XPROAX – Local explanations for text classification with progressive neighborhood approximation

Yi Cai[1], **Arthur Zimek**[2], **Eirini Ntoutsi**[1,3]

[1]L3S Research Center, Hannover, Germany

[2]Department of Mathematics and Computer Science (IMADA), University of Southern Denmark (SDU), Odense, Denmark

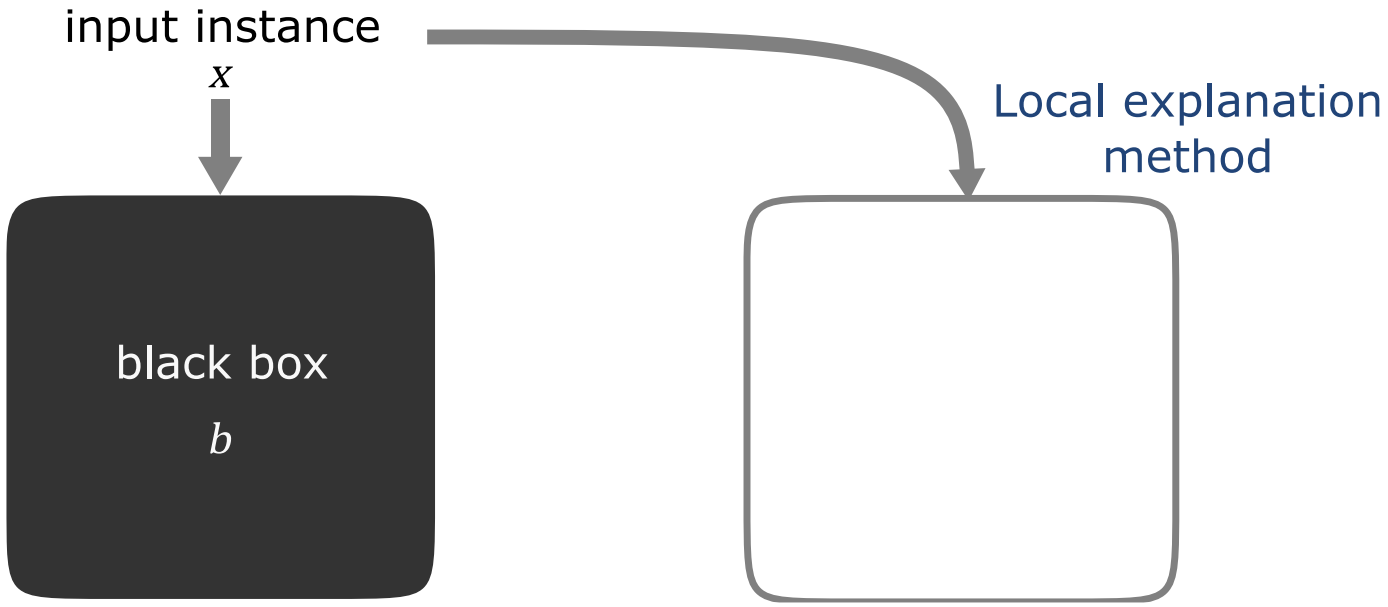
[3]Department of Mathematics and Computer Science, Freie Universität Berlin, Berlin, Germany

IEEE DSAA, 2021

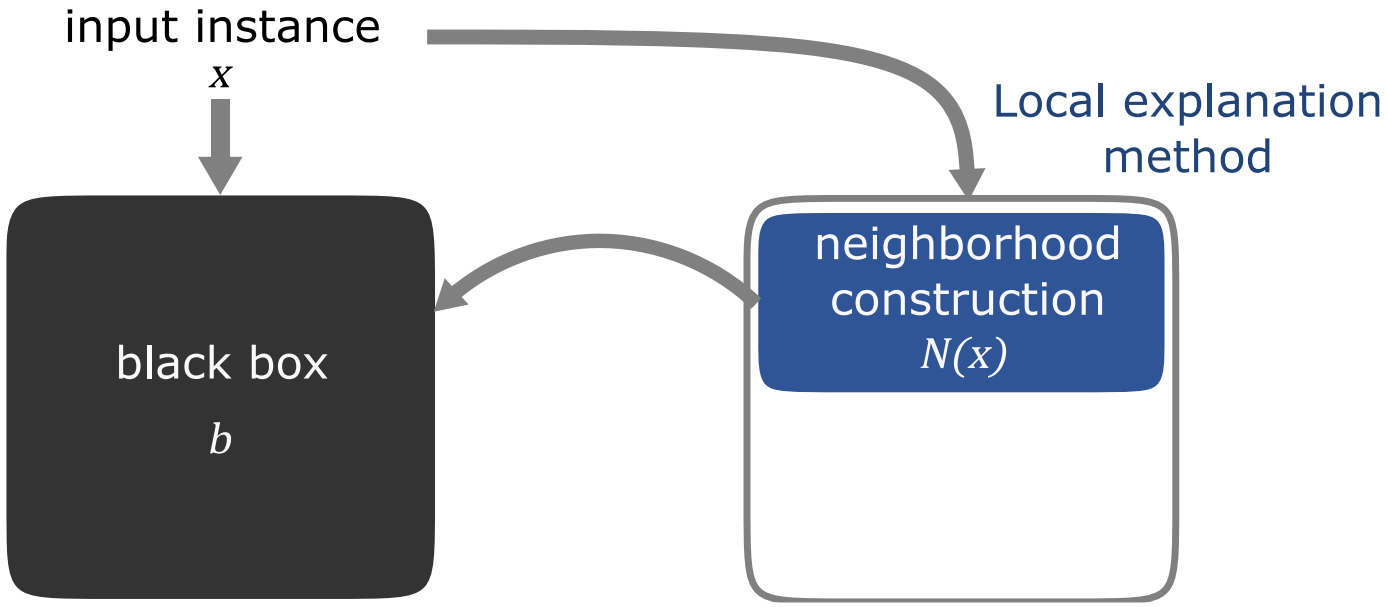
This work is supported by the State Ministry of Science and Culture of Lower Saxony, within the PhD program "Responsible Artificial Intelligence in the Digital Society".

