CSSE463: Image Recognition Day 27

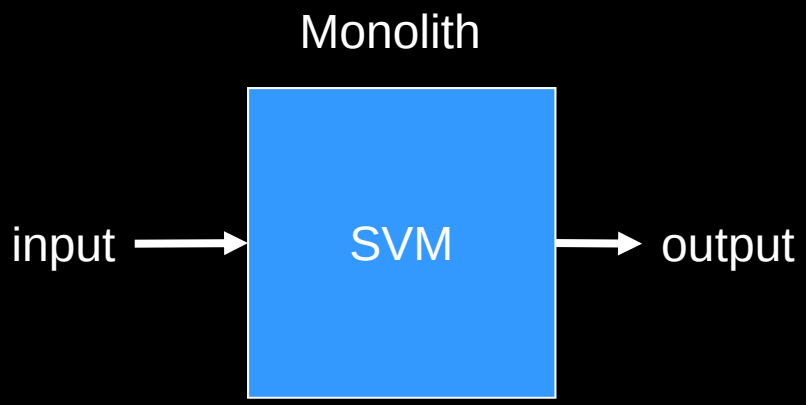
- This week
  - Last night: k-means lab due.
  - Today*:* Classification by "boosting"

    Yoav Freund and Robert Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Proceedings of the 2nd European Conference on Computational Learning Theory*, March, 1995.

    http://www.cs.princeton.edu/~schapire/publist.html

  - Sunday night: project plans and prelim work due
- Questions?

Motivation for Adaboost



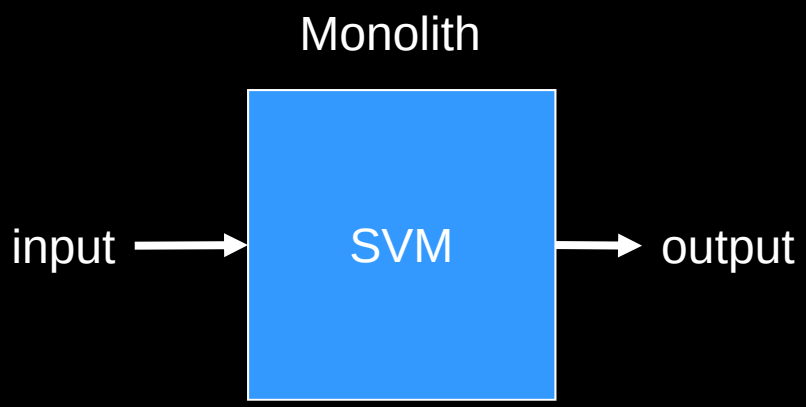
- SVMs are fairly expensive in terms of memory.
  - Need to store a list of support vectors, which for RBF kernels, can be long.

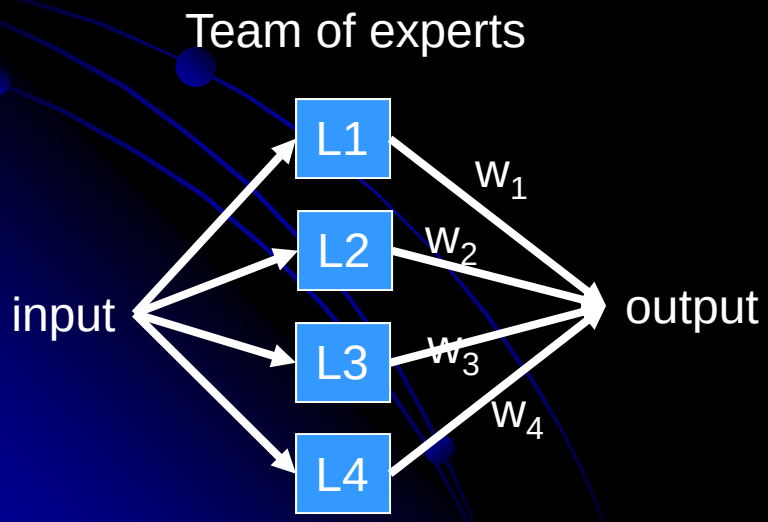
By the way, how does svmfwd compute y1?
y1 is just the weighted sum of contributions of individual support vectors:
d = data dimension, e.g., 294

$$y1 = \sum_{i=1}^{numSupVecs} \left(svcoeff_i * e^{(1/d\sigma)*\|x - sv_i\|^2} \right) + bias$$

numSupVecs, svcoeff (alpha) and bias are learned during training.
BTW: looking at which of your training examples are support vectors can be revealing! (Keep in mind for term project)

