PROTOTEX: Explaining Model Decisions with Prototype Tensors

Anubrata Das[1*] **Chitrank Gupta**[2*] **Venelin Kovatchev**[1] **Matthew Lease**[1] **Junyi Jessy Li**[3]
[1]School of Information and [3]Dept. of Linguistics, The University of Texas at Austin
[2]Dept. of Computer Science, Indian Institute of Technology Bombay
{anubrata, venelin, ml, jessy}@utexas.edu, chigupta2011@gmail.com

Abstract

We present PROTOTEX, a novel *white-box* NLP classification architecture based on prototype networks (Li et al., 2018). PROTOTEX faithfully explains model decisions based on prototype tensors that encode latent clusters of training examples. At inference time, classification decisions are based on the distances between the input text and the prototype tensors, explained via the training examples most similar to the most influential prototypes. We also describe a novel interleaved training algorithm that effectively handles classes characterized by the *absence* of indicative features. On a propaganda detection task, PROTOTEX accuracy matches BART-large and exceeds BERT-large with the added benefit of providing faithful explanations. A user study also shows that prototype-based explanations help non-experts to better recognize propaganda in online news.

1 Introduction

Neural models for NLP have yielded significant gains in predictive accuracy across a wide range of tasks. However, these state-of-the-art models are typically less interpretable than simpler, traditional models, such as decision trees or nearest-neighbor approaches. In general, less interpretable models can be more difficult for people to use, trust, and adopt in practice. Consequently, there is growing interest in going beyond simple "black-box" model accuracy to instead design models that are both highly accurate and human-interpretable.

While much research on white-box explainable models focuses on attributing parts of the input (e.g., word sequences) to a model's prediction (Xu et al., 2015; Lei et al., 2016; Bastings et al., 2019; Jain et al., 2020; Glockner et al., 2020), there is much debate around their faithfulness and reliability (Serrano and Smith, 2019; Jain and Wallace, 2019; Wiegreffe and Pinter, 2019; Pruthi et al., 2020). Additionally, while such local explanations (if faithful) can be extremely useful in more intuitive tasks such as sentiment classification, that may not be the case for difficult tasks where human judgments may require a high degree of training or domain expertise. In such cases, understanding how models make their decisions for a particular input based on its *training data* can be insightful especially for engaging with users to develop an intuition on the model's decision making process.

In this paper, we propose **Prototype Tensor Explainability Network** (PROTOTEX)[1] to faithfully explain classification decisions in the tradition of case-based reasoning (Kolodner, 1992). Our novel *white-box* NLP architecture augments prototype classification networks (Li et al., 2018) with large-scale pretrained transformer language models. Through a novel training regime, the network learns a set of prototype tensors that encode latent clusters of training examples. At inference time, classification decisions are entirely based on similarity to prototypes. This enables model predictions to be faithfully explained based on these prototypes, *directly* via similar training examples (i.e., those most similar to top-matched prototypes). We build upon the state-of-the-art NLP neural architectures to augment their accuracy with faithful and human-interpretable explanations. Figure 1 shows an example of PROTOTEX on the task of *propaganda detection* (Da San Martino et al., 2019).

Another contribution of PROTOTEX concerns effective modeling of positive vs. negative classes in the presence of asymmetry. In a typical binary classification (e.g., sentiment detection), the presence of positive vs. negative language can be used to distinguish classes. However, with a task such as Web search, what most distinguishes relevant vs. irrelevant search results is the presence vs. absence of relevant content. Having this absence (rather than presence) of certain features most clearly dis-

*Both authors contributed equally.

[1]https://github.com/anubrata/ProtoTEx/

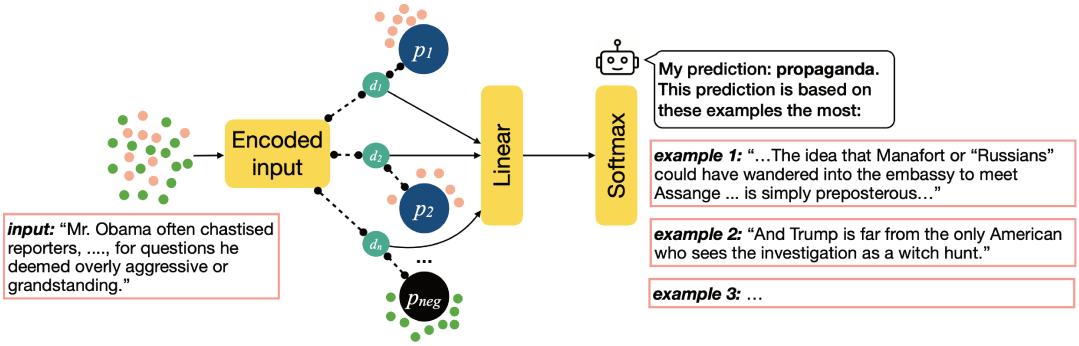


Figure 1: PROTOTEX architecture along with a use case demonstration. Pink/Green dots denote training examples, which are clustered around positive prototypes (blue) and a single negative prototype (black). Dotted lines represent distances. In this use case, the user gives PROTOTEX an input, which produces a prediction while retrieving a set of highest ranked examples that *directly* influenced model decision. In this diagram, by using "overly aggressive or grandstanding" the input creates propaganda via exaggeration (Da San Martino et al., 2019). PROTOTEX learns to identify sentences that contain propaganda phrases. In example 1, using *"Russian"* to describe Manafort (Former American political consultant) constitutes propaganda, so does using *"witch hunt"* in example 2. Exposure to similar examples helps users build an intuition towards the language used in propaganda.

tinguish a class complicates both predicting it and explaining these predictions to users. To address this, we introduce a single *negative prototype* for representing the negative class, learned via a novel training regime. We show that including this negative prototype significantly improves results.