Niedersächsisches Ministerium für Wissenschaft und Kultur

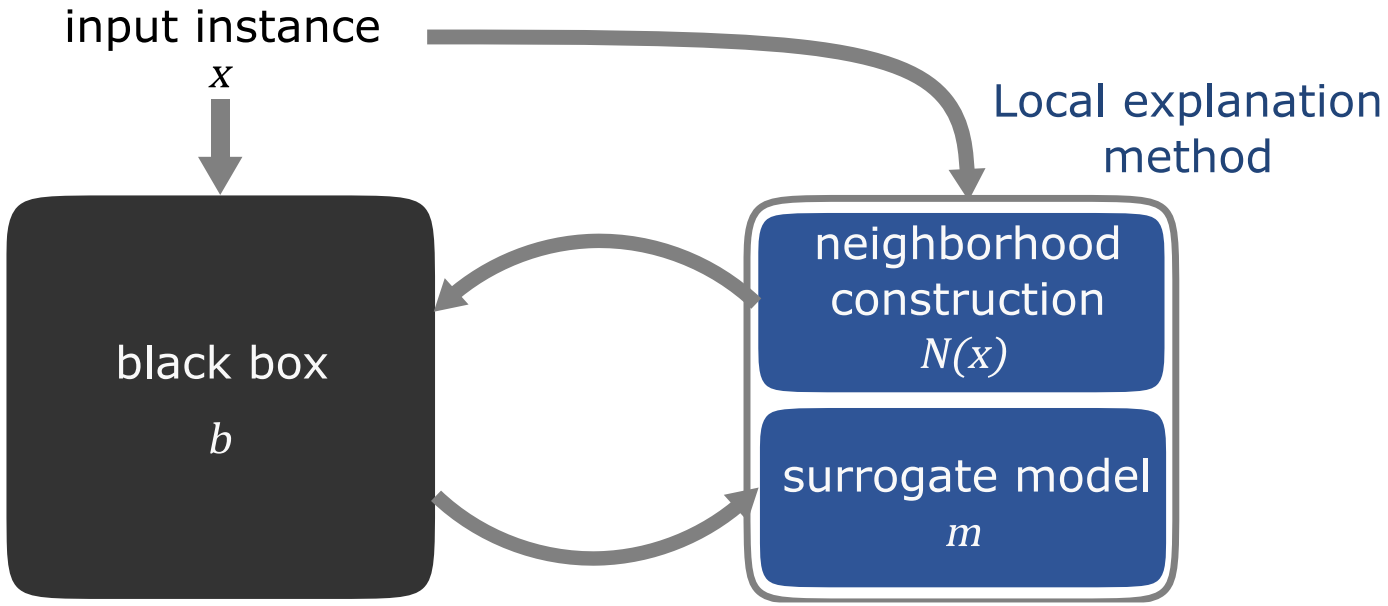
Local explanation method



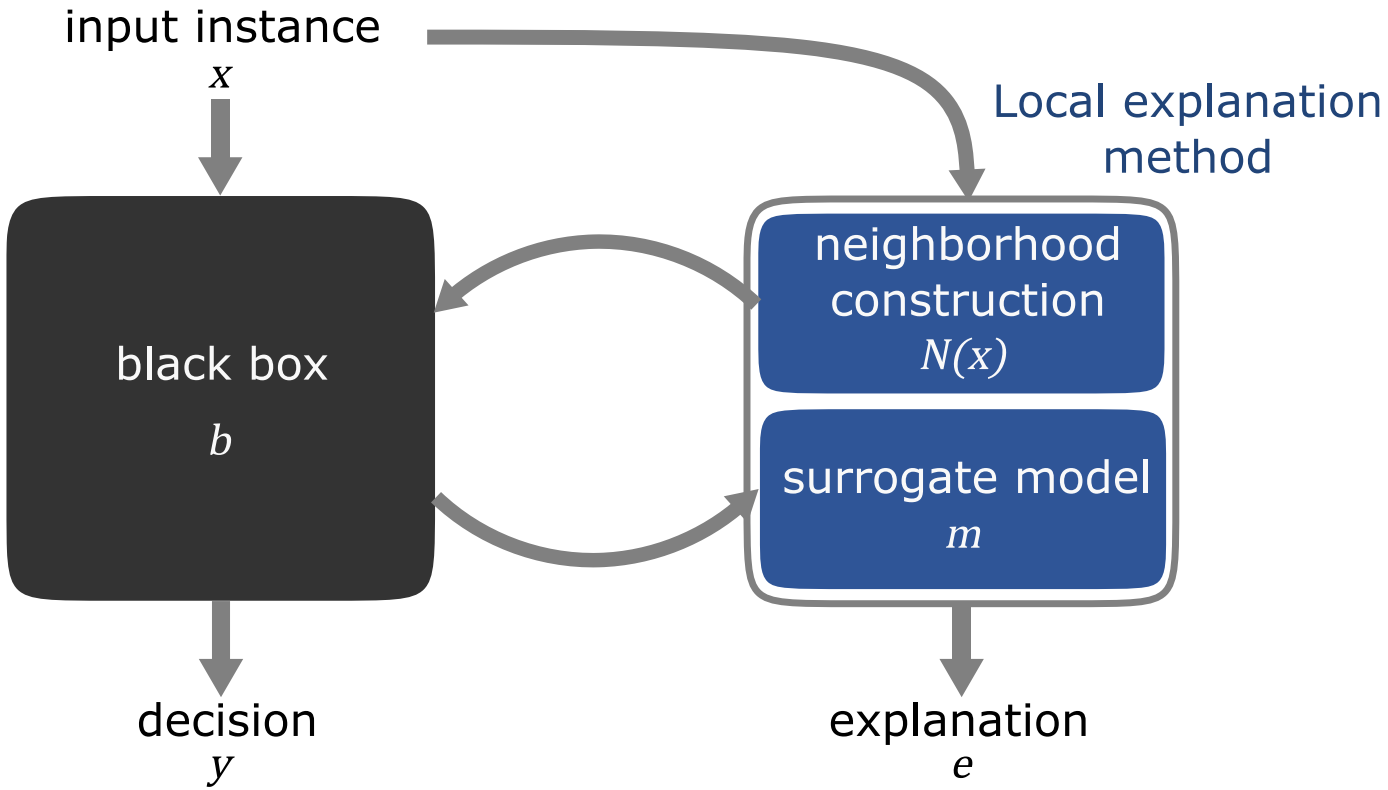
Local explanation method



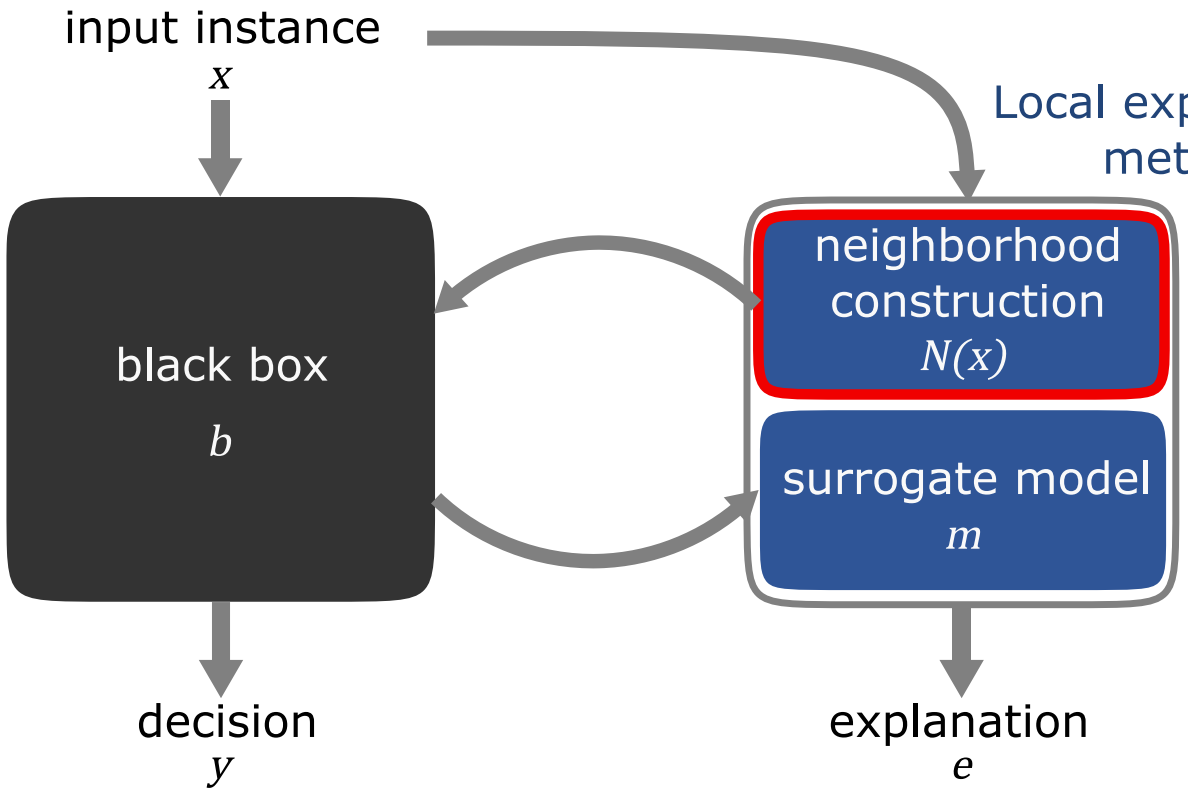
Local explanation method



Local explanation method



Local explanation method for text classification



What should be the neighborhood of a text?

Local explanation method for text classification

Input text: “good food.”
(Binary classification problem)

Local explanation method for text classification

Input text: “good food.”
(Binary classification problem)

- **Perturbation** (Word Dropping)

  “good food.”
  “good.”
  “food.”
  “”

Local explanation method for text classification

Input text: “good food.”
(Binary classification problem)

- **Perturbation** (Word Dropping)

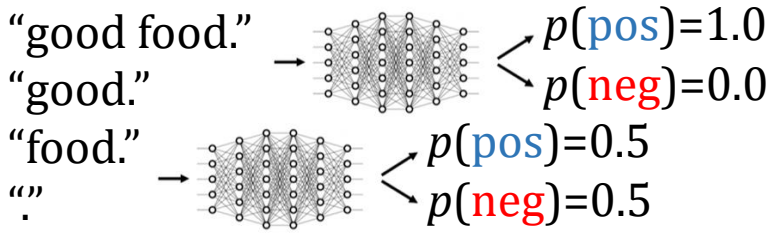
  “good food.”
  “good.”
  “food.”
  “”
  .

