MACHINE LEARNING AND PATTERN RECOGNITION

Fall 2004, Lecture 11:

Learning Theory, Ensemble Methods, Boosting, VC-Dimension.

Yann LeCun
The Courant Institute,
New York University
http://yann.lecun.com

Ensemble Methods

- Democracy applied to decision making: 1,000,000 Lerning Machines Can't Be Wrong.
- Generate many learning machines that are slightly different from each other, and make them vote.
- This can be viewed as a way to perform regularization
- Example: Bayesian learning. The outputs produced by replicas of the machine for each possible value of the parameter vector are added with weights that reflect the conditional probability of the parameter vector given the training dataset.
- The key idea of ensemble methods: find a way to make each replica of the learning machine different from the others, yet useful.

Full Bayesian Learning: keep'em all

Training set: $\mathcal{X} = (X^1, X^2, \dots X^p)$, and $\mathcal{Y} = (Y^1, Y^2, \dots Y^p)$
Full Bayesian prediction: average answers for all W, weighted by $P(W|\mathcal{X}, \mathcal{Y})$:

$$P(Y|X, \mathcal{X}, \mathcal{Y}) = \int P(Y|X, W) P(W|\mathcal{X}, \mathcal{Y}) dW$$

Where

$$P(W|\mathcal{S}) = P(W|\mathcal{Y}, \mathcal{X}) = \frac{P(\mathcal{Y}|\mathcal{X}, W) P(W|\mathcal{X})}{P(\mathcal{Y}|\mathcal{X})}$$

where the denominator is a normalization term:

$$P(\mathcal{Y}|\mathcal{X}) = \int P(\mathcal{Y}|\mathcal{X}, W) P(W|\mathcal{X}) dW$$

that ensures that $\int P(W|\mathcal{S}) dW = 1$.
this is generally intractable. Approximations must be used: Laplace, Sampling, Variational approximations...

Variational Bayesian

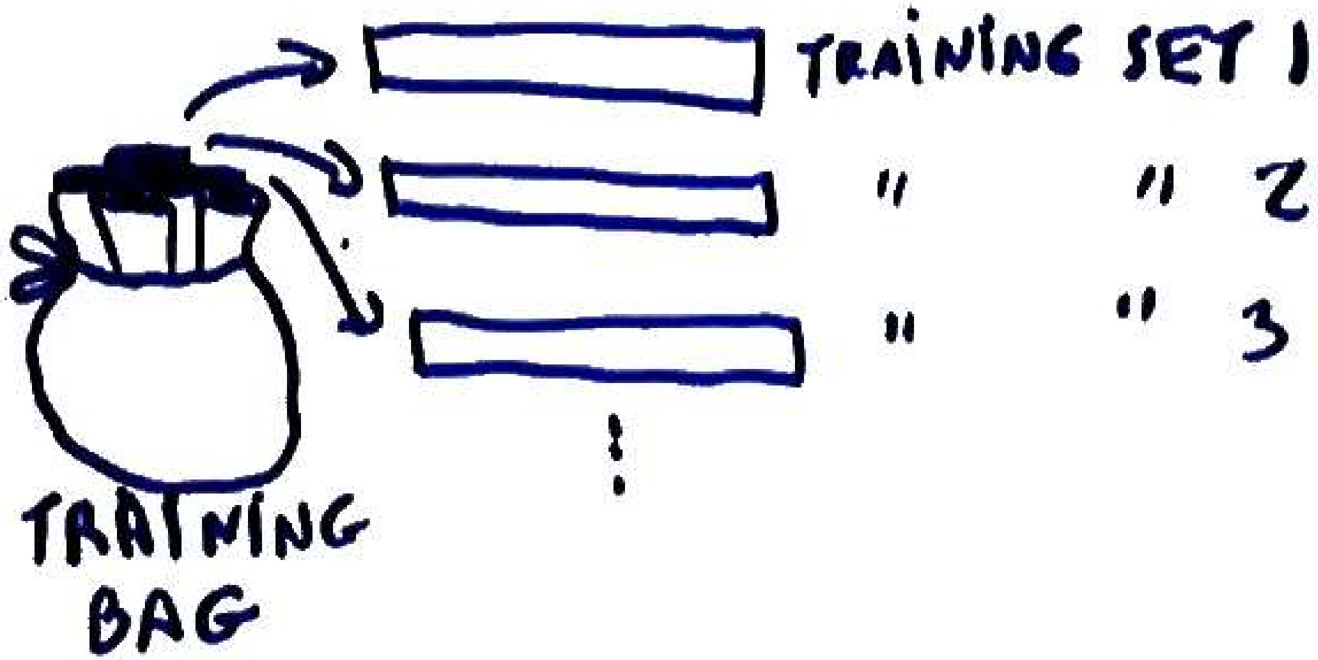
Main idea: approximate $P(W|\mathcal{S})$ with some parameterized model that we can actually compute.
Choose $P(W|\mathcal{S})$ in a family so that it best approximates the real $P(W|\mathcal{S})$ in Kulback-Leibler terms.
Replace the log likelihood by a variational free energy (just like with EM):

$$-\log P(Y|X,\mathcal{X},\mathcal{Y}) = -\log \int P(Y|X,W)P(W|\mathcal{X},\mathcal{Y})dW = -\log \int P(Y|X,W)Q(W)P($$

$$-\log P(Y|X,\mathcal{X},\mathcal{Y}) < -\int Q(W) \log[P(Y|X,W)P(W|\mathcal{X},\mathcal{Y})/Q(W)]dW$$

More details at `http://variational-bayes.org`

Bootstrap/Bagging



- A simple idea: use different subsets of the training set.
- The Bootstrap method consists in generating multiple training set by drawing samples from the original training set *with replacement*.
- The Bagging method consists simply in averaging of the outputs produced by each of the instances.