While our model is largely agnostic to the prediction task, we evaluate PROTOTEX on a sentence-level binary propaganda detection task (Da San Martino et al., 2019). Recent work on explainable fact-checking (Kotonya and Toni, 2020a) has provided explanations via attention (Popat et al., 2018; Shu et al., 2019), rule discovery (Gad-Elrab et al., 2019), and summarization (Atanasova et al., 2020; Kotonya and Toni, 2020b,a), but not prototypes. Better explanations could enable support for human fact-checkers (Nakov et al., 2021).

We show that PROTOTEX provides faithful explanations without reducing classification accuracy, which remains comparable to the underlying encoder, BART-large (Lewis et al., 2020), superior to that of BERT-large (Devlin et al., 2019), and with the added benefit of faithful explanations in the spirit of case-based reasoning. Furthermore, to the best of our knowledge, we are the first work in NLP that examines the utility of global case-based explanations for non-expert users in model understanding and downstream task accuracy.

2 Related work

Explainable classification Unlike post-hoc analysis approaches for explainability (Ribeiro et al., 2016; Sundararajan et al., 2017), prototype classification networks (Li et al., 2018; Chen et al., 2019; Hase et al., 2019) are white-box models with explainability built-in via case-based reasoning (Kolodner, 1992) rather than extractive rationales (Lei et al., 2016; Bastings et al., 2019; Jain et al., 2020; Glockner et al., 2020). They are the neural variant of prototype classifiers (Bien and Tibshirani, 2011; Kim et al., 2014), predicting based on similar known instances. Contemporary work (Rajagopal et al., 2021) also stressed the importance of "global" explainability through training examples, yet in their approach, the similar training examples are not directly integrated in the decision itself; in contrast, we do so via learned prototypes to provide more transparency.

Our work builds on Li et al. (2018), which we lay out in Section 3.1. Later work (Chen et al., 2019; Hase et al., 2019) enables prototype learning of partial images. In NLP, Guu et al. (2018) retrieved prototype examples from the training data for edit-based natural language generation. Hase and Bansal (2020) used a variant of Chen et al. (2019)'s work to examine among other approaches; unlike our work, they used feature activation to obtain explanations similar to post-hoc approaches, and did not handle the absence of relevant content.

Evaluating explainability Explainability is a multi-faceted problem. HCI concerns include: a) For whom are we designing the explanations? b) What goals are they trying to achieve? c) How can we best convey information without imposing excessive cognitive load? and d) Can explainable

systems foster more effective human+AI partnerships (Amershi et al., 2019; Wickramasinghe et al., 2020; Wang et al., 2019; Liao et al., 2020; Wang et al., 2021; Bansal et al., 2021)? On the other hand, algorithmic concerns include generating faithful and trustworthy explanations (Jacovi and Goldberg, 2020), local vs. global explanations, and post-hoc vs. self-explanations (Danilevsky et al., 2020).

Explainability evaluation methods (Doshi-Velez and Kim, 2017) include measuring faithfulness (Jacovi and Goldberg, 2020), enabling model simulatability (Hase et al., 2019), behavioral testing (Ribeiro et al., 2020), and evaluating intelligent user interactions (Nguyen et al., 2018).

Human+AI fake news detection While explainable fact-checking (Kotonya and Toni, 2020a) could better support human-in-the-loop fact-checking (Nakov et al., 2021; Demartini et al., 2020), studies rarely assess a human+AI team in combination (Nguyen et al., 2018). In fact, human+AI teams often under-perform the human or AI working alone (Bansal et al., 2021), emphasizing the need to carefully baseline performance.

Propaganda detection (Da San Martino et al., 2019) constitutes a form of disinformation detection. Because propaganda detection is a hard task for non-expert users and state-of-the-art models are not accurate enough for practical use, explainability may promote adoption of computational propaganda detection systems (Da San Martino et al., 2021).

3 Methodology

We adopt prototype classification networks (Li et al., 2018) first proposed for vision tasks as the foundation for our prototype modeling work (Section 3.1). We design a novel interleaved training procedure, as well as a new batching process, to (a) incorporate large-scale pretrained language models, and (b) address within classification tasks where some classes can only be predicted by the *absence* of characteristics indicative of other classes.

3.1 Base architecture

PROTOTEX is based on Li et al. (2018)'s Prototype Classification Network, and we integrate pretrained language model encoders under this framework. Their architecture is based on learning *prototype tensors* that serve to represent latent clusters of similar training examples (as identified by the model). Classification is performed via a linear model that takes as an input the distances to the prototype tensors. As such, the network is a *white-box* model where global explanation is attained by *directly* linking the model to learned clusters of the training data.

Shown in Figure 1, the input is first encoded into a latent representation. This representation is fed through a prototype layer, where each unit of that layer is a learned prototype tensor that represents a cluster of training examples through loss terms $\mathcal{L}_{p1}$ and $\mathcal{L}_{p2}$ (specified by equations 2 and 3 below).

For each prototype j, the prototype layer calculates the L2 distance between its representation $\mathbf{p_j}$ and that of the input $\mathbf{x}_i$, i.e., $||\mathbf{x}_i - \mathbf{p}_j||_2^2$. The output of the prototype layer, which is a matrix of L2 distances, is then fed into a linear layer; this learns a weight matrix of dimension $K \times m$ for K classes and m prototypes, where the K weights learned for each prototype indicates that prototype's relative affinity to each of the K classes. Classification is performed via softmax.

The total loss is a weighted sum of three terms:[2]

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda_1 \mathcal{L}_{p1} + \lambda_2 \mathcal{L}_{p2} \quad (1)$$

with hyperparameter λs, standard classification cross-entropy loss $\mathcal{L}_{ce}$, and two prototype loss terms, $\mathcal{L}_{p1}$ and $\mathcal{L}_{p2}$.

