Learning to find a face

How do you teach a computer to find a face in a photo? You show it examples of faces and non-faces, and train it to distinguish between the two types of images. That’s a basic classification task in the field of machine learning. There are several ways to do it, and support vector machines is a relatively new technique that has already proven to perform very well.

The feature space trick

Training a support vector machine can be thought of as finding a separating plane that correctly divides example points into two classes.

Linear separation doesn’t always exist. But what if we had a universal way to map our examples into a space where they became separable?

Training for accuracy

You take some examples of faces and non-faces and train a function that separates those examples. You then run this function on an arbitrary image and it “finds” faces in places you would least expect.

So how do you become good at recognizing something? You learn from your mistakes! If you do several rounds of training and include false hits as negative examples in the next round, accuracy significantly increases.

Recipe for an eye detector

Training

» Consider 20x20 pixel grayscale images as your samples
» Take 2000 samples of eyes and 10000 random samples
» Do 4-5 rounds of training, testing the function after every round
» Use false hits from testing as non-eye samples

Using

» Run a 20x20 pixel window over different scales of the target image
» Evaluate the function for every window position
» Treat any matches as eyes!

What’s next?

As a practical application we are building an online face detection system that will allow automated tagging of people in photos. But apart from the implementation challenges there’s a number of areas that we hope to contribute to.

Speed

Support vector machines in their raw form are quite slow. What’s the best way of reducing their complexity without losing accuracy?

Accuracy

Rather than using generic mappings into feature space, could we find one that separates the given training data best? How much would the accuracy improve?

What are these ‘support vectors’ after all?

Think of the samples you are trying to separate as vectors. The plane that separates the points representing the vectors can be expressed as a combination of some of those vectors! In other words, your detection function is defined by some (but not all) of the examples that it was trained on. Those important samples are the support vectors!