Learning Robust Representations of Text

A Discussion

Naganand Y

Abstract

- Deep nets sensitive to noise, adversarial attacks

Abstract

- Deep nets sensitive to noise, adversarial attacks
- Regularization method to limit network sensitivity to inputs

Abstract

- Deep nets sensitive to noise, adversarial attacks
- Regularization method to limit network sensitivity to inputs
    - Models more robust

Abstract

- Deep nets sensitive to noise, adversarial attacks
- Regularization method to limit network sensitivity to inputs
  - Models more robust
  - Inspired by computer vision

Abstract

- Deep nets sensitive to noise, adversarial attacks
- Regularization method to limit network sensitivity to inputs
  - Models more robust
  - Inspired by computer vision
- Superior performance on noisy inputs, out-of-domain data

Abstract

- Deep nets sensitive to noise, adversarial attacks
- Regularization method to limit network sensitivity to inputs
  - Models more robust
  - Inspired by computer vision
- Superior performance on noisy inputs, out-of-domain data
  - Sentiment datasets

Introduction

- Primary cause of neural nets' vulnerability is linear nature[1]

[1] Goodfellow et al. , Explaining and harnessing adversarial examples, ICLR 2014
[2] Analysis of classifiers' robustness to adversarial perturbations
[3] Contractive autoencoders: Explicit invariance during feature extraction, ICML 2011

Introduction

- Primary cause of neural nets' vulnerability is linear nature[1]
  - LSTMs, ReLUs, maxout designed linearly to facilitate optimization

[1] Goodfellow et al. , Explaining and harnessing adversarial examples, ICLR 2014
[2] Analysis of classifiers' robustness to adversarial perturbations
[3] Contractive autoencoders: Explicit invariance during feature extraction, ICML 2011

Introduction

- Primary cause of neural nets' vulnerability is linear nature[1]
  - LSTMs, ReLUs, maxout designed linearly to facilitate optimization
- Fawzi et al.[2]showed linear models not robust to adversarial noise

[1]Goodfellow et al. , Explaining and harnessing adversarial examples, ICLR 2014
[2]Analysis of classifiers' robustness to adversarial perturbations
[3]Contractive autoencoders: Explicit invariance during feature extraction, ICML 2011

Introduction

- Primary cause of neural nets' vulnerability is linear nature[1]
  - LSTMs, ReLUs, maxout designed linearly to facilitate optimization
- Fawzi et al.[2]showed linear models not robust to adversarial noise
- Present a regularization method to make neural nets more robust to noise

[1]Goodfellow et al. , Explaining and harnessing adversarial examples, ICLR 2014
[2]Analysis of classifiers' robustness to adversarial perturbations
[3]Contractive autoencoders: Explicit invariance during feature extraction, ICML 2011

Introduction

- Primary cause of neural nets' vulnerability is linear nature[1]
    - LSTMs, ReLUs, maxout designed linearly to facilitate optimization
- Fawzi et al.[2] showed linear models not robust to adversarial noise
- Present a regularization method to make neural nets more robust to noise
    - Inspired by Rifai et al [3]

[1] Goodfellow et al. , Explaining and harnessing adversarial examples, ICLR 2014
[2] Analysis of classifiers' robustness to adversarial perturbations
[3] Contractive autoencoders: Explicit invariance during feature extraction, ICML 2011

Approach

- **Intuition**: Minimize ability of features to perturb predictions

Approach

- **Intuition**: Minimize ability of features to perturb predictions
  - to stabilize predictions

Approach

- **Intuition**: Minimize ability of features to perturb predictions
  - to stabilize predictions
- **Idea**: Train models using first-order derivatives of training loss as part of regularization term

Approach

- **Intuition**: Minimize ability of features to perturb predictions
  - to stabilize predictions
- **Idea**: Train models using first-order derivatives of training loss as part of regularization term
  - Necessitates second-order derivatives for computing gradient

Training for Robustness

- **Training**: SGD to min L (measures $y_{pred} - y_{true}$)

Training for Robustness

- **Training**: SGD to min L (measures $y_{pred} - y_{true}$)
  - w: Input, a sequence of (discrete) words

Training for Robustness

- **Training**: SGD to min L (measures $y_{pred} - y_{true}$)
  - w: Input, a sequence of (discrete) words
  - h: fixed-size vector of continuous values representing w

Training for Robustness

- **Training**: SGD to min L (measures $y_{pred} - y_{true}$)
  - w: Input, a sequence of (discrete) words
  - h: fixed-size vector of continuous values representing w
  - $y_{pred} = f(h)$

Training for Robustness

- **Training**: SGD to min L (measures $y_{pred} - y_{true}$)
  - w: Input, a sequence of (discrete) words
  - h: fixed-size vector of continuous values representing w
  - $y_{pred} = f(h)$
- **Goal**: Learn models that are more robust to strange/invalid inputs