$\mathcal{L}_{p1}$ minimizes avg. squared distance between each of the m prototypes and ≥ 1 encoded input:

$$\mathcal{L}_{p1} = \frac{1}{m} \sum_{j=1}^{m} \min_{i=1,n} ||\mathbf{p}_j - \mathbf{x}_i||_2^2 \quad (2)$$

encouraging each learned prototype representation to be similar to at least one training example.

$\mathcal{L}_{p2}$ encourages training examples to cluster around prototypes in the latent space by minimizing the average squared distance between every encoded input and at least one prototype:

$$\mathcal{L}_{p2} = \frac{1}{n} \sum_{i=1}^{n} \min_{j=1,m} ||\mathbf{x}_i - \mathbf{p}_j||_2^2 \quad (3)$$

Li et al. (2018) used convolutional autoencoders to represent input images. However, in the context of NLP, convolutional neural networks do not have sufficient representation power (Elbayad et al., 2018) and transformer-based language models, which are pretrained on large amounts of data, have consistently performed better in recent research. Thus to encode inputs, we experiment with

[2] In Li et al. (2018), a fourth reconstruction loss is used with their convolutional network. We found that incorporating a reconstruction loss led to unstable training, so we omit it.

Algorithm 1 Training for SIMPLEPROTOTEX.

$\mathbf{p} \coloneqq \{p_1 ... p_m\}$ ▷ prototypes
$\mathbf{x} \leftarrow \text{Encoder}(s_1, s_2, ...s_n)$ ▷ encode input sentences
Init($\mathbf{p}$)
LinearLayer← XavierInit
for k iterations **do**
  for batch $\mathbf{x}_b$ **in** Train **do**
    $d \leftarrow \text{distance}(\mathbf{x}_b, \mathbf{p})$
    $\mathcal{L}_{ce} \leftarrow \text{CE}(\textit{LinearLayer}(\text{norm}(d), \mathbf{y}_b))$
    $loss \leftarrow \mathcal{L}_{ce} + \lambda_1 \mathcal{L}_{p1}(d) + \lambda_2 \mathcal{L}_{p2}(d)$
    Update($Encoder, LinearLayer, \mathbf{p}$; $loss$)

two such models: BERT (Devlin et al., 2019) (a masked language model) and BART (Lewis et al., 2020) (a sequence-to-sequence autoencoder).

Intuition & explainability based on case-based reasoning. Because learned prototypes occupy the same space as encoded inputs, we can directly measure the distance between prototypes and encoded train or test instances. During inference time, prototypes closer to the encoded test example become more "activated", with larger weights from the prototype layer output. Consequently, model prediction is thus the weighted affinity of each prototype to the test example, where each prototype has K weights over the possible class assignments.

In the context of classification in NLP, we operationalize case-based reasoning (Kolodner, 1992) by providing similar training examples. Once the model is trained, for each prototype we rank the training examples by proximity in the latent space. During inference, we rank the prototypes by proximity to the test example. Thus, for a test example, we can obtain the training examples closest to the prototypes most influential to the classification decision. Jacovi and Goldberg (2020) define *faithfulness* as "how accurately [explanations] reflects the true reasoning process of the model." Since prototypes are directly linked to the model predictions via a linear classification layer, explanations derived by the prototypes are faithful by design. We also provide a mathematical intuition of how prototype layers relates to soft-clustering (which is inherently interpretable) in the appendix A.1.

3.2 Handling asymmetry: *negative prototype*

Section 1 noted a challenge in effectively modeling positive vs. negative classes in the presence of asymmetry. With detection tasks (e.g., finding relevant documents (Kutlu et al., 2020) or propaganda (Da San Martino et al., 2019)), the negative class may be most distinguished by the *lack* of positive features (rather than presence of negative ones). If

Algorithm 2 Decoupled training for prototypes and classification, which enables the learning of the negative prototype.

$\mathbf{p}_{\text{pos}} \coloneqq \{p_1 ... p_{m-1}\}$ ▷ prototypes for $\oplus$ class
$\mathbf{p}_{\text{neg}}$ ▷ single prototype for $\ominus$ class
$\mathbf{x} \leftarrow \text{Encoder}(s_1, s_2, ...s_n)$ ▷ encode input sentences
Init($\mathbf{p}_{\text{pos}}, \mathbf{p}_{\text{neg}}$)
LinearLayer← XavierInit
for k iterations **do**
  for $i \in 1 : \delta$ epochs **do** ▷ Minimize $\mathcal{L}_{p1}$ loss
    $c \leftarrow i \bmod 2$ ▷ pick $\{\oplus, \ominus\}$ class this iteration
    $\mathbf{p}_c \leftarrow$ prototype(s) for selected class c
    for batch $\mathbf{x}_b$ **in** Train **do**
      $\mathbf{d}_c \leftarrow \text{distance}(\mathbf{x}_{bc}, \mathbf{p}_c)$, $\mathbf{x}_{bc} \subset$ class c
      Update($\mathbf{p}_c$; $\mathcal{L}_{p1}(\text{norm}(\mathbf{d}_c))$)
  for $j \in 1 : \gamma$ epochs **do** ▷ Minimize $\mathcal{L}_{ce}$ & $\mathcal{L}_{p2}$ loss
    $c \leftarrow j \bmod 2$ ▷ pick $\{\oplus, \ominus\}$ class this iteration
    $\mathbf{p}_c \leftarrow$ prototype(s) for selected class c
    for batch $\mathbf{x}_b$ in Train **do**
      $\mathbf{d}_c \leftarrow \text{distance}(\mathbf{x}_{bc}, \mathbf{p}_c)$, $\mathbf{x}_{bc} \subset$ class c
      $\mathbf{d} \leftarrow \text{distance}(\mathbf{x}_b, \mathbf{p}_{\text{pos}}, \mathbf{p}_{\text{neg}})$
      $\mathcal{L}_{ce} \leftarrow \text{CE}(\textit{LinearLayer}(\text{norm}(\mathbf{d}), \mathbf{y}_b))$
      $loss \leftarrow \mathcal{L}_{ce} + \lambda \mathcal{L}_{p2}(\text{norm}(\mathbf{d}_c))$
      Update(*Encoder, LinearLayer; loss*)

a document is relevant only if it contains relevant content, how can one show the lack of such content? This poses a challenge both in classifying negative instances and in explaining such classification decisions on the basis of missing features.