  → **Incomplete sentences**

Local explanation method for text classification

Input text: “good food.”
(Binary classification problem)

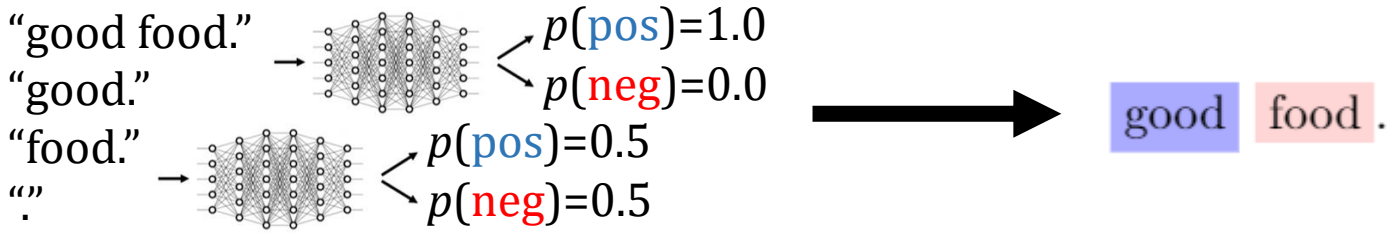
- **Perturbation** (Word Dropping)



Local explanation method for text classification

Input text: “good food.”
(Binary classification problem)

- **Perturbation** (Word Dropping)



Local explanation method for text classification

Input text: “good food.”
(Binary classification problem)

- **Perturbation** (Word Dropping)

  “good food.”
  “good.”
  “food.”
  “”

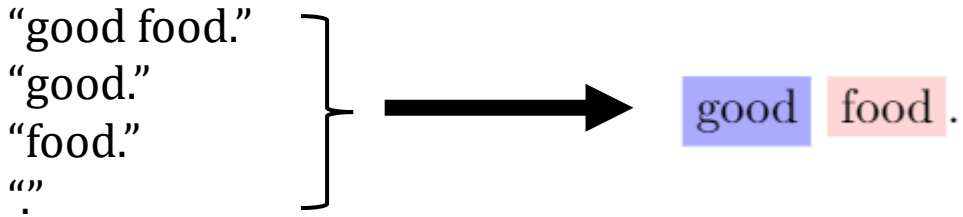
- Construct neighborhood in **latent space**

  “good food.”
  “very good!”
  “great food.”
  “bad food.”
  “horrible food.”
  …

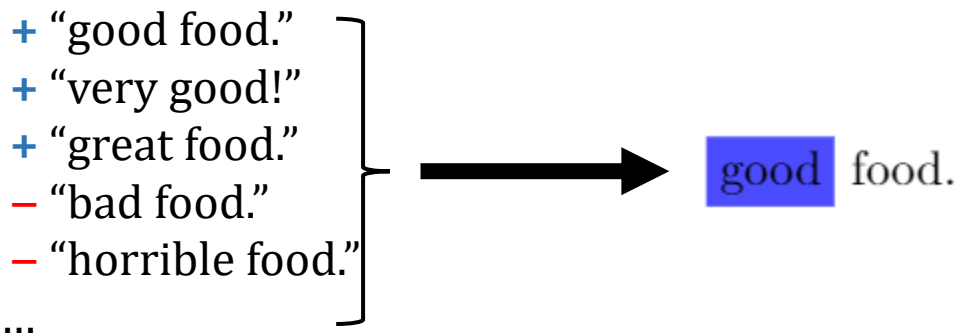
Local explanation method for text classification

Input text: “good food.”
(Binary classification problem)

- **Perturbation** (Word Dropping)



- Construct neighborhood in **latent space**

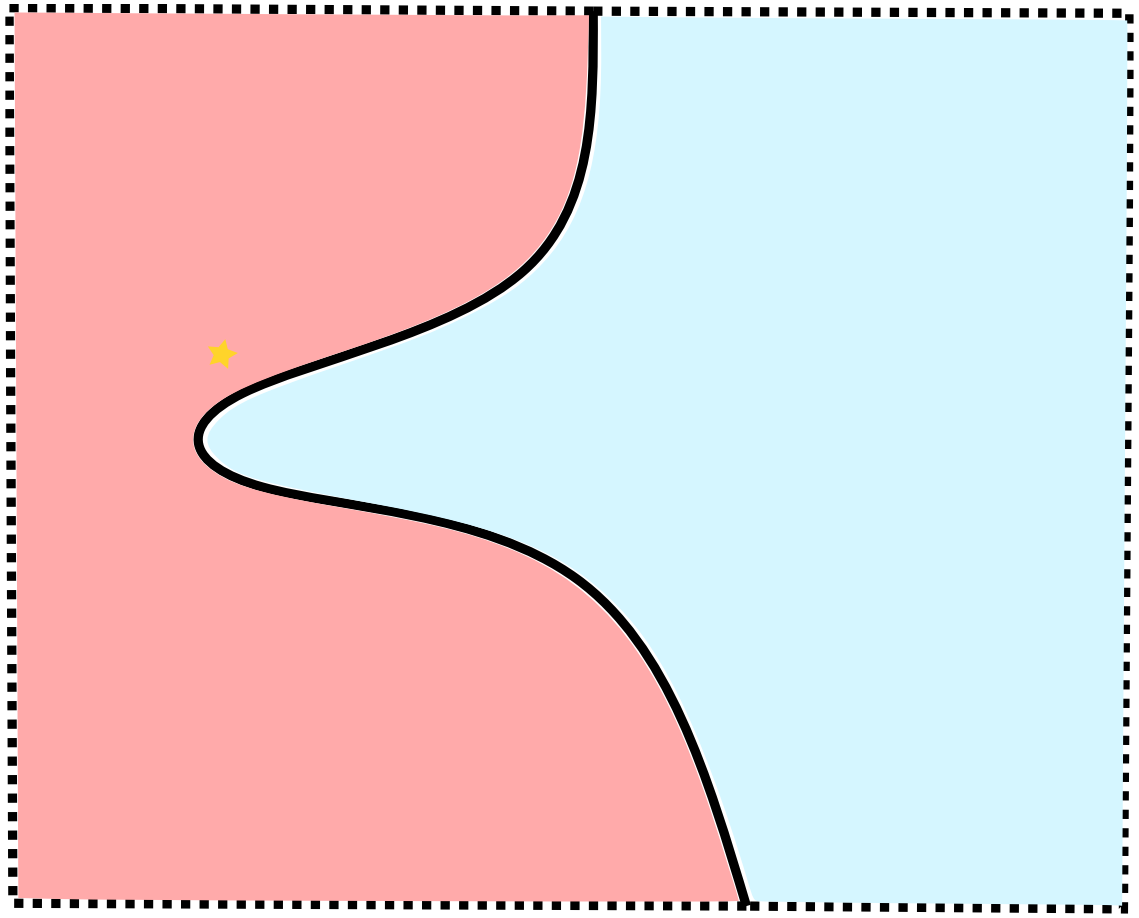


XPROAX – local eXplanation with PROgressive neighborhood ApproXimation

- How to construct a neighborhood in a high-dimensional latent space?