Training for Robustness

- **Training**: SGD to min L (measures $y_{pred} - y_{true}$)
  - w: Input, a sequence of (discrete) words
  - h: fixed-size vector of continuous values representing w
  - $y_{pred} = f(h)$
- **Goal**: Learn models that are more robust to strange/invalid inputs
  - y_{pred} remains stable on perturbations on w (or h)

Training for Robustness

- **Training**: SGD to min L (measures $y_{pred} - y_{true}$)
  - w: Input, a sequence of (discrete) words
  - h: fixed-size vector of continuous values representing w
  - $y_{pred} = f(h)$
- **Goal**: Learn models that are more robust to strange/invalid inputs
  - y_{pred} remains stable on perturbations on w (or h)
- **Application**: Transfer learning scenarios such as domain adaptation

Training for Robustness

- **Training**: SGD to min L (measures $y_{pred} - y_{true}$)
  - w: Input, a sequence of (discrete) words
  - h: fixed-size vector of continuous values representing w
  - $y_{pred} = f(h)$
- **Goal**: Learn models that are more robust to strange/invalid inputs
  - y_{pred} remains stable on perturbations on w (or h)
- **Application**: Transfer learning scenarios such as domain adaptation
  - Inputs in distinct domains drawn from different distributions

Training for Robustness

- **Training**: SGD to min L (measures $y_{pred} - y_{true}$)
  - w: Input, a sequence of (discrete) words
  - h: fixed-size vector of continuous values representing w
  - $y_{pred} = f(h)$
- **Goal**: Learn models that are more robust to strange/invalid inputs
  - y_{pred} remains stable on perturbations on w (or h)
- **Application**: Transfer learning scenarios such as domain adaptation
  - Inputs in distinct domains drawn from different distributions
  - Highly variable but convey same information

Training for Robustness

- **Training**: SGD to min L (measures $y_{pred} - y_{true}$)
  - w: Input, a sequence of (discrete) words
  - h: fixed-size vector of continuous values representing w
  - $y_{pred} = f(h)$
- **Goal**: Learn models that are more robust to strange/invalid inputs
  - y_{pred} remains stable on perturbations on w (or h)
- **Application**: Transfer learning scenarios such as domain adaptation
  - Inputs in distinct domains drawn from different distributions
  - Highly variable but convey same information
  - Different word choice, different syntactic structures, typographical errors, stylistic changes, etc

Robust Regularization

- Minimize variation of output when noise applied to input

Robust Regularization

- Minimize variation of output when noise applied to input
  - $\Delta_y = f(x + \Delta_x) - f(x)$

Robust Regularization

- Minimize variation of output when noise applied to input
  - $\Delta_y = f(x + \Delta_x) - f(x)$
  - $$\lim_{\Delta_x \to 0} \Delta_y$$

Robust Regularization

- Minimize variation of output when noise applied to input
  - $\Delta_y = f(x + \Delta_x) - f(x)$
  - $$\lim_{\Delta_x \to 0} \Delta_y = \lim_{\Delta_x \to 0} \left(f(x + \Delta_x) - f(x) \right)$$

Robust Regularization

- Minimize variation of output when noise applied to input
  - $\Delta_y = f(x + \Delta_x) - f(x)$
  - $$\lim_{\Delta_x \to 0} \Delta_y = \lim_{\Delta_x \to 0} \left(f(x + \Delta_x) - f(x) \right) = \frac{\partial y}{\partial x} \cdot \Delta_x$$

Robust Regularization

- Minimize variation of output when noise applied to input
  - $\Delta_y = f(x + \Delta_x) - f(x)$
  - $$\lim_{\Delta_x \to 0} \Delta_y = \lim_{\Delta_x \to 0} \big(f(x + \Delta_x) - f(x)\big) = \frac{\partial y}{\partial x} \cdot \Delta_x$$
  - Minimising noise sensitivity $\equiv$ minimising $\left\| \frac{\partial y}{\partial x} \right\|_F$

Robust Regularization

- Minimize variation of output when noise applied to input
  - $\Delta_y = f(x + \Delta_x) - f(x)$
  - $$\lim_{\Delta_x \to 0} \Delta_y = \lim_{\Delta_x \to 0} \left(f(x + \Delta_x) - f(x)\right) = \frac{\partial y}{\partial x} \cdot \Delta_x$$
  - Minimising noise sensitivity $\equiv$ minimising $\left\| \frac{\partial y}{\partial x} \right\|_F$
- $\mathcal{L} = L + \lambda \cdot \left\| \frac{\partial L}{\partial h} \right\|_2$