For propaganda, Da San Martino et al. (2019) side-step the issue by only providing rationales for positive instances. For relevance, Kutlu et al. (2020) define a *negative rationale* as summarizing the instance, to succinctly show it is not germane to the positive class. However, if we conceptualize the positive class as a specific *foreground* to be distinguished from a more general *background*, such "summary" negative rationales drawn from the background distribution are likely to provide only weak, noisy evidence for the negative class.

We investigate the potential value of including or excluding a single *negative prototype* to model this "background" negative class, and design an interleaved training procedure to learn this prototype.

3.3 Training

We present two algorithms for training. The vanilla one, which we call SIMPLEPROTOTEX, does not interleave the training of positive and negative prototypes. This is illustrated in **Algorithm 1**.

One of our contributions is the design of an iterative, interleaved approach to training that balances competing loss terms, encouraging each learned prototype to be similar to at least one training example ($\mathcal{L}_{p1}$) and encouraging training examples to cluster around prototypes ($\mathcal{L}_{p2}$). We perform each

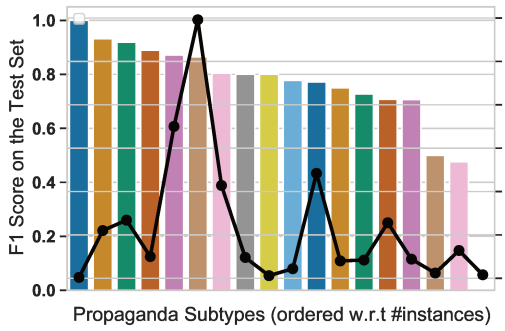


Figure 2: Macro-F1 score of PROTOTEX predicting examples that belong to each propaganda subclass. The black line corresponds to the number of examples in the test set. Classes are reordered in terms of F1.

type of representation update separately to ensure that we progressively push the prototypes and the encoded training examples closer to one another.

We illustrate this process in **Algorithm 2**. We initialize prototypes with Xavier, which allows the prototype tensors to start blind (thus unbiased) with respect to the training data and discover novel patterns or clusters on their own. After initialization, in each iteration, we first update the prototype tensors to become closer to at least one training example (henceforth δ loop). Then, in a separate training iteration, we update the representations of the training examples to push them closer to the nearest prototype tensor (henceforth γ loop). Since prototypes themselves do not have directly trainable parameters, we train the classification layer together with the encoder representations during the γ loop. We further separate the training of the positive and negative prototypes in order to push the negative "background" examples to form its own cluster. To this end, we perform class-level masking by setting the distances between the examples and prototypes of different classes to `inf`.

Finally, we perform instance normalization (Ulyanov et al., 2016) for all distances in order to achieve segregation among different prototypes (namely, the prototypes of the same class do not rely solely on a handful of examples). We discuss the effects of instance normalization in Section 4.2.

4 Experiments

Task We evaluate a binary sentence-level classification task predicting whether or not each sentence contains propaganda. We adopt Da San Martino et al. (2019)'s dataset of 21,230 sentences from news articles, with a 70/10/20 train/development/test split. Only 35.2% of sentences contain propaganda. The data is further classified into 18 fine-grained categories of propaganda; see analysis of prototypes in Section 4.2.

	Neg $\ominus$	Pos $\oplus$	Macro
Random	0.60	0.34	0.47
BERT-large	0.86	0.59	0.73
BART-large	0.86	0.65	**0.75**
KNN-BART(-large)	0.82	0.52	0.67
SIMPLEPROTOTEX	0.83	0.40	0.62
PROTOTEX (-norm)	0.82	0.56	0.69
PROTOTEX (+norm)	0.85	0.64	**0.75**

Table 1: F1 measures on the task of propaganda detection. PROTOTEX performs similar to BART.

4.1 Models and Settings

Hyperparameters are tuned on the validation data. Optimization for all neural models use AdamW (Loshchilov and Hutter, 2019) algorithm with a learning rate of 3e-5 and a batch size of 20. We use early-stopping (Fomin et al., 2020) with Macro F1 on validation data. We further perform upsampling within each batch to balance the number of examples in the positive and the negative classes.

Prototype Models PROTOTEX can be used across different underlying encoders on which interpretability components are added. Empirically, we found BART performed better on classification and so adopt it. We empirically determine the optimal number of prototypes to be 20, with one negative prototype. $\delta = 1, \lambda = 2, \gamma_1 = \gamma_2 = 0.9$. To achieve the maximum transparency, we set the bias term in the linear layer to 0 so that all information goes through the prototypes.[3] Additionally, we compare to SIMPLEPROTOTEX, which trains without use of the negative prototype.

Baselines As a strong *blackbox* benchmark we use pretrained LMs without prototypes.

BERT-large (Devlin et al., 2019): we use a simple linear layer over the output of the `CLS` token from the BERT encoder for classification.

