymbol{\delta}$ containing as few superpixels and interpretable concepts from the original image as possible. This leads to the following optimization problem:

$$\min_{\boldsymbol{\delta} \in \mathcal{M}(\boldsymbol{x}_0)} \gamma \sum_i \max\{g_i(\boldsymbol{\delta}) - g_i(\boldsymbol{x}_0), 0\} + \beta \|M_{\boldsymbol{\delta}}\|_1 - c \cdot \min\{[f(\boldsymbol{\delta})]_{t_0} - \max_{i \neq t_0} [f(\boldsymbol{\delta})]_i, \kappa\}. \tag{19.9}$$

The first term in the objective function here is the novelty for PPs and penalizes the addition of attributes since we seek a sparse explanation. The last term is the PP loss from (Dhurandhar et al., 2018) and is minimized when $[f(\boldsymbol{\delta})]_{t_0}$ is greater than $\max_{i \neq t_0} [f(\boldsymbol{\delta})]_i$ by at least $\kappa \geq 0$, which is a margin/confidence parameter. The parameters $\gamma, c, \beta \geq 0$ are the associated regularization coefficients. The optimization details for solving the PP and PN formulations (19.8) and (19.9) are discussed by Luss et al. (2021).

## 19.3 Empirical comparison

Dhurandhar et al. (2018) applied the CEM method to MNIST with a variety of examples illustrated in Fig. 19.1. The results using a convolutional autoencoder (CAE) to learn the pertinent positives and negatives are also displayed. Whereas results without CAE are quite convincing, CAE clearly improves the pertinent positives and negatives in many cases. Regarding pertinent positives, the cyan (light gray in print version) highlighted pixels in the column with CAE (CAE CEM PP) are a superset to the cyan-highlighted pixels in column without (CEM PP). Whereas these explanations are at the same level of confidence regarding the classifier, explanations using an AE are visually more interpretable. Take, for instance, the digit classified as a 2 in row 2. A small part of the tail of a 2 is used to explain the classifier without CAE, whereas the explanation using CAE has a much thicker tail and larger part of the vertical curve. In row 3, the explanation of the 3 is quite clear, but CAE highlights the same explanation but much thicker with more pixels. The same pattern holds for pertinent negatives. The horizontal line in row 4 that makes a 4 into a 9 is much more pronounced when using CAE. The change of a predicted 7 into a 9 in row 5 using the CAE is much more pronounced.

The two state-of-the-art methods used for explaining the classifier in Fig. 19.1 are LRP (Lapuschkin et al., 2016) and LIME (Ribeiro et al., 2016). LRP has a visually appealing explanation at the pixel level. Most pixels are deemed irrelevant (green (gray in print version)) to the classification (note the black background of LRP results was actually neutral). Positively relevant pixels (yellow (light gray in print version)/red (dark gray in print version)) are mostly consistent with our pertinent positives, though the pertinent positives do highlight more pixels for easier visualization. The most obvious such examples are row 3, where the yellow (light gray in print version) in LRP outlines a similar 3 to the pertinent positive, and row 6, where the yellow outlines most of what the pertinent positive provably deems necessary for the given prediction. There is little negative relevance in these examples, though we point out two interesting cases. In row 4, LRP shows that the little curve extending the upper left of the 4 slightly to the right has negative relevance (also shown by CEM as not being posi-

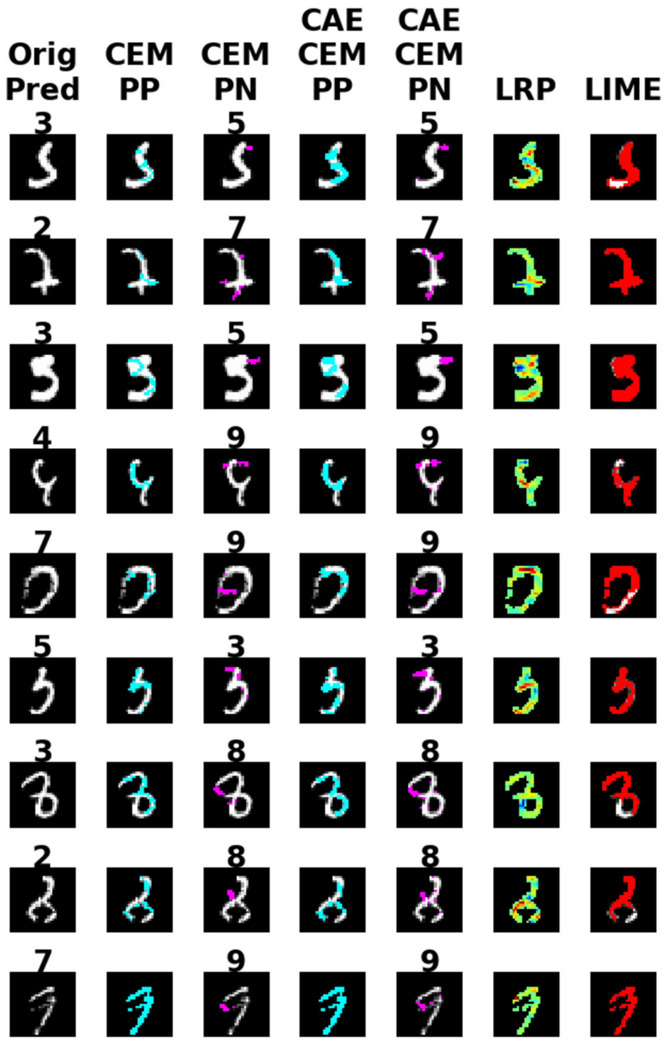

**Figure 19.1** CEM versus LRP and LIME on MNIST. PP/PN are highlighted in cyan (light gray in print version)/pink (mid gray in print version), respectively. For LRP, green (gray in print version) lightis neutral, red (dark gray in print version)/yellow (light gray in print version) is positive relevance, and blue (black in print version)is negative relevance. For LIME, red (dark gray in print version) is positive relevance, and white is neutral.

tively pertinent). Similarly, in row 3 the blue (black in print version) pixels in LRP are a part of the image that must obviously be deleted to see a clear 3. LIME is also visually appealing. However, the results are based on superpixels: the images were first segmented, and relevant segments were discovered. This explains why most of the pixels forming the digits are found relevant. Although both methods give important intuitions, neither illustrates what is necessary and sufficient about the classifier results as does our contrastive explanations method.

## 19.4 Extended reading

- Survey paper for explainability of machine learning models (Arya et al., 2019).