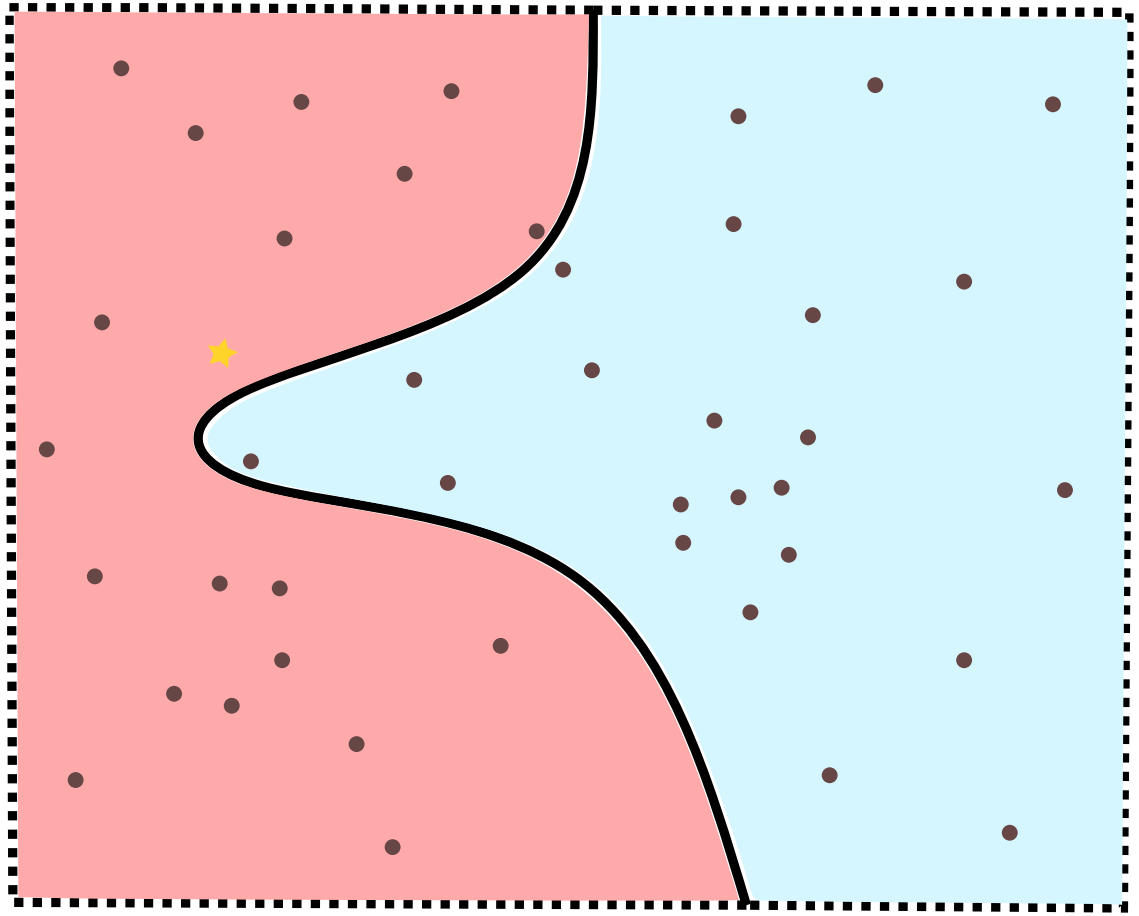
XPROAX – local eXplanation with PROgressive neighborhood ApproXimation

- Picking neighbors from a corpus as landmarks



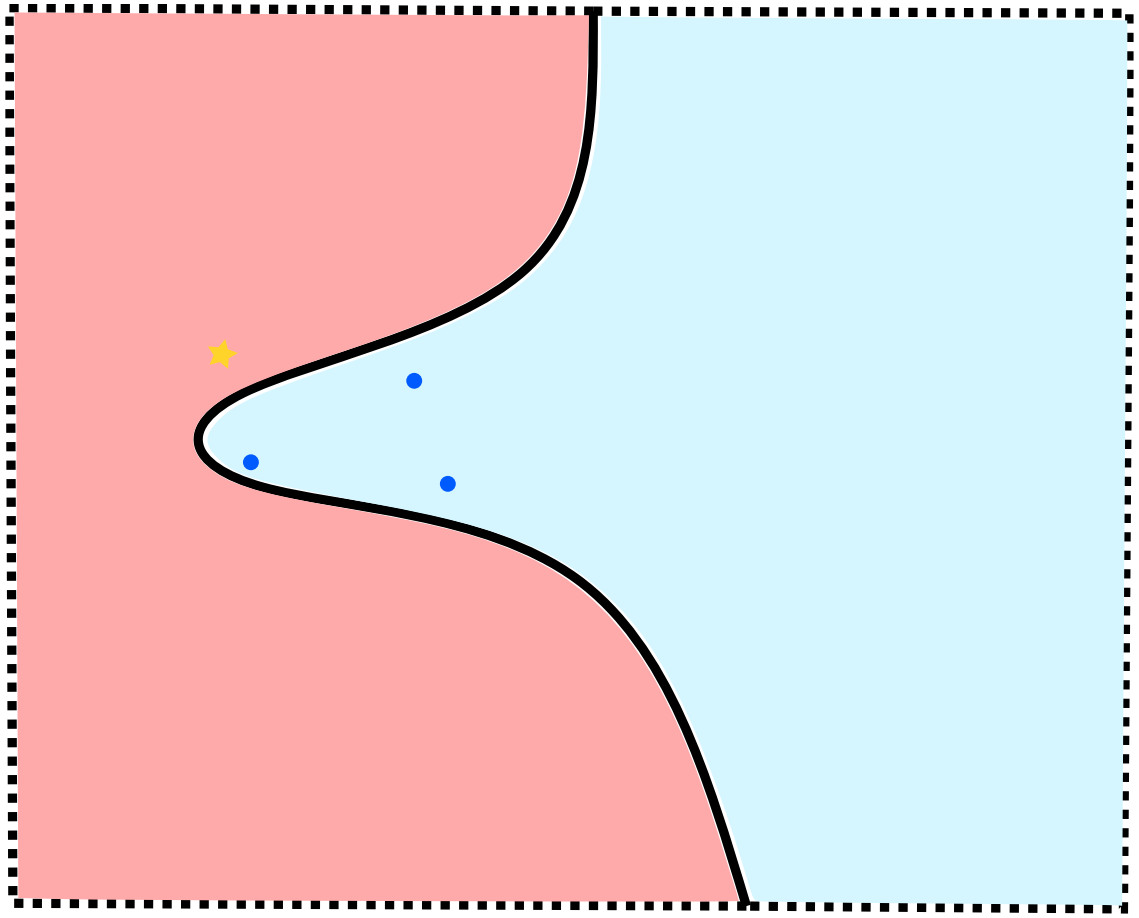
XPROAX – local eXplanation with PROgressive neighborhood ApproXimation

- Picking neighbors (with the opposite label) from a corpus as landmarks



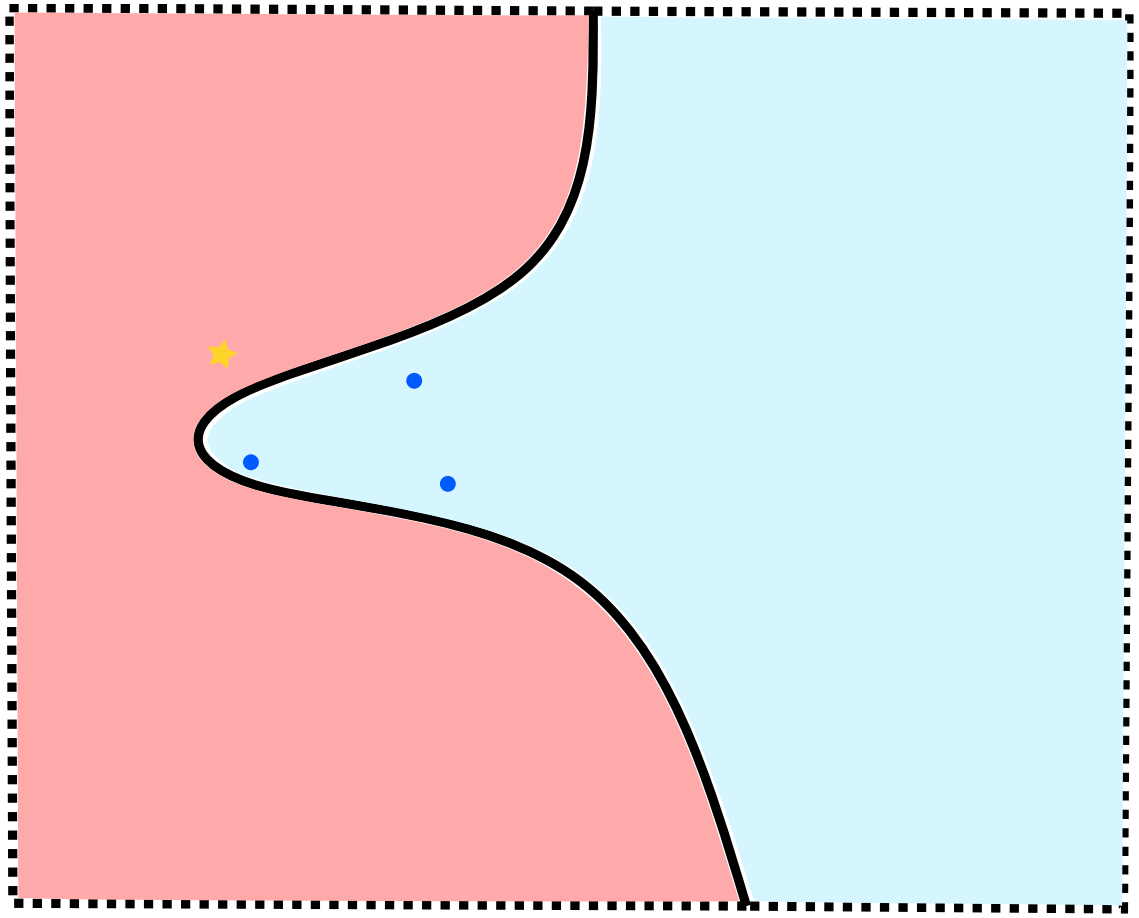
XPROAX – local eXplanation with PROgressive neighborhood ApproXimation