BART-large (Lewis et al., 2020): we use the `eos` token's representation from the BART encoder as input to the linear layer of the model.

We also include a random baseline and a case-based reasoning K-Nearest-Neighbor (**KNN-BART**) baseline with the BART-large encoder.

[3] Early experiments showed no difference between including vs. excluding the bias term with instance normalization.

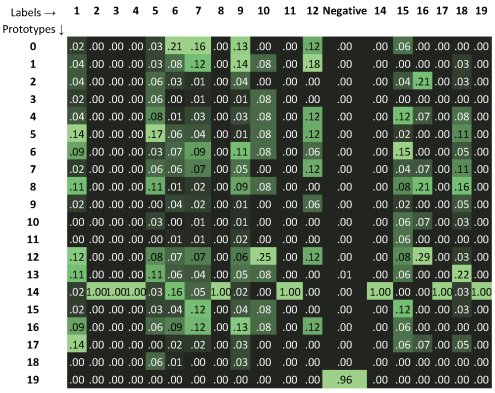


Figure 3: For each subcategory of propaganda (and the ⊖ class), the fraction of validation examples from that subcategory that are associated with each prototype; "association" defined as the closest prototype for that example. We see that PROTOTEX learns prototypes that "focusses" differently on the subcategories.

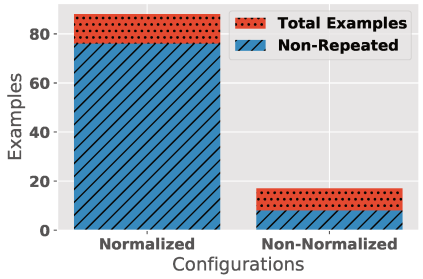


Figure 4: Number of unique 5-nearest training examples to each prototype (blue+red), and the number of examples associated with only 1 prototype (blue-only). Without normalization, very few examples (out of 100) are close to *all* prototypes; with normalization, we observe more diversity: different training examples are near different prototypes.

4.2 Classification Results

Table 1 shows F1 scores achieved by models. Among the black-box baselines, the BART-large encoder representation outperformed BERT-large significantly ($p < 0.05$, bootstrap test (Berg-Kirkpatrick et al., 2012)). PROTOTEX performed on-par with its underlying encoder BART, showing that PROTOTEX's explainability came at no cost of classification performance. It also substantially outperforms the KNN-BART baseline.

Figure 2 shows F1 for the examples, pretaining to each subclass labeled by Da San Martino et al. (2019). We can see that the model performance is relatively consistent across subclasses. The two subclasses that are most difficult for the model are "Reductio ad Hitleru" and "Appeal to Authority".

In **Figure 3**, we visualize and show that different prototypes "focus" on each subclass differently. We also see that negative examples are associated only with the negative prototype, and vice-versa.

Negative Prototype. Using a negative prototype slightly improves SIMPLEPROTOTEX results. Lacking a negative prototype, the only way to classify a negative class would be via a negative correlation on the distance between the test input and the learned prototypes. The use of the negative prototype simplifies the discriminatory process by dissociating the classification process of the negative class from the classification process of the positive class.

Instance Normalization. As shown in Table 1, normalization boosts classification performance. We also observe its benefit for explainability.

Because PROTOTEX's explainability comes from retrieving the most informative training examples, it will not be helpful for people if all prototypes are close to only a few training examples. Instead, it would be more beneficial for the prototypes to represent more subtle patterns within the training examples belonging to the same class. We refer to this phenomenon as prototype *segregation*. While the classification layer ensures that positive and negative examples (and their prototypes) are separated, it does not take into account segregation *within* the positive class. Similarly, the prototype losses $\mathcal{L}_{p1}$ and $\mathcal{L}_{p2}$ only locally ensure the closeness of examples to prototypes and vice-versa. To encourage segregation, we perform instance normalization (Ulyanov et al., 2016) for all distances.

This effect is shown in Figure 4. Specifically, we retrieve the 5 closest training examples for each of our 20 prototypes; good segregation would mean that a large portion of these examples are unique examples (the highest value is 100 meaning that all examples are unique), while bad segregation means that a large portion of these examples are the same (the lowest value is 5 meaning that all prototypes are the closest to only 5 training examples). Without normalization, we have only 17 unique examples for all 20 prototypes, yet with normalization this number is 88. Furthermore, almost all of the 88 training examples are associated with only one prototype.

5 Human Evaluation

PROTOTEX is designed to provide faithful case-

Input Sentence	True Label	Model Prediction	Similar Examples
"This scandal has set off a feeding frenzy as Internet sleuths search for other incidents in which Franken has acted inappropriately."	Propaganda	Propaganda	- "And Trump is far from the only American who sees the investigation as a witch hunt. (**Name Calling,Labeling**)" - "Do the FBI and law enforcement think people won't talk about it or speculate as to what happened? (**Doubt**)" - "And the father of Muslim spy ring Imran Awan transferred a USB drive to a Pakistani senator and former head of a Pakistani intelligence agency." (**Name Calling,Labeling**)

Table 2: Examples of similar sentences identified by our model. The input sentence uses the phrase *feeding frenzy* which is an example of propaganda phrasing. The model identifies training examples that also contain propaganda phrases as highlighted. Note that the model does not obtain the highlights shown here. Highlights are also not part of our human evaluation.

based explanations (as shown in **Table 2**) for its classification decisions. Given the set of top prototypes most influential in predicting the class for a given example, we hypothesize that these top prototypes will be representative of the example and the label corresponding to the example. We carry out two user studies to assess the utility of these prototype-based explanations for non-expert end users. Specifically, we examine whether model explanations help non-expert users to: 1) better recognize propaganda in online news; and 2) better understand model behavior.

