CS231n

Convolutional Neural Networks for Visual Recognition

Sungchan Yi

September 8, 2019

Contents

1 Introduction

1.1 History of Computer Vision

- ...
- ImageNet: Image classification challenge
- Huge improvement on 2012 - Convolutional Neural Networks (**CNN**)

1.2 CS231n Overview

- Focus on **image classification**
- Object detection, action classification, image captioning ...
- **CNN** !

2 Image Classification

2.1 Image Classification

- Map: Input Image $\rightarrow$ Predetermined Category
- Semantic Gap - computers only see the pixel values of the image
- Many problems that cause the pixel values to change
  - Viewpoints
  - Illumination
  - Deformation (poses)
  - Occlusion (part of the object)
  - Background Clutter (object looks similar to the background)
  - Interclass Variation (objects come in different sizes and shapes)
- Image classification algorithms must be able to handle all these problems

2.2 Attempts to Classify Images

- Finding corners or edges to determine the object
  - Brittle, doesn't work very well

  - **Not scalable** to work with other objects

- **Data-Driven Approach**

  1. Collect a dataset of images and labels
  2. Use ML algorithms to train a classifier
  3. Evaluate the classifier on new images

- Data-driven approach is much more general

2.3 Nearest Neighbor

1. Memorize all data and labels
2. Predict the label of the **most similar** training image

- How to compare images? **Distance Metric** !
  - L_1 distance (Manhatten Distance)[1]

$$d_1(I_1, I_2) = \sum_{\text{pixel } p} |I_1^p - I_2^p|$$

- With N examples, training takes $\mathcal{O}(1)$, prediction takes $\mathcal{O}(N)$.
- Generally, we want prediction to be fast, but slow training time is OK
- Only looking at a **single** neighbor may cause problems (outliers, noisy data, etc.)
- Motivation for **kNN**s

2.4 k-Nearest Neighbors

- Take the **majority vote** from k closest points
- L_2 distance (Euclidean Distance)

$$d_2(I_1, I_2) = \sqrt{\sum_{\text{pixel } p} (I_1^p - I_2^p)^2}$$

- L_1 distance depends on the coordinate system L_2 doesn't matter
- Generalizing the distance metric can lead to classifying other objects such as texts

[1] Dumb way to compare, but does reasonable things *sometimes* ...

2.5 Hyperparameters

- **Hyperparameters** are choices about the algorithm that we set, rather than by learning the parameter from data
- In kNNs, the value of k and the distance metric are hyperparameters
- **Problem dependent**. Must go through trial and error to choose the best hyperparameter
- Setting Hyperparameters
  - Split the data into 3 sets. Training set, validation set, and test set
  - Train the algorithm with many different hyperparameters on the training set
  - Evaluate with validation set, choose the best hyperparameter from the results
  - Run it once on the test set, and this result goes on the paper
- Never used on image data ...
  - Slow on prediction
  - Distance metrics on pixels are not informative (Distance metric may not caputre the difference between images)
  - Curse of dimensionality - data points increase exponentially as dimension increases, but there may not be enough data to cover the area densly (Need to densly cover the regions of the n-dimensional space, for the kNNs to work well)

2.6 Linear Classification

- Lego blocks - different components, as building blocks of neural networks
- **Parametric Approach**
  - Image x, weight parameters W for each pixel in the image
  - A function $f : (x, W) \rightarrow$ numbers giving class scores for each class
  - If input x is an $n \times 1$ vector and there were c classes, W must be $c \times n$ matrix
- Classifier summarizes our knowlegde on the data and store it inside the parameter W
- At test time, we use the parameter W to predict
- How to come up with the right structure for f ?
- **Linear** Classifier : $f(x, W) = Wx \ (+b)$
- There may be a bias term b to show preference for some class label

- (Idea) Each row of W will work as a template for matching the input image, and the dot product of each row and the input image vector will give the **similarity** of the input data to the class
- Problems with linear classifier
  - Learning single template for each class
  - If the class has variations on how the class might appear, the classifier averages out all the variations and tries to recognize the object by a single template
  - Using deeper models to remove this restriction will lead to better accuracy
- Linear classifiers draws hyperplanes on the n-dimensional space to classify images
- If hyperplanes cannot be drawn on the space, the linear classifier may struggle (parity problem, multi-modal data)

3 Loss Functions and Optimization