- Picking neighbors (with the opposite label) from a corpus as landmarks



XPROAX – local eXplanation with PROgressive neighborhood ApproXimation

- Picking neighbors (with the opposite label) from a corpus as landmarks
- Two-staged progressive approximation

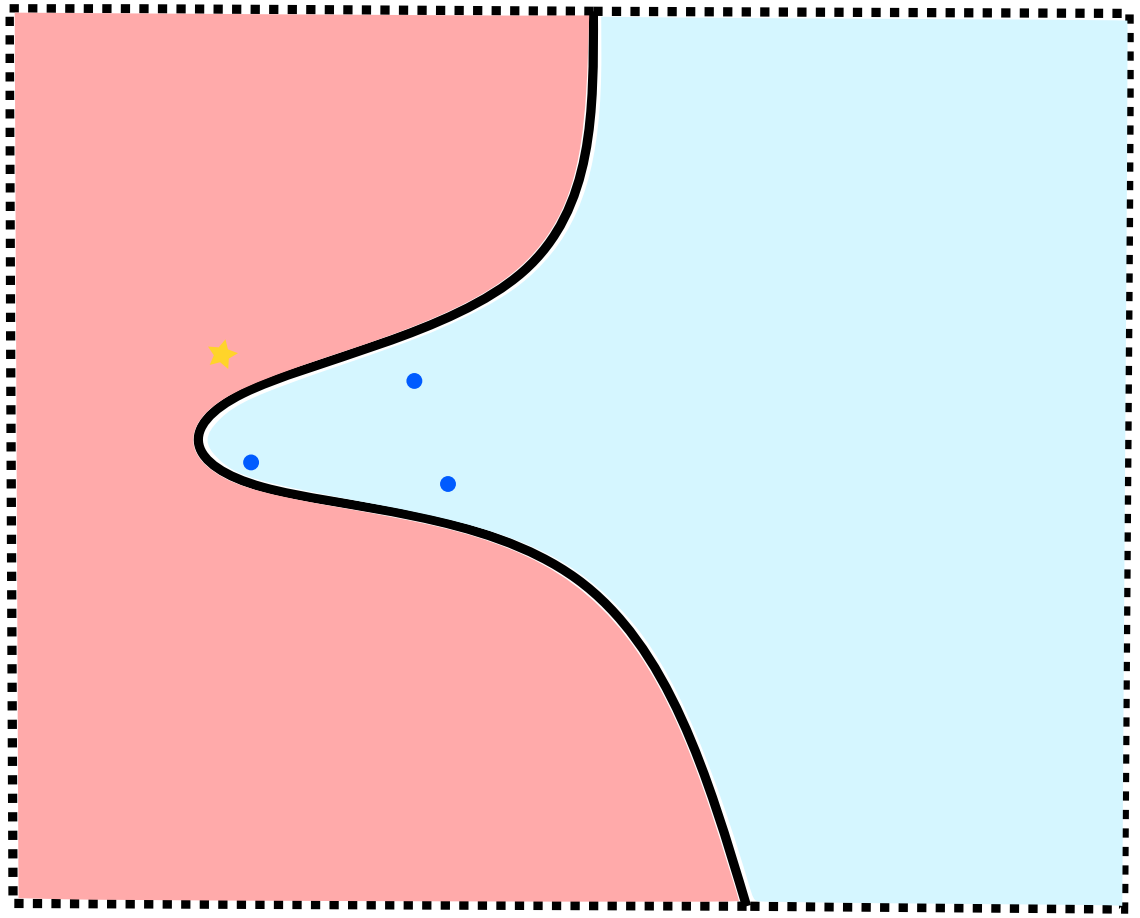


Two-staged progressive approximation

Algorithm 1 Neighborhood approximation

Input: x: query instance; C: set of counterfactual landmarks
Output: N_{new}: generated neighbors; C_{new}: updated C

$C_{new} = \emptyset$
repeat
  $z_p, z_q = \text{RandomlyDraw}(C, 2)$
    # draw randomly 2 vectors from the landmark set
  $Z' = I(z_p, z_q)$
      # 1st interpolation: between counterfactuals
  $Z = \emptyset$
  for $z_i \in Z'$ **do** *# 2nd interpolation: between polarities*
    if $b(D(z_i)) \neq b(x)$ **then**
      $Z \leftarrow Z + I(z_i, E(x))$
    end if
  end for
  $z_c = \text{ClosestCounterfactual}(Z, 1)$
  $C_{new}.insert(z_c)$
  $N_{new} \leftarrow N_{new} + D(Z)$
until k times
return N_{new}, C_{new}

