WIKIPEDIA

Cascading classifiers

Cascading is a particular case of ensemble learning based on the concatenation of several classifiers, using all information collected from the output from a given classifier as additional information for the next classifier in the cascade. Unlike voting or stacking ensembles, which are multiexpert systems, cascading is a multistage one.

Cascading classifiers are trained with several hundred "positive" sample views of a particular object and arbitrary "negative" images of the same size. After the classifier is trained it can be applied to a region of an image and detect the object in question. To search for the object in the entire frame, the search window can be moved across the image and check every location with the classifier. This process is most commonly used in image processing for object detection and tracking, primarily facial detection and recognition.

The first cascading classifier was the face detector of Viola and Jones (2001). The requirement for this classifier was to be fast in order to be implemented on low-power CPUs, such as cameras and phones.

Contents

Algorithm properties

Scaling and rotations

It can be seen from this description that the classifier will not accept faces that are upside down (the eyebrows are not in a correct position) or the side of the face (the nose is no longer in the center, and shadows on the side of the nose might be missing). Separate cascade classifiers have to be trained for every rotation that is not in the image plane (side of face) and will have to be retrained or run on rotated features for every rotation that is in the image plane (face upside down or tilted to the side). Scaling is not a problem, since the features can be scaled (centerpixel, leftpixels and rightpixels have a dimension only relative to the rectangle examined). In recent cascades, pixel value from some part of a rectangle compared to another have been replaced with Haar wavelets.

Stage properties

To have good overall performance, the following criteria must be met:

1. Each stage must validate all faces, and can produce many false positives. For example, if stage 1 were to mark as 'does not contain a face' 20% of rectangles containing a face (false negative rate=20%), then the total performance of the chain cannot be higher than 80% true positive, whatever the next stages are, since 20% of faces have already been rejected.
2. This suggests that a good stage must have 100% true positive and for example 40% false positive, that is accept all rectangles containing faces and erroneously mark many rectangles as potentially containing a face, to be eliminated by later stages. For a first stage, 100% true positive and 40% false positive still gives a lot of false negative, if only 1 in a 1000 rectangles in an image contain a face, there will still be 400 to 1 false possible faces after the first stage.
3. If the first stage is very fast (a few operations), we have eliminated 60% of rectangles not containing a face very quickly.

The training procedure for one stage is therefore to have many weak learners (simple pixel difference operators), train them as a group (raise their weight if they give correct result), but be mindful of having only a few active weak learners so the computation time remains low.

The first detector of Viola & Jones had 38 stages, with 1 feature in the first stage, then 10, 25, 25, 50 in the next five stages, for a total of 6000 features. The first stages remove unwanted rectangles rapidly to avoid paying the computational costs of the next stages, so that computational time is spent analyzing deeply the part of the image that have a high probability of containing the object.

Cascade training

Cascades are usually done through cost-aware ADAboost. The sensitivity threshold (0.8 in our example) can be adjusted so that there is close to 100% true positives and some false positives. The procedure can then be started again for stage 2, until the desired accuracy/computation time is reached.

After the initial algorithm, it was understood that training the cascade as a whole can be optimized, to achieve a desired true detection rate with minimal complexity. Examples of such algorithms are RCBoost, ECBoost or RCECBoost. In their most basic versions, they can be understood as choosing, at each step, between adding a stage or adding a weak learner to a previous stage, whichever is less costly, until the desired accuracy has been reached. Every stage of the classifier cannot have a detection rate (sensitivity) below the desired rate, so this is a constrained optimization problem. To be precise, the total sensitivity will be the product of stage sensitivities.

Cascade classifiers are available in OpenCV, with pre-trained cascades for frontal faces and upper body. Training a new cascade in OpenCV is also possible with either haar_training or train_cascades methods. This can be used for rapid object detection of more specific targets, including non-human objects with Haar-like features. The process requires two sets of samples: negative and positive, where the negative samples correspond to arbitrary non-object images. The time constraint in training a cascade classifier can be circumvented using cloud-computing methods.

Cascading classifiers in statistics

The term is also used in statistics to describe a model that is staged. For example, a classifier (for example *k*-means), takes a vector of features (decision variables) and outputs for each possible classification result the probability that the vector belongs to the class. This is usually used to take a decision (classify into the class with highest probability), but cascading classifiers use this output as the input to another model (another stage). This is particularly useful for models that have highly combinatorial or counting rules (for example, class1 if exactly two features are negative, class2 otherwise), which cannot be fitted without looking at all the interaction terms. Having cascading classifiers enables the successive stage to gradually approximate the combinatorial nature of the classification, or to add interaction terms in classification algorithms that cannot express them in one stage.

As a simple example, if we try to match the rule (class1 if exactly 2 features out of 3 are negative, class2 otherwise), a decision tree would be:

- feature 1 negative
  - feature 2 negative
    - feature 3 negative -> class2
    - feature 3 positive -> class1
  - feature 2 positive
    - feature 3 negative -> class1
    - feature 3 positive -> class2
- feature 1 positive
  - feature 2 negative
    - feature 3 negative -> class1
    - feature 3 positive -> class2
  - feature 2 positive
    - feature 3 negative -> class2
    - feature 3 positive -> class2

The tree has all the combinations of possible leaves to express the full ruleset, whereas (feature1 positive, feature2 negative) and (feature1 negative, feature2 positive) should actually join to the same rule. This leads to a tree with too few samples on the leaves. A two-stage algorithm can effectively merge these two cases by giving a medium-high probability to class1 if feature1 or (exclusive) feature2 is negative. The second classifier can pick up this higher probability and make a decision on the sign of feature3.

In a bias-variance decomposition, cascaded models are usually seen as lowering bias while raising variance.

See also

- Boosting (meta-algorithm)
- Bootstrap aggregating

References

Sources

- Gama, J.; Brazdil, P. (2000). "Cascade Generalization". *Machine Learning*. **41** (3): 315–343. CiteSeerX 10.1.1.46.635 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.46.635). doi:10.1023/a:1007652114878 (https://doi.org/10.1023%2Fa%3A1007652114878).
- Minguillón, J. (2002). *On Cascading Small Decision Trees* (http://www.tdx.cat/handle/10803/3027) (PhD thesis). Universitat Autònoma de Barcelona.
- Zhao, H.; Ram, S. (2004). "Constrained Cascade Generalization of Decision Trees". *IEEE Transactions on Knowledge and Data Engineering*. **16** (6): 727–739. CiteSeerX 10.1.1.199.2077 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.199.2077). doi:10.1109/tkde.2004.3 (https://doi.org/10.1109%2Ftkde.2004.3).