Robust Regularization

- Minimize variation of output when noise applied to input
  - $\Delta_y = f(x + \Delta_x) - f(x)$
  - $$\lim_{\Delta_x \to 0} \Delta_y = \lim_{\Delta_x \to 0} \left(f(x + \Delta_x) - f(x)\right) = \frac{\partial y}{\partial x} \cdot \Delta_x$$
  - Minimising noise sensitivity $\equiv$ minimising $\left\| \frac{\partial y}{\partial x} \right\|_F$
- $\mathcal{L} = L + \lambda \cdot \left\| \frac{\partial L}{\partial h} \right\|_2$
  - Supports gradient optimization

Robust Regularization

- Minimize variation of output when noise applied to input
  - $\Delta_y = f(x + \Delta_x) - f(x)$
  - $$\lim_{\Delta_x \to 0} \Delta_y = \lim_{\Delta_x \to 0} \left(f(x + \Delta_x) - f(x)\right) = \frac{\partial y}{\partial x} \cdot \Delta_x$$
  - Minimising noise sensitivity $\equiv$ minimising $\left\|\frac{\partial y}{\partial x}\right\|_F$
- $\mathcal{L} = L + \lambda \cdot \left\|\frac{\partial L}{\partial h}\right\|_2$
  - Supports gradient optimization
  - Need to compute second-order derivatives of L during back-propagation

Experiments and Datasets

- Model f is CNN proposed by Yoon Kim[4]

[1] Convolutional neural networks for sentence classification, EMNLP 2014

Experiments and Datasets

- Model f is CNN proposed by Yoon Kim[4]
  - MR: Sentence polarity dataset
  - Subj: Subjectivity dataset
  - CR: Customer review dataset
  - SST: Stanford Sentiment Treebank, using the 3-class configuration

[1] Convolutional neural networks for sentence classification, EMNLP 2014

Experiments and Datasets

- Model f is CNN proposed by Yoon Kim[4]
  - MR: Sentence polarity dataset
  - Subj: Subjectivity dataset
  - CR: Customer review dataset
  - SST: Stanford Sentiment Treebank, using the 3-class configuration
- **Noise**: Apply world-level dropout noise to each document

[1] Convolutional neural networks for sentence classification, EMNLP 2014

Experiments and Datasets

- Model f is CNN proposed by Yoon Kim[4]
  - MR: Sentence polarity dataset
  - Subj: Subjectivity dataset
  - CR: Customer review dataset
  - SST: Stanford Sentiment Treebank, using the 3-class configuration
- **Noise**: Apply world-level dropout noise to each document
  - Randomly replace words by a unique sentinel symbol

[1] Convolutional neural networks for sentence classification, EMNLP 2014

Experiments and Datasets

- Model f is CNN proposed by Yoon Kim[4]
  - MR: Sentence polarity dataset
  - Subj: Subjectivity dataset
  - CR: Customer review dataset
  - SST: Stanford Sentiment Treebank, using the 3-class configuration
- **Noise**: Apply world-level dropout noise to each document
  - Randomly replace words by a unique sentinel symbol
  - Apply this to each word with probability $\alpha \in \{0, 0.1, 0.2, 0.3\}$

[1] Convolutional neural networks for sentence classification, EMNLP 2014

Experiments and Datasets

- Model f is CNN proposed by Yoon Kim[4]
  - MR: Sentence polarity dataset
  - Subj: Subjectivity dataset
  - CR: Customer review dataset
  - SST: Stanford Sentiment Treebank, using the 3-class configuration
- **Noise**: Apply world-level dropout noise to each document
  - Randomly replace words by a unique sentinel symbol
  - Apply this to each word with probability $\alpha \in \{0, 0.1, 0.2, 0.3\}$
- Cross-domain evaluation

[1] Convolutional neural networks for sentence classification, EMNLP 2014

Experiments and Datasets

- Model f is CNN proposed by Yoon Kim[4]
  - MR: Sentence polarity dataset
  - Subj: Subjectivity dataset
  - CR: Customer review dataset
  - SST: Stanford Sentiment Treebank, using the 3-class configuration
- **Noise**: Apply world-level dropout noise to each document
  - Randomly replace words by a unique sentinel symbol
  - Apply this to each word with probability $\alpha \in \{0, 0.1, 0.2, 0.3\}$
- Cross-domain evaluation
  - Train on one dataset, apply it to another

[1] Convolutional neural networks for sentence classification, EMNLP 2014

Experiments and Datasets

- Model f is CNN proposed by Yoon Kim[4]
  - MR: Sentence polarity dataset
  - Subj: Subjectivity dataset
  - CR: Customer review dataset
  - SST: Stanford Sentiment Treebank, using the 3-class configuration
- **Noise**: Apply world-level dropout noise to each document
  - Randomly replace words by a unique sentinel symbol
  - Apply this to each word with probability $\alpha \in \{0, 0.1, 0.2, 0.3\}$
- Cross-domain evaluation
  - Train on one dataset, apply it to another
  - Pair MR (movie reviews) and CR (product reviews) that use same label set

[1] Convolutional neural networks for sentence classification, EMNLP 2014