We obtain 540 user-responses, based on 20 test-set examples, balancing gold labels and model predictions to include 5 examples from each group: true-positives, false-negatives, true-negatives, and false-positives. To simplify propaganda definitions for non-experts, we pick only four types of propaganda and we provide participants with definitions and examples for each type: *Appeal to Authority*, *Exaggeration or Minimisation*, *Loaded Language*, and *Doubt*. We select these categories because they cover the majority of the examples in the test set.

For each example, we select the top-5 prototypes that most influenced the model's prediction. We then represent each prototype by the closest training example in the embedding space. As with case-base reasoning, we explain model decisions to participants by showing for each test example the five training examples that best represent the evidence (prototypes) consulted by the model in making its prediction. Participants are primed that the model is wrong in 50% of the cases (to prevent over-trust).

5.1 Recognizing Propaganda

In this first likert-scale rating task, participants are asked whether the test example contains propaganda. Options included: definitely, probably, probably not, definitely not, or "I have no idea (completely unsure how to respond)". We compare the following four study conditions:

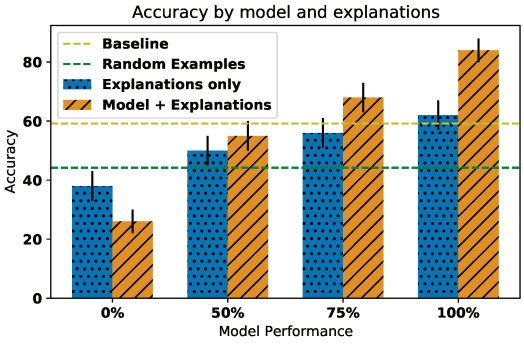


Figure 5: Accuracy of human annotations when provided with PROTOTEX explanations or PROTOTEX explanations + prediction. **Model Performance**: the accuracy of the model generating the explanations. **Baseline**: Annotation accuracy without explanations. **Random**: Randomly selected examples for explanation.

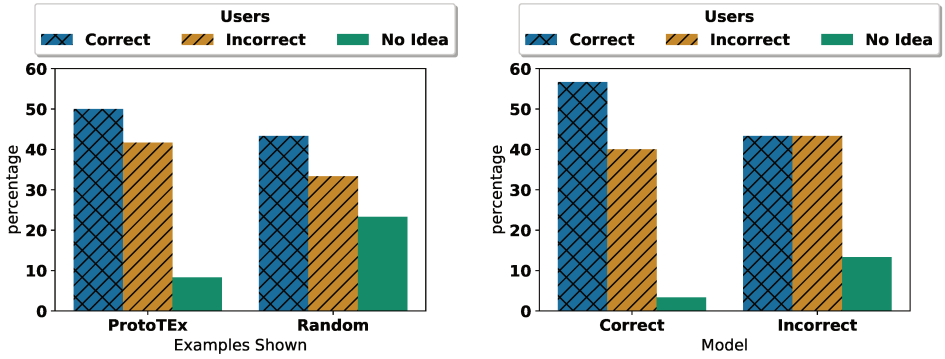
No Explanation (Baseline) We show only the test example that needs to be classified.

Random Examples We show five randomly selected training examples[4].

Explanation Only (EO) We also show five training examples, each representing a top-5 prototype influencing the model prediction, as the evidence consulted by the model in arriving at its prediction.

Model Prediction + Explanations (ME) Both the model prediction and explanations are shown.

[4]Random sampling of examples has been successfully used in tasks such as Semantic Textual Similarity (STS) and Natural Language Inference (NLI) (Agirre et al., 2013; Gold et al., 2019) to obtain a reasonable lower-bound. Comparison with random baseline demonstrates that our system selects examples that can improve human performance.



(a) (b)

Figure 6: Model Simulatability. User assessment of the model prediction **a)** Comparing PROTOTEX selected training examples vs. random examples; **b)** Comparing examples where the model prediction is accurate and examples where the model prediction is wrong

Results As Figure 5 shows, in the first baseline condition (without any additional information), participants were able to correctly predict the presence of propaganda in 59% of the cases. In the second baseline condition, when we providing random examples as "explanation", accuracy drops to 44%.

We also measure how varying model accuracy impacts the effect of model explanations, comparing four model accuracy conditions: 0% (always incorrect), 50%, 75%, and 100% (always correct)[5]. When the model is always wrong, explanations reduce the human performance below both baselines (38% in the EO condition, 26% in ME). At 50% model accuracy, human performance is higher than the "random" condition, but lower than the baseline. At 75% , the ME condition outperforms the baseline (67%). Finally, at 100% model performance both model conditions improve the accuracy of the human annotation, with ME condition reaching 84%. Our sample size of 540 exceeds the necessary 70 to holds a statistical power for between-subject studies (Bojko, 2013) .

Results from this experiment demonstrate that case-based explanations can improve human performance compared to a random baseline. However, the utility of the explanations is a function of the model accuracy.

5.2 Model Understanding

The second user task investigates model understanding by simulatability (Hase et al., 2019): can the participant predict the model decision given the most important evidence consulted by the model? Specifically, we show five training examples to the user, either **Random Examples (RE)** or PROTOTEX **Examples (PE)** (i.e., the same training examples used in the EO condition above). We ask participants to predict the model's decision using the same 5-point likert-scale as earlier.

Results Per Figure 6a, PROTOTEX's explanations help the users predict the model behavior better than random examples: 50% correct user assessment for PE vs 43.3% for RE. In 23.3% of the RE examples users are unable to make a prediction vs. 8% for the PE. Random guessing would be 40% accurate on a five-way rating task with 2 positive, 1 neutral, and 2 negative options (§5.1).

