Tutorial on Support Vector Machines, Edge Detection, Image Pyramids, and Corner Detection

Ioan Andrei Bârsan
iab@cs.toronto.edu

CSC420: Introduction to Image Understanding

September 28, 2017

Outline

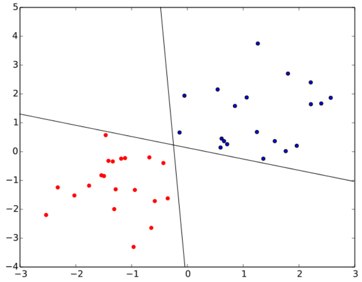
Classification Recap

- We focus on **binary** classification.
    - Examples: Edge vs. not edge, spam vs. non-spam, cat vs. dog, etc.
- Train time:
    - Start with lots of examples whose labels we know in advance.
    - Each data point can be seen as a vector $x \in \mathbb{R}^n$.
    - Each label is (in our case) a binary label $y \in \{0, 1\}$.
    - Learn a model, which is just a set of weights, by minimizing an error function
- Test time:
    - Receive a new data point $x' \in \mathbb{R}^n$ with an unknown label.
    - Classify the data point using the trained model (the specifics will be described soon).

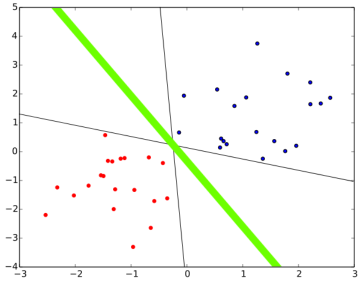
Not covered in this course: detailed mathematical formulation, kernels, and non-linear SVMs.

Support Vector Machines

- Support vector machines (SVMs) are one of the many tools for performing classification.
- Key insight: integrate generalization capacity into formulation.
- Other (older) algorithms such as Perceptron are OK with simply finding ANY way of separating two classes, which leads to poor generalization.



Two possible (wildly different) solutions computed by Perceptron (unstable, **bad**).



Robust solution produced by an SVM, overlaid in green (stable, **good**).

SVMs: How-to

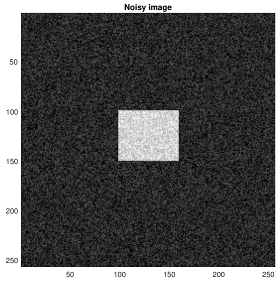
- We will be using MATLAB for consistency with the assignments.
- Multiple ways of doing it
    - `libsvm`'s MATLAB wrappers (covered in this tutorial)
        - + trains much faster
        - + overall better performance
        - – ugly API
    - MATLAB's built-in SVM support
        - + nicer API
        - – more difficult to tune

In Python

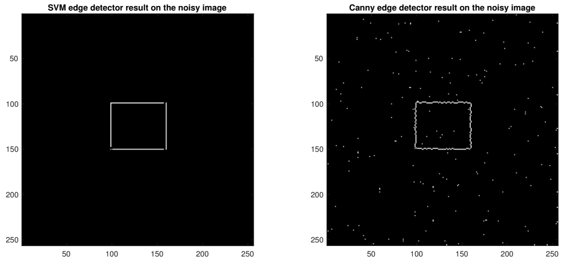
The most popular way of working with SVMs in Python is via the `scikit-learn` module. The `sklearn.svm.SVC` class is a good starting point if you want to explore on your own.

Application: SVMs for edge detection

- Motivation: Traditional methods (e.g., Canny) are powerful but not very robust to challenges such as noise.



Toy example of a noisy image.



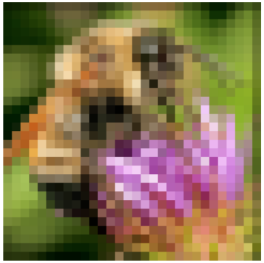
Results using a machine learning-based edge detector (left) and the classic Canny edge detector (right).

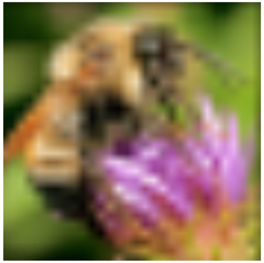
MATLAB demo

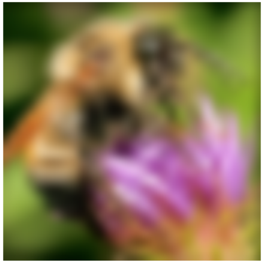
- MATLAB demo: SVMs for edge detection on a toy example (using `libsvm`).
- See `edgeDetection.m` in the handout code (the most up-to-date version will be posted after the tutorial).

Interpolation and Pyramids

- Naive image down- and up-sampling can produce unpleasant artifacts.
- Using even very simple interpolation methods can already reduce these issues by a substantial margin.


Nearest-neighbor interpolation


Bilinear interpolation


Bicubic interpolation

[Source: N. Snavely]

Beyond Interpolation: Super-resolution

- ... but can we do even better than simple interpolation? Yes! Deep learning to the rescue!
- We can train neural networks to predict high-resolution patches (e.g., 64×64) from lower-resolution patches (e.g., 16×16) and "hallucinate" details from low-res images.