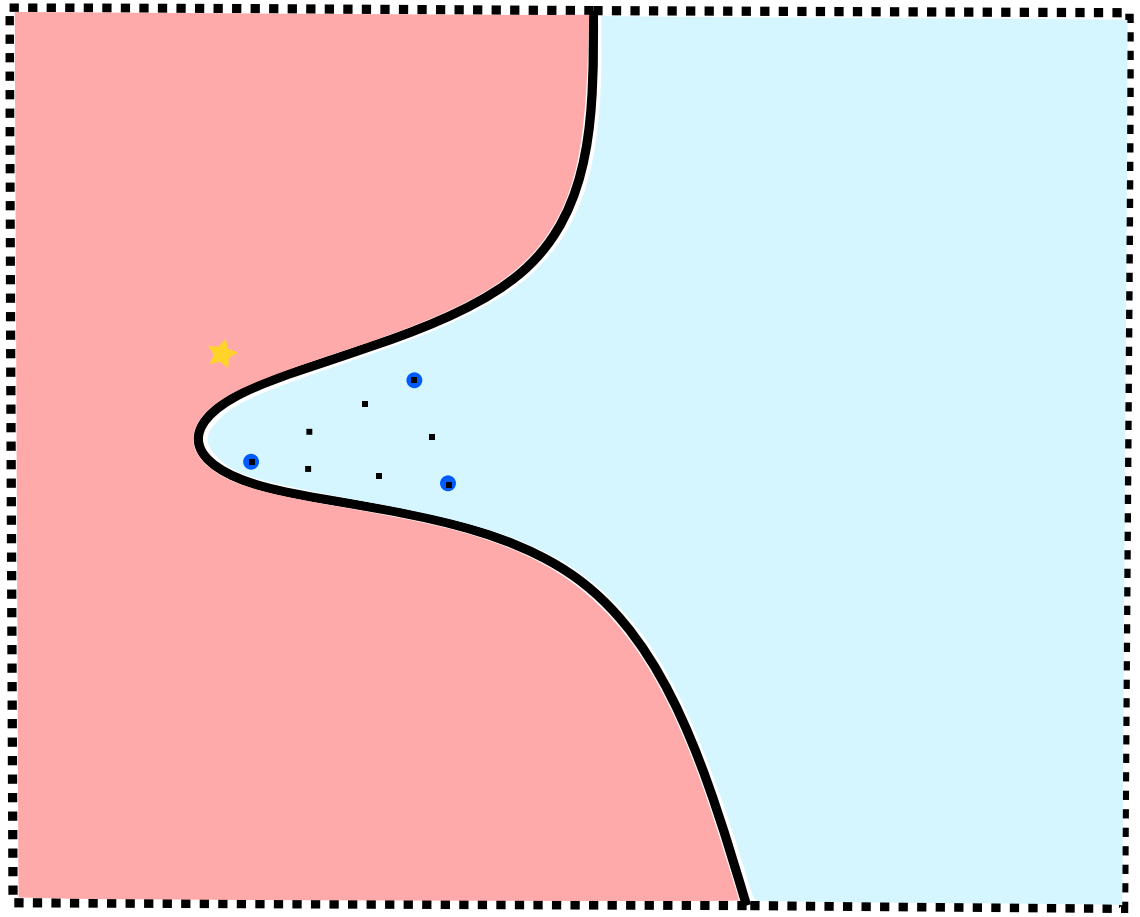


Two-staged progressive approximation

Algorithm 1 Neighborhood approximation

Input: x: query instance; C: set of counterfactual landmarks
Output: N_{new}: generated neighbors; C_{new}: updated C

$C_{new} = \emptyset$
repeat
  $z_p, z_q = \text{RandomlyDraw}(C, 2)$
    # draw randomly 2 vectors from the landmark set
  $Z' = I(z_p, z_q)$
    # 1st interpolation: between counterfactuals
  $Z = \emptyset$
  for $z_i \in Z'$ **do** *# 2nd interpolation: between polarities*
    if $b(D(z_i)) \neq b(x)$ **then**
      $Z \leftarrow Z + I(z_i, E(x))$
    end if
  end for
  $z_c = \text{ClosestCounterfactual}(Z, 1)$
  $C_{new}.insert(z_c)$
  $N_{new} \leftarrow N_{new} + D(Z)$
until k times
return N_{new}, C_{new}

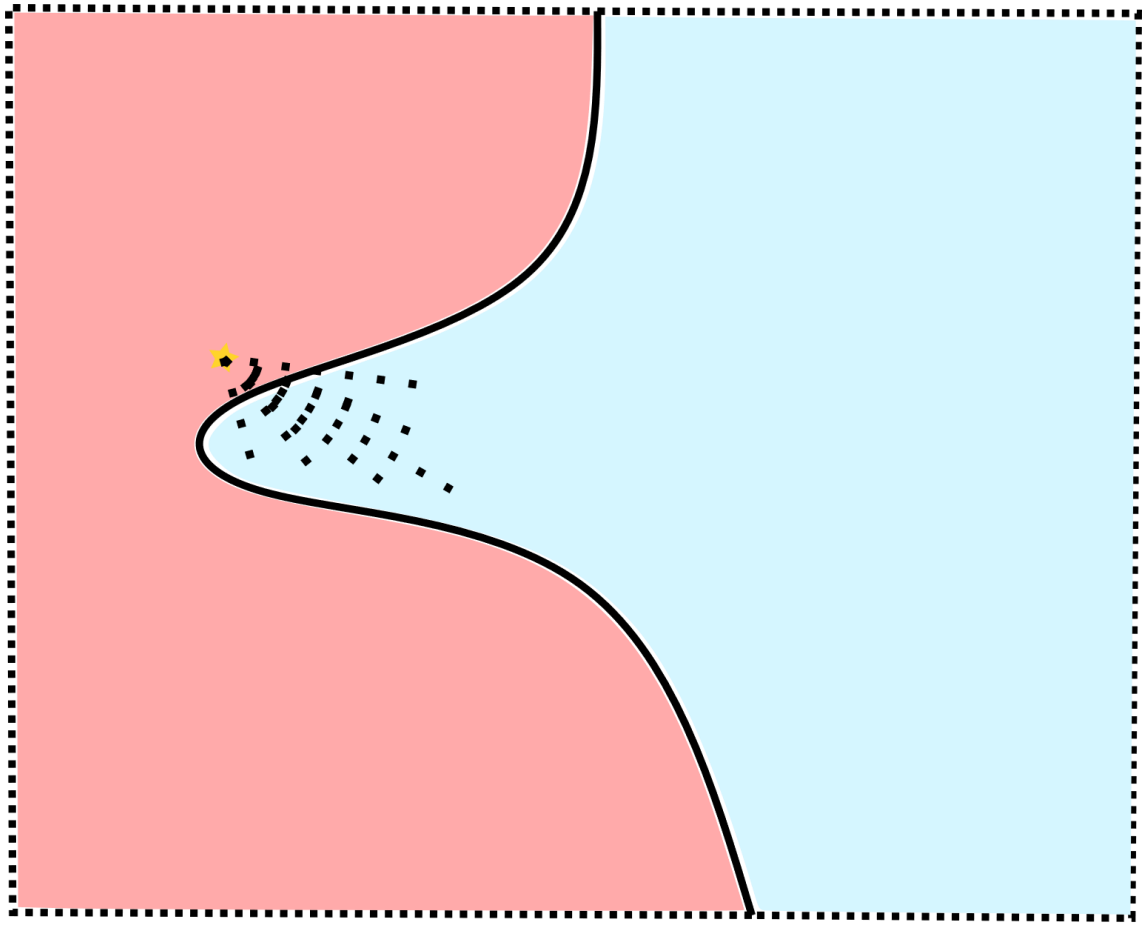


Two-staged progressive approximation

Algorithm 1 Neighborhood approximation

Input: x: query instance; C: set of counterfactual landmarks
Output: N_{new}: generated neighbors; C_{new}: updated C

$C_{new} = \emptyset$
repeat
  $z_p, z_q = \text{RandomlyDraw}(C, 2)$
    # draw randomly 2 vectors from the landmark set
  $Z' = I(z_p, z_q)$
      # 1st interpolation: between counterfactuals
  $Z = \emptyset$
  for $z_i \in Z'$ **do** *# 2nd interpolation: between polarities*
    if $b(D(z_i)) \neq b(x)$ **then**
      $Z \leftarrow Z + I(z_i, E(x))$
    end if
  end for
  $z_c = \text{ClosestCounterfactual}(Z, 1)$
  $C_{new}.insert(z_c)$
  $N_{new} \leftarrow N_{new} + D(Z)$
until k times
return N_{new}, C_{new}