In Figure 6b we can see that the users are better at assessing the model prediction when the model is right (57%) vs when the model is wrong (43%). Additionally, we see that less users report inability to identify mode prediction when the model is correct (3.33%) vs. when the model is not (13.3%).

6 Conclusion

PROTOTEX is a novel approach to faithfully explain classification decisions by directly connecting model decisions with training examples via learned prototypes. PROTOTEX builds upon the state of the art in NLP. It integrates an underlying transformer encoder with prototype classification networks, and uses a novel, interleaving training algorithm for prototype learning. On the challenging propaganda detection task, PROTOTEX performed on-par in classification as its underlying encoder (BART-large), and exceeded BERT-large, with the

[5]We simulate desired model accuracy by post-hoc subsampling annotation instances where the model is correct/incorrect with corresponding frequency.

added benefit of providing faithful model explanations via prototypes. Our pilot human evaluation study shows that additional input provided by PROTOTEX contains relevant information for the task and can improve the annotation performance, provided sufficient model accuracy. We further demonstrate that explanations help non-expert users better understand and simulate model predictions.

Ethical Statement

For annotation, we source participants from Amazon Mechanical Turk only within the United States, paying $10/hour based on average task time. We did not reject any work but exclude data from participants who failed an attention check.

Acknowledgements

We thank the reviewers for their valuable feedback, the online workers who participated in our study and provided annotations, and the Texas Advanced Computing Center (TACC) at UT Austin for its computational resources. This research was supported in part by NSF grants IIS-1850153 and IIS-2107524, as well as by Wipro, the Knight Foundation, the Micron Foundation, and by Good Systems[6], a UT Austin Grand Challenge to develop responsible AI technologies. The statements made herein are solely the opinions of the authors and do not reflect the views of the sponsoring agencies.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. 2019. Guidelines for human-ai interaction. In *Proceedings of the 2019 chi conference on human factors in computing systems*, pages 1–13.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. Generating fact checking explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7352–7364, Online. Association for Computational Linguistics.

[6] http://goodsystems.utexas.edu/

Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. 2021. Does the whole exceed its parts? the effect of ai explanations on complementary team performance. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–16.

Jasmijn Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2963–2977, Florence, Italy. Association for Computational Linguistics.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005, Jeju Island, Korea. Association for Computational Linguistics.

Jacob Bien and Robert Tibshirani. 2011. Prototype selection for interpretable classification. *The Annals of Applied Statistics*, pages 2403–2424.

Agnieszka Bojko. 2013. *Eye Tracking the User Experience: A Practical Guide to Research*.

Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. 2019. This looks like that: Deep learning for interpretable image recognition. *Advances in Neural Information Processing Systems*, 32:8930–8941.

Giovanni Da San Martino, Stefano Cresci, Alberto Barrón-Cedeño, Seunghak Yu, Roberto Di Pietro, and Preslav Nakov. 2021. A survey on computational propaganda detection. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 4826–4832.

Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news article. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5636–5646, Hong Kong, China. Association for Computational Linguistics.

Marina Danilevsky, Kun Qian, Ranit Aharonov, Yannis Katsis, Ban Kawas, and Prithviraj Sen. 2020. A survey of the state of explainable AI for natural language processing. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 447–459, Suzhou, China. Association for Computational Linguistics.

Gianluca Demartini, Stefano Mizzaro, and Damiano Spina. 2020. Human-in-the-loop artificial intelligence for fighting online misinformation: Challenges and opportunities. *The Bulletin of the Technical Committee on Data Engineering*, 43(3).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Maha Elbayad, Laurent Besacier, and Jakob Verbeek. 2018. Pervasive attention: 2D convolutional neural networks for sequence-to-sequence prediction. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 97–107, Brussels, Belgium. Association for Computational Linguistics.

V. Fomin, J. Anmol, S. Desroziers, J. Kriss, and A. Tejani. 2020. High-level library to help with training neural networks in pytorch. https://github.com/pytorch/ignite.

Mohamed H. Gad-Elrab, Daria Stepanova, Jacopo Urbani, and Gerhard Weikum. 2019. Exfakt: A framework for explaining facts over knowledge graphs and text. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, page 87–95, New York, NY, USA. Association for Computing Machinery.

Max Glockner, Ivan Habernal, and Iryna Gurevych. 2020. Why do you think that? exploring faithful sentence-level rationales without supervision. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1080–1095, Online. Association for Computational Linguistics.

Darina Gold, Venelin Kovatchev, and Torsten Zesch. 2019. Annotating and analyzing the interactions between meaning relations. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 26–36, Florence, Italy. Association for Computational Linguistics.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

Peter Hase and Mohit Bansal. 2020. Evaluating explainable AI: Which algorithmic explanations help users predict model behavior? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5540–5552, Online. Association for Computational Linguistics.

Peter Hase, Chaofan Chen, Oscar Li, and Cynthia Rudin. 2019. Interpretable image recognition with hierarchical prototypes. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 7, pages 32–40.

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online. Association for Computational Linguistics.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Sarthak Jain, Sarah Wiegreffe, Yuval Pinter, and Byron C. Wallace. 2020. Learning to faithfully rationalize by construction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4459–4473, Online. Association for Computational Linguistics.

Been Kim, Cynthia Rudin, and Julie A Shah. 2014. The bayesian case model: A generative approach for case-based reasoning and prototype classification. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Janet L Kolodner. 1992. An introduction to case-based reasoning. *Artificial intelligence review*, 6(1):3–34.

Neema Kotonya and Francesca Toni. 2020a. Explainable automated fact-checking: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5430–5443, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Neema Kotonya and Francesca Toni. 2020b. Explainable automated fact-checking for public health claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7740–7754, Online. Association for Computational Linguistics.