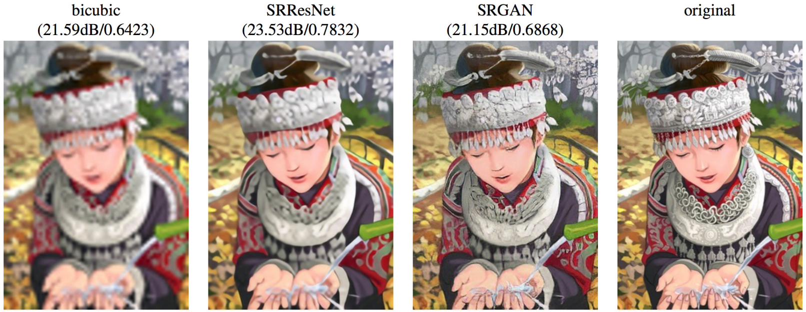
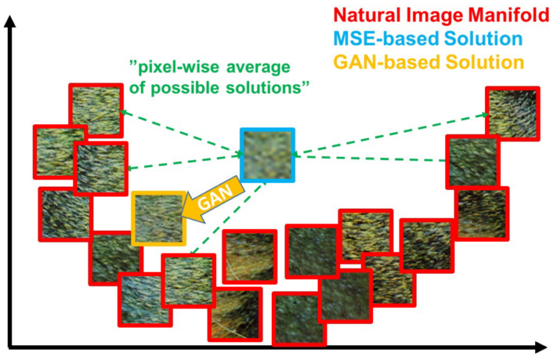


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

Illustration from: Ledig et al., 2016 (`http://arxiv.org/abs/1609.04802`).

Super-resolution (continued)

- Super-resolution = learn to map low-resolution patches to high-resolution ones.
- Here, we see a set of possible high-resolution matches corresponding to a low-resolution patch.
- Under the hood, the systems use many different approaches for selecting the best possible patch (in this paper, generative adversarial networks).



Super-resolution (continued)

- Super-resolution neural networks essentially learn what certain textures look like at low resolution, so they can recognize them in new images and reconstruct them.
- Training data is very easy to generate: just grab a huge dataset of images (e.g., from a web crawl), and downsample them. You now have (low-res, high-res) pairs of each image, which you can use to train a neural network.
- Good starting point with nice documentation (Python): `https://github.com/alexjc/neural-enhance`

Image Pyramids

- Idea: run a CV algorithm, like template matching, on multiple scales of the original image.
- Widely used in practice: many computer vision algorithms operate in a *coarse-to-fine* manner, starting on a low-resolution version of the input image, and then moving to a higher-resolution one, using the previous (rough) solution as an initialization. Examples include:
  - object detection
  - visual odometry / camera tracking
  - optical flow
- Now let's see some examples in MATLAB (`interpAndFeature.{m,mlx}`).

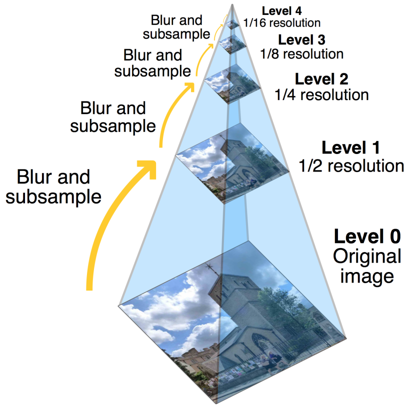
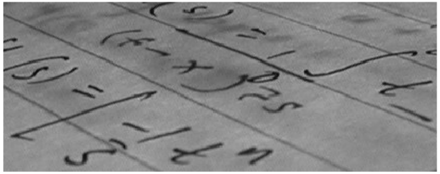


Image source: Wikipedia

Bonus: Corner Detection

- We want to figure out what the interesting/distinctive parts of an image are.
- In the lecture, we saw why corners are a good place to start, since they are easier to match across images (less ambiguity; cf. aperture problem slides).
- Note that *detecting* keypoints only solves half the problem, as we'll see in next week's lectures; we still need a way to *describe* the areas around the keypoints.



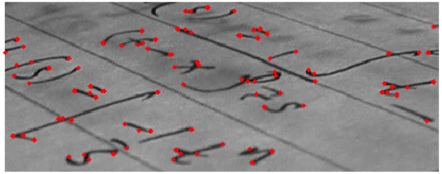
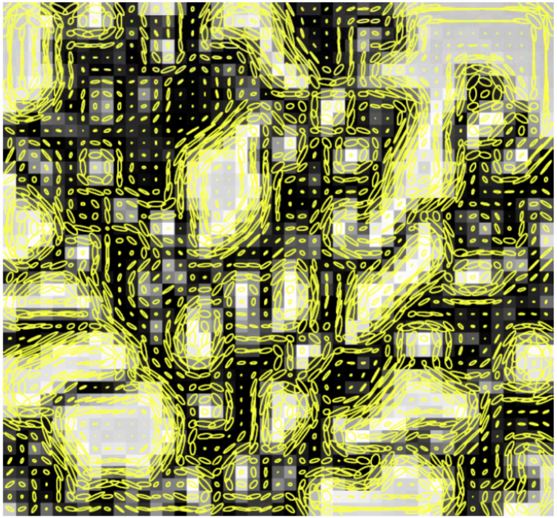


Image source: Wikipedia

Visualizing the Second Moment Matrices

Each pixel has its own corresponding 2×2 second moment matrix, and here we visualize the matrices' eigenpairs as ellipses.

Image credits: Prof. Margarita Chli, ETH Zurich



Further Reading

- **Question:** Are keypoint detectors still relevant in the age of deep learning? **Answer:** YES, especially in robotics!
    - State of the art simultaneous localization and mapping: ORB-SLAM, ORB-SLAM2 (`http://webdiis.unizar.es/~raulmur/orbslam/`).
    - SIFT, SURF features still used in tasks like panorama stitching and place recognition.
    - Autonomous drone and UAV navigation (e.g., FOVIS), etc.
- More details on the SVM internals, part of MIT 6.034 Artificial Intelligence, very well-explained: `https://www.youtube.com/watch?v=_PwhiWxHK8o`
- The official scikit-learn tutorial is a very good way of getting started with machine learning in Python: `http://scikit-learn.org/stable/tutorial/basic/tutorial.html`. `scikit-image` and `opencv-python` are well-established image processing libraries.