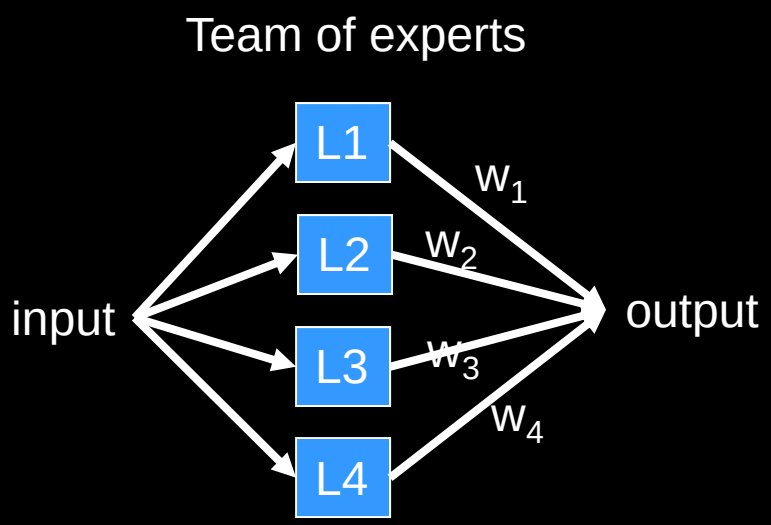
Motivation for Adaboost





- SVMs are fairly expensive in terms of memory.
  - Need to store a list of support vectors, which for RBF kernels, can be long.
- Can we do better?
  - Yes, consider simpler classifiers like thresholding or decision trees.
  - The idea of boosting is to combine the results of a number of “weak” classifiers in a smart way.

Idea of Boosting (continued)



Consider each weak learner as an expert that produces “advice” (a classification result, possibly with confidence).

We want to combine their predictions intelligently.

Call each weak learner L multiple times with a different distribution of the data (perhaps a subset of the whole training set).

Use a weighting scheme in which each expert’s advice is weighted by its accuracy on the training set.

- Not memorizing the training set, since each classifier is weak

Notation

- Training set contains labeled samples:
  - $\{(x_1, C(x_1)), (x_2, C(x_2)), \ldots (x_N, C(x_N))\}$
  - $C(x_i)$ is the class label for sample x_i, either 1 or 0.
- *w* is a weight vector, 1 entry per example, of how important each example is.
  - Can initialize to uniform (1/N) or weight guessed "important" examples more heavily
- *h* is a hypothesis (weak classifier output) in the range [0,1]. When called with a vector of inputs, h is a vector of outputs

Adaboost Algorithm

Initialize weights $w_i^1 = \frac{1}{N}$

for t = 1:T

1. Set $\vec{p}^t = \vec{w}^t / \sum_{i=1}^{N} w_i^t$ Normalizes so p weights sum to 1

2. Call weak learner L_t with x, and get back result vector $\vec{h}_t$

3. Find error ε_t of $\vec{h}_t$: $\varepsilon_t = \sum_{i=1}^{N} p_i^t \left| h_t(x_i) - c(x_i) \right|$ Average error on training set weighted by p

4. Set $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$ Low overall error → β close to 0; (big impact on next step)
error almost 0.5 → β almost 1 (small impact " " ")

5. Re-weight the training data $w_i^{t+1} = \begin{cases} w_i^t & if\ incorrect \\ w_i^t \beta_t & if\ correct \end{cases}$

Weighting the ones it got right less so next time it will focus more on the incorrect ones

Output answer:

$$h_f(x_i) = \begin{cases} 1 & if \sum_{t=1}^{T} \left(\left(\log \frac{1}{\beta_t} \right) h_t(x_i) \right) \geq \frac{1}{2} \sum_{t=1}^{T} \log \frac{1}{\beta_t} \\ 0 & otherwise \end{cases}$$

Adaboost Algorithm

Output answer:

$$h_f(x_i) = \begin{cases} 1 & if \sum_{t=1}^{T} \left(\left(\log \frac{1}{\beta_t} \right) h_t(i) \right) \geq \frac{1}{2} \sum_{t=1}^{T} \log \frac{1}{\beta_t} \\ 0 & otherwise \end{cases}$$

- Combines predictions made at all T iterations.
- Label as class 1 if weighted average of all T predictions is over ½.
- Weighted by error of overall classifier at that iteration (better classifiers weighted higher)

Practical issues

- For binary problems, weak learners have to have at least 50% accuracy (better than chance).
- Running time for training is O(*nT*), for *n* samples and T iterations.
- How do we choose T?
  - Trade-off between training time and accuracy
  - Best determined experimentally (see optT demo)

Multiclass problems

- Can be generalized to multiple classes using variants Adaboost.M1 and Adaboost.M2
- Challenge finding a weak learner with 50% accuracy if there are lots of classes
  - Random chance gives only 17% if 6 classes.
  - Version M2 handles this.

Available software

- GML AdaBoost Matlab Toolbox
- Weak learner: Classification tree
- Uses less memory if weak learner is simple
- Better accuracy than SVM in tests (toy and real problems)
- Demo
- Sample use in literature