Introduction to Computer Vision

Alvin Lin

August 2018 - December 2018

Image Classification

Given a feature representation for images, how do we train a model for distinguishing features from different classes? If we have a picture of a parrot, image classification would tell us that the image contains a parrot, while object detection would try to locate the bounding box of the parrot. This is generally quite complex.

Nearest Neighbor Classifier

Nearest neighbor classifiers try to assign the label of the nearest training data point to each test data point. It is one of the easiest classification algorithms. K-nearest neighbors is a variant that finds the k closest neighbors from the training data. The labels of the k points can vote to classify the point in consideration.

Distance Functions for Features

- Euclidean distance:
$$D(h_1, h_2) = \sqrt{\sum_{i=1}^{N}(h_1(i) - h_2(i))^2}$$
- L1 distance:
$$D(h_1, h_2) = \sum_{i=1}^{N}\left|h_1(i) - h_2(i)\right|$$

- χ^2 distance:

$$D(h_1, h_2) = \sum_{i=1}^{N} \frac{(h_1(i) - h_2(i))^2}{h_1(i) + h_2(i)}$$

- Histogram intersection (similarity):

$$I(h_1, h_2) = \sum_{i=1}^{N} min(h_1(i), h_2(i))$$

- Hellinger kernel (similarity):

$$K(h_1, h_2) = \sum_{i=1}^{N} \sqrt{h_1(i) h_2(i)}$$

Histogram Classifiers and the Curse of Dimensionality

We can also represent class-conditional densities with a histogram. With enough data, estimates become quite good but the histogram can become very large with high-dimensional data.

It is hard to build a histogram based classifier when the feature space is high dimensional. It fails when there are too many histogram buckets.

Linear Classifiers and Support Vector Machines

Linear classifiers try to find the linear function (hyperplane) between a set of positive and negative examples. Support vector machines find the hyperplane that maximizes the margin between the positive and negative examples. Datasets that are linearly separable work well for support vector machines, but if the dataset is too difficult, we can map it to a higher dimensional space.

Nonlinear Support Vector Machines

The general idea is that the original input space can always be mapped to some higher dimensional feature space where the training set is separable. Instead of explicitly computing the lifting transformation $\phi(x)$, we define a kernel function K such that

$$K(x, y) = \phi(x) \cdot \phi(y)$$

This gives a nonlinear decision boundary in the original feature space. We treat the distance metric as if it were in the higher feature space without transforming the features to that space.

$$\sum_i \alpha_i y_i \phi(x_i) \cdot \phi(x) + b = \sum_i \alpha_i y_i K(x_i, x) + b$$

Polynomial Kernel:

$$K(x, y) = (c + x \cdot y)^d$$

Gaussian Kernel:

$$K(x, y) = exp\left(-\frac{1}{\sigma^2}\|x - y\|^2\right)$$

These are used so that classifiers can be trained for high dimensional feature spaces. The data may be linearly separable in the high dimensional space, but not necessarily the original feature space.

Support Vector Machines for Classification

1. Pick an image representation.
2. Pick a kernel function for that representation.
3. Compute the matrix of kernel values between every pair of training examples.
4. Feed the kernel matrix into your favorite SVM solver to obtain support vectors and weights.
5. At test time, compute the kernel values for your test example and each support vector, and combine them with the learned weights to vet the value of the decision function.

In general, there are many publicly available packages for perform SVM classification and the kernel-based framework is very powerful and flexible. While it works very well even with small training sample sizes, there is no easy way to do multi-class SVMs and we must combine two-class SVMs. For large scale problems, SVMs can consume a lot of resources since a matrix of kernel values must be computed for each pair of examples.

Evaluating Classifiers

Always train on a training set, evaluate on a validation dataset, and test on a testing set. The test set performance is generally worse than the training set. This prevents overfitting. The total error rate should always be less than 50% for a two class problem. We can also use a confusion matrix or a receiver-operator characteristic curve with different thresholds to evaluate the performance of our classifier.

You can find all my notes at `http://omgimanerd.tech/notes`. If you have any questions, comments, or concerns, please contact me at alvin@omgimanerd.tech