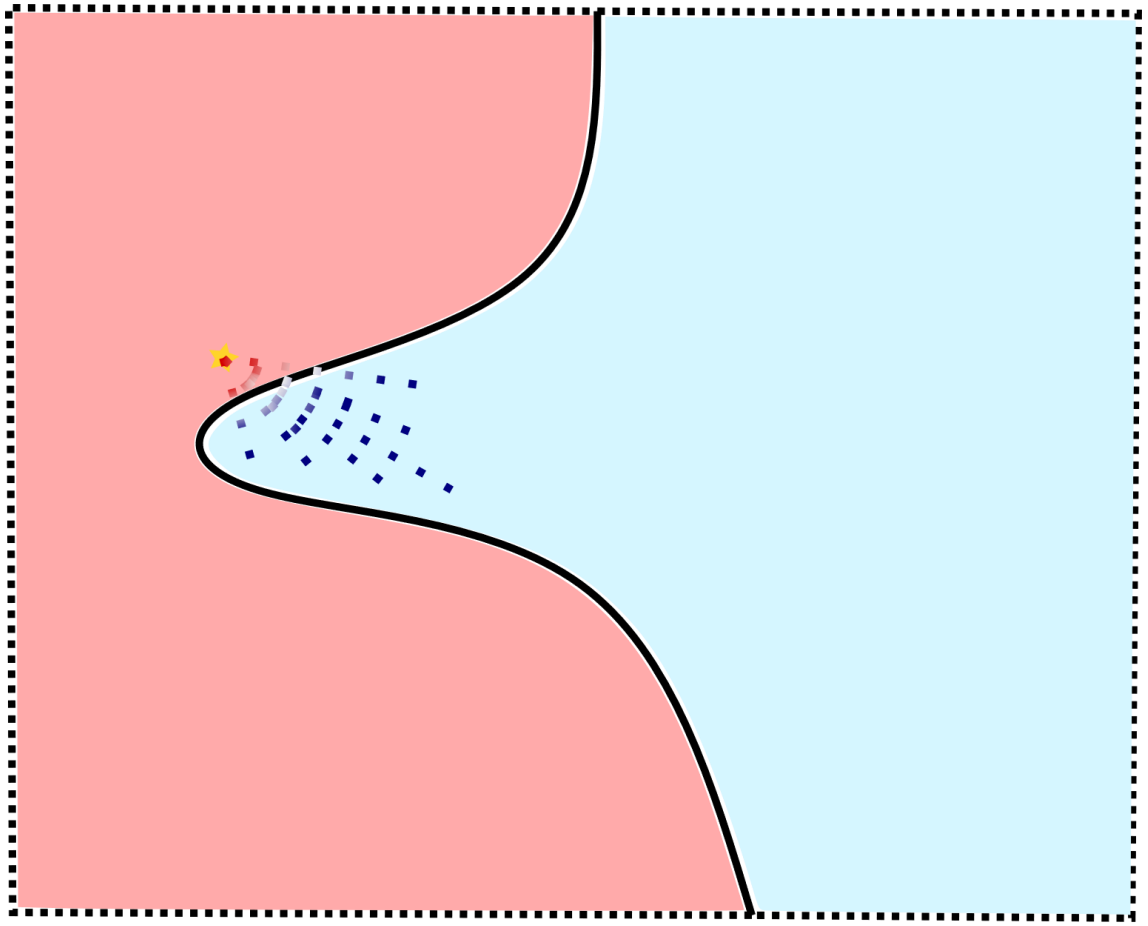


Two-staged progressive approximation

Algorithm 1 Neighborhood approximation

Input: x: query instance; C: set of counterfactual landmarks
Output: N_{new}: generated neighbors; C_{new}: updated C

$C_{new} = \emptyset$
repeat
  $z_p, z_q = \text{RandomlyDraw}(C, 2)$
    # draw randomly 2 vectors from the landmark set
  $Z' = I(z_p, z_q)$
      # 1st interpolation: between counterfactuals
  $Z = \emptyset$
  for $z_i \in Z'$ **do** *# 2nd interpolation: between polarities*
    if $b(D(z_i)) \neq b(x)$ **then**
      $Z \leftarrow Z + I(z_i, E(x))$
    end if
  end for
  $z_c = \text{ClosestCounterfactual}(Z, 1)$
  $C_{new}.insert(z_c)$
  $N_{new} \leftarrow N_{new} + D(Z)$
until k times
return N_{new}, C_{new}

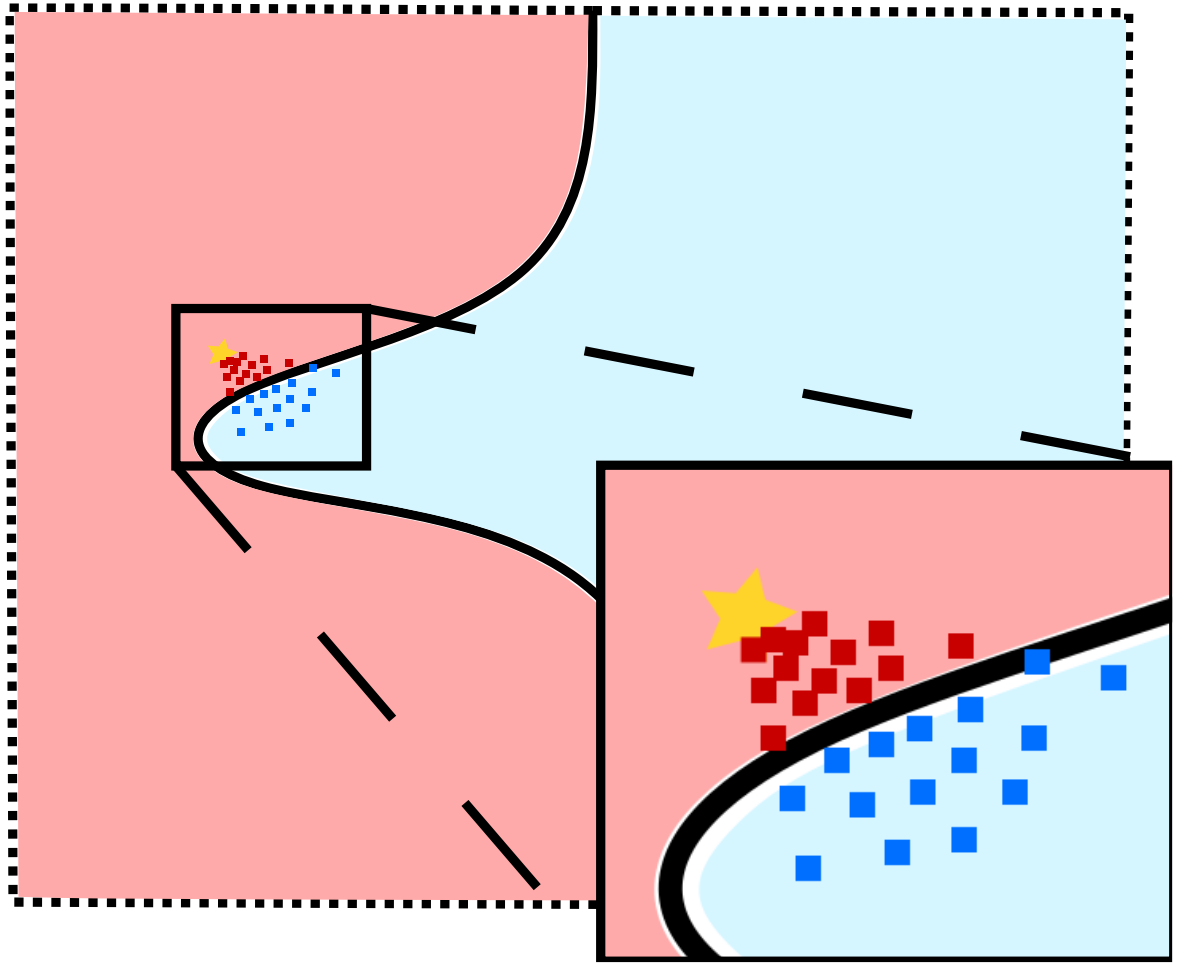


Two-staged progressive approximation

Algorithm 1 Neighborhood approximation

Input: x: query instance; C: set of counterfactual landmarks
Output: N_{new}: generated neighbors; C_{new}: updated C

$C_{new} = \emptyset$
repeat
  $z_p, z_q = \text{RandomlyDraw}(C, 2)$
    # draw randomly 2 vectors from the landmark set
  $Z' = I(z_p, z_q)$
      # 1st interpolation: between counterfactuals
  $Z = \emptyset$
  for $z_i \in Z'$ **do** *# 2nd interpolation: between polarities*
    if $b(D(z_i)) \neq b(x)$ **then**
      $Z \leftarrow Z + I(z_i, E(x))$
    end if
  end for
  $z_c = \text{ClosestCounterfactual}(Z, 1)$
  $C_{new}.insert(z_c)$
  $N_{new} \leftarrow N_{new} + D(Z)$
until k times
return N_{new}, C_{new}



Experimental results

Competitors:

- **LIME**: applies word dropping to the input text for the neighborhood construction
- **XSPELLS**: deploys a generative autoencoder and performs random sampling in the latent space

Experimental results

Dataset: **Yelp**, Black box model $b(\cdot)$: **DNN**

Input Text	$b(\cdot)$	XPROAX	LIME	XSPELLS[a]
great food.	**1.00**	great 0.69 food 0.05	great 0.70 food -0.24	food 0.11 italian* 0.08
great sushi.	**0.99**	great 0.62 sushi 0.12	great 0.56 sushi -0.11	salsa 0.17 guys 0.14
great pizza.	**1.00**	great 0.70 pizza 0.01	great 0.73 pizza -0.27	good 0.14 food 0.08
great beer.	**1.00**	great 0.64 beer 0.02	great 0.73 beer -0.27	good 0.15 story* 0.07