Mucahid Kutlu, Tyler McDonnell, Tamer Elsayed, and Matthew Lease. 2020. Annotator Rationales for Labeling Tasks in Crowdsourcing. *Journal of Artificial Intelligence Research (JAIR)*, 69:143–189.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training

for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. 2018. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI Press.

Q Vera Liao, Daniel Gruen, and Sarah Miller. 2020. Questioning the AI: informing design practices for explainable ai user experiences. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–15.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Preslav Nakov, D. Corney, Maram Hasanain, Firoj Alam, Tamer Elsayed, A. Barr'on-Cedeno, Paolo Papotti, Shaden Shaar, and Giovanni Da San Martino. 2021. Automated fact-checking for assisting human fact-checkers. In *IJCAI*.

An T Nguyen, Aditya Kharosekar, Saumyaa Krishnan, Siddhesh Krishnan, Elizabeth Tate, Byron C Wallace, and Matthew Lease. 2018. Believe it or not: Designing a human-ai partnership for mixed-initiative fact-checking. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, pages 189–199.

Kashyap Popat, Subhabrata Mukherjee, Andrew Yates, and Gerhard Weikum. 2018. DeClarE: Debunking fake news and false claims using evidence-aware deep learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 22–32, Brussels, Belgium. Association for Computational Linguistics.

Danish Pruthi, Mansi Gupta, Bhuwan Dhingra, Graham Neubig, and Zachary C. Lipton. 2020. Learning to deceive with attention-based explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4782–4793, Online. Association for Computational Linguistics.

Dheeraj Rajagopal, Vidhisha Balachandran, Eduard H Hovy, and Yulia Tsvetkov. 2021. SELFEXPLAIN: A self-explaining architecture for neural text classifiers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 836–850, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Sofia Serrano and Noah A. Smith. 2019. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy. Association for Computational Linguistics.

Kai Shu, Limeng Cui, Suhang Wang, Dongwon Lee, and Huan Liu. 2019. defend: Explainable fake news detection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 395–405.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2016. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

Danding Wang, Qian Yang, Ashraf Abdul, and Brian Y Lim. 2019. Designing theory-driven user-centric explainable AI. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pages 1–15.

Zijie J. Wang, Dongjin Choi, Shenyu Xu, and Diyi Yang. 2021. Putting humans in the natural language processing loop: A survey. In *Proceedings of the First Workshop on Bridging Human–Computer Interaction and Natural Language Processing*, pages 47–52, Online. Association for Computational Linguistics.

Chathurika S Wickramasinghe, Daniel L Marino, Javier Grandio, and Milos Manic. 2020. Trustworthy AI development guidelines for human system interaction. In *2020 13th International Conference on Human System Interaction (HSI)*, pages 130–136. IEEE.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR.

A Appendix

A.1 Prototypes as Soft-Clustering

We provide more insights into prototypes by illustrating how the prototype layer relates to soft-clustering .

Let $t_{1:n}$ denote n training examples, having binary labels $y_{1:n}$. Let $D(a, b)$ denote symmetric distance of any two training examples a and b. Assume m additional *prototypes* (i.e., points) $p_{1:m}$ are defined in the same space as the training examples. Then $D(a, b)$ can also be computed between any training example and prototype, or between any two prototypes. Let $d_i^j = D(p_j, t_i)$ denote the symmetric distance between p_j and training example t_i. Then any two training examples, t_u and t_v, will have respective distances d_u^j and d_v^j to prototype p_j.

Let $P_j = \pi_{1:n}^j$ denote a probability distribution for prototype p_j over the training examples $t_{1:n}$. Specifically, induce π_i^j for training example t_i as a function of its distance d_i^j from prototype p_j: $\pi_i^j = z_j/d_i^j$, where z_j is a normalization constant. Then the relative probabilities for two training examples t_v and t_u are $\pi_v^j/\pi_u^j = \frac{z_j/d_v^j}{z_j/d_u^j} = d_u^j/d_v^j$. By total probability, $1 = \sum_i \pi_i^j = \sum_i z_j/d_i^j = z_j \sum_i 1/d_i^j$, so $z_j = \frac{1}{\sum_i 1/d_i^j}$. Based on this, we can say that each prototype effectively denotes a *soft-clustering* over the set of training examples.

Further, the ratio of distances (d_u^j / d_v^j) between training examples t_u and t_v and prototype p_j, is the reciprocal of their probabilities: π_v^j/π_u^j. In other words, if a training example t_u is twice as far away from prototype p_j as another training example t_v (i.e., d_u^j / $d_v^j = 2$), then t_v will be twice as probable as t_u in probability distribution P_j (i.e., $\pi_v^j/\pi_u^j = 2$).

Inference. The inference calculation shown here uses only the prototype layer. $P_j(y = 1) = \psi_j = \sum_{1:n} \pi_i^j y_i$ denote the relative frequency estimated probability of prototype p_j having true class label $y = 1$. Let x denote a test example (defined in the same vector space as training examples and prototypes). Then $D(x, p_j) = d_x^j$ defines the symmetric distance between x and prototype p_j. Let $\Theta_x = \theta_{1:m}^x$ denote a probability distribution for test example x over the prototypes $p_{1:m}$. As with training examples and prototypes above, induce this probability distribution based on relative distances between x and each prototype p_j. Then similar to before, if $d_x^j/d_x^k = 2$, meaning prototype p_j is twice as far from x as prototype p_k, then we have $\theta_k^x/\theta_j^x = 2$ meaning p_k will be twice as probable as p_j in probability distribution Θ_x. Class label $y = 1$ for test example x is predicted by probability $\Theta_x(y = 1) = \sum_{1:m} \theta_j^x \psi_j = \sum_{j \in 1:m} \theta_j^x \sum_{i \in 1:n} \pi_i^j y_i$.