Experimental results

Dataset: **Yelp**, Black box model $b(\cdot)$: **DNN**

Input Text	$b(\cdot)$	XPROAX	LIME	XSPELLS[a]
great food.	**1.00**	great 0.69 food 0.05	great 0.70 ➡ food -0.24	food 0.11 italian* 0.08
great sushi.	**0.99**	great 0.62 sushi 0.12	great 0.56 ➡ sushi -0.11	salsa 0.17 guys 0.14
great pizza.	**1.00**	great 0.70 pizza 0.01	great 0.73 ➡ pizza -0.27	good 0.14 food 0.08
great beer.	**1.00**	great 0.64 beer 0.02	great 0.73 ➡ beer -0.27	good 0.15 story* 0.07

Experimental results

Dataset: **Yelp**, Black box model $b(\cdot)$: **DNN**

Input Text	$b(\cdot)$	XPROAX	LIME	XSPELLS[a]
great food.	**1.00**	great 0.69 ➡ food 0.05	great 0.70 food -0.24	food 0.11 italian* 0.08
great sushi.	**0.99**	great 0.62 ➡ sushi 0.12	great 0.56 sushi -0.11	salsa 0.17 guys 0.14
great pizza.	**1.00**	great 0.70 ➡ pizza 0.01	great 0.73 pizza -0.27	good 0.14 food 0.08
great beer.	**1.00**	great 0.64 ➡ beer 0.02	great 0.73 beer -0.27	good 0.15 story* 0.07

Experimental results

Input 2	fries are n't worth coming back .	Random Forest $b(\cdot)$: **positive**, Dataset: **Yelp**
XPROAX	**Saliency**: fries are n't worth coming back . **Factuals**: 1) the fries were n't worth coming. 2) _unk_[b] are n't worth going back. 3) the fries were worth coming back. 4) the fries were worth going back. 5) you do n't be worth coming.	**Extrinsic words**[a]: not perfect **Counterfactuals**: 1) _unk_ do n't bother in back. 2) _unk_ do n't bother going back. 3) _unk_ do n't be anybody back. 4) a few fries were definately coming back. 5) _unk_ do n't be anybody.
XSPELLS	**Factuals**: 1) it seems well they did 2) and i feel like on service 3) dave is excellent 4) everything we will get 5) all i hung up is nice **Common words in factuals**: seems (0.091), well (0.091), feel (0.091)	**Counterfactuals**: 1) both to die 2) all else s 3) every i may 4) who makes me money last 5) all were nt pricey **Common words in counterfactuals**: die (0.111), else (0.111), every: (0.111)
LIME	**Saliency**: fries are n't worth coming back .	

Experimental results

Input 2	fries are n't worth coming back .	Random Forest $b(\cdot)$: **positive**, Dataset: **Yelp**
XPROAX	**Saliency**: fries are n't worth coming back . **Factuals**: 1) the fries were n't worth coming. 2) _unk_[b] are n't worth going back. 3) the fries were worth coming back. 4) the fries were worth going back. 5) you do n't be worth coming.	**Extrinsic words**[a]: not perfect **Counterfactuals**: 1) _unk_ do n't bother in back. 2) _unk_ do n't bother going back. 3) _unk_ do n't be anybody back. 4) a few fries were definately coming back. 5) _unk_ do n't be anybody.
XSPELLS	**Factuals**: 1) it seems well they did 2) and i feel like on service 3) dave is excellent 4) everything we will get 5) all i hung up is nice **Common words in factuals**: seems (0.091), well (0.091), feel (0.091)	**Counterfactuals**: 1) both to die 2) all else s 3) every i may 4) who makes me money last 5) all were nt pricey **Common words in counterfactuals**: die (0.111), else (0.111), every: (0.111)
LIME	**Saliency**: fries are n't worth coming back .	

Experimental results

Input 2	fries are n't worth coming back .	Random Forest $b(\cdot)$: **positive**, Dataset: **Yelp**
XPROAX	**Saliency**: fries are n't worth coming back . **Factuals**: 1) the fries were n't worth coming. 2) _unk_[b] are n't worth going back. 3) the fries were worth coming back. 4) the fries were worth going back. 5) you do n't be worth coming.	**Extrinsic words**[a]: not perfect **Counterfactuals**: 1) _unk_ do n't bother in back. 2) _unk_ do n't bother going back. 3) _unk_ do n't be anybody back. 4) a few fries were definately coming back. 5) _unk_ do n't be anybody.
XSPELLS	**Factuals**: 1) it seems well they did 2) and i feel like on service 3) dave is excellent 4) everything we will get 5) all i hung up is nice **Common words in factuals**: seems (0.091), well (0.091), feel (0.091)	**Counterfactuals**: 1) both to die 2) all else s 3) every i may 4) who makes me money last 5) all were nt pricey **Common words in counterfactuals**: die (0.111), else (0.111), every: (0.111)
LIME	**Saliency**: fries are n't worth coming back .	

Experimental results

Input 2	fries are n't worth coming back .	Random Forest $b(\cdot)$: **positive**, Dataset: **Yelp**
XPROAX	**Saliency**: fries are n't worth coming back . **Factuals**: 1) the fries were n't worth coming. 2) _unk_[b] are n't worth going back. 3) the fries were worth coming back. 4) the fries were worth going back. 5) you do n't be worth coming.	**Extrinsic words**[a]: not perfect **Counterfactuals**: 1) _unk_ do n't bother in back. 2) _unk_ do n't bother going back. 3) _unk_ do n't be anybody back. 4) a few fries were definately coming back. 5) _unk_ do n't be anybody.
XSPELLS	**Factuals**: 1) it seems well they did 2) and i feel like on service 3) dave is excellent 4) everything we will get 5) all i hung up is nice **Common words in factuals**: seems (0.091), well (0.091), feel (0.091)	**Counterfactuals**: 1) both to die 2) all else s 3) every i may 4) who makes me money last 5) all were nt pricey **Common words in counterfactuals**: die (0.111), else (0.111), every: (0.111)
LIME	**Saliency**: fries are n't worth coming back .	

Experimental results

Input 2	fries are n't worth coming back .	Random Forest $b(\cdot)$: **positive**, Dataset: **Yelp**
XPROAX	**Saliency**: fries are n't worth coming back . **Factuals**: 1) the fries were n't worth coming. 2) _unk_[b] are n't worth going back. 3) the fries were worth coming back. 4) the fries were worth going back. 5) you do n't be worth coming.	**Extrinsic words**[a]: not perfect **Counterfactuals**: 1) _unk_ do n't bother in back. 2) _unk_ do n't bother going back. 3) _unk_ do n't be anybody back. 4) a few fries were definately coming back. 5) _unk_ do n't be anybody.
XSPELLS	**Factuals**: 1) it seems well they did 2) and i feel like on service 3) dave is excellent 4) everything we will get 5) all i hung up is nice **Common words in factuals**: seems (0.091), well (0.091), feel (0.091)	**Counterfactuals**: 1) both to die 2) all else s 3) every i may 4) who makes me money last 5) all were nt pricey **Common words in counterfactuals**: die (0.111), else (0.111), every: (0.111)
LIME	**Saliency**: fries are n't worth coming back .	

Conclusion

- A local model-agnostic explanation method for text classification

Conclusion

- A local model-agnostic explanation method for text classification
- Construct the neighborhood of a text in a latent space with the progressive approximation approach

Conclusion

- A local model-agnostic explanation method for text classification
- Construct the neighborhood of a text in a latent space with the progressive approximation approach
- Detailed explanations for better understanding

Thank you for listening!

Authors: **Yi Cai**, **Arthur Zimek**, **Eirini Ntoutsi**

Contact: **cai@l3s.de**

Code: **https://github.com/caiy0220/XPROAX**

This work is supported by the State Ministry of Science and Culture of Lower Saxony, within the PhD program “Responsible Artificial Intelligence in the Digital Society”.

Niedersächsisches Ministerium
für Wissenschaft und Kultur