Boosting

- Train a machine on the training set.
- construct a new training set in which samples that the first machine got wrong have a higher weight (or probability of being picked).
- train another copy of the machine on this set
- make the overall output of the system a weighted sum of all the trained models where the weights are higher for machines that make less mistakes.
- iterate
- This technique trains new machines on samples that the previously trained machines found difficult.
- This is a way to build multiple “experts” that specialize on different types of patterns.

Adaboost

- 0. given a training set $(X^1, y^1), ...(X^P, y^P)$
- 1. Initialize the observation weights $v_i = 1/P, i = 1, 2, ..., P$.
- 2. For $m = 1$ to M do:
  - (a) Train a classifier $G_m(X)$ on the training set by drawing each sample i with probability v^i.
  - (b) compute $E_m = \sum_{i=1}^{P} v_i \{y^i \neq G_m(X^i)\}$
  - (c) compute $\alpha_m = \frac{1}{2} log((1 - E_m)/E_m)$
  - (d) set $D_i \leftarrow v_i \exp[\alpha_m \{y^i \neq G_m(X^i)\}]$ and, $v_i = \frac{D_i}{\sum_{i=1}^{P} D_i}$ for $i = 1, 2, ..., P$.
- output $G(X) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(X)\right)$

Adaboost = Greedy Gradient Descent

- Adaboost greedily minimizes $L = \sum_{i=1}^{P} \exp\left(-y^i F(X^i)\right)$ assuming that $F(X)$ is a linear combination of elementary classifiers $F(X) = \sum_{m=1}^{M} \alpha_m G_m(X)$.
- Let's assume that we have already trained the first $M-1$ elementary classifiers (called *weak learners*).
- How should we train the M-th classifier, and what value should we give α_M so as to maximally decrease the loss exponential $\mathcal{L}_{\mathrm{exp}}$.

$$\mathcal{L}_{\mathrm{exp}} = \sum_{i=1}^{P} \exp\left(-y^i \sum_{m=1}^{M} \alpha_m G_m(X)\right)$$

Adaboost = Greedy Gradient Descent

$$\mathcal{L}_{\exp} = \sum_{i=1}^{P} \exp\left(-y^i \sum_{m=1}^{M} \alpha_m G_m(X)\right)$$

This loss can be decomposed as follows:

$$\mathcal{L}_{\exp} = \sum_{i=1}^{P} \exp\left(-y^i \sum_{m=1}^{M-1} \alpha_m G_m(X)\right) \exp\left(-y^i \alpha_M G_M(X)\right)$$

$$\mathcal{L}_{\exp} = \sum_{i=1}^{P} D^i_{M-1} \exp\left(-y^i \alpha_M G_M(X)\right)$$

Finding the $G_M(X)$ that mininizes $\mathcal{L}_{\exp}$ can be performed by giving each training a sample a weight v_i proportional to D^i_{M-1} and training the M-th copy of the weak learner on this re-weighted training set. v_i is simply a normalized version of D^i: $v_i = D^i_{M-1} / \sum_{i=1}^{P} D^i_{M-1}$

Computing the combination weight

Now we need to compute the combination weight for the new weak learner α_M. We will choose the value that minimizes the loss. To find the optimal value, we differentiate the loss:

$$\frac{\partial \mathcal{L}_{\exp}}{\partial \alpha_M} = -\sum_{i=1}^{P} D^i_{M-1} y^i G_M(X) \exp\left(-y^i \alpha_M G_M(X)\right) = 0$$

We can decompose the sum into two terms, those for which the weak learner gave the right answer ($y^i G_M(X) = 1$), and those for which it was wrong ($y^i G_M(X) = -1$):

$$\sum_{i \in \mathrm{right}} D^i_{M-1} \exp\left(-\alpha_M\right) + \sum_{i \in \mathrm{wrong}} -D^i_{M-1} \exp\left(\alpha_M\right) = 0$$

Computing the combination weight

Since each training sample was weighted by D^i_{M-1}, the sum $\sum_{i \in \text{wrong}} -D^i_{M-1}$ is simply the expected number of mistakes made by the weak learner on the weighted training set P_{wrong}:

$$P_{right} \exp\left(-\alpha_M\right) - P_{\text{wrong}} \exp\left(\alpha_M\right) = 0$$

Or

$$\left(P - P_{\text{wrong}}\right) \exp\left(-\alpha_M\right) - P_{\text{wrong}} \exp\left(\alpha_M\right) = 0$$

where P is the total size of the training set. Solving for α_M, we get:

$$\alpha_M = \frac{1}{2} log \frac{P - P_{\text{wrong}}}{P_{\text{wrong}}} = \frac{1}{2} log \frac{1 - E_M}{E_M}$$

Where E_M is the proportion of errors made by the M-th learning on the training set weigthed by the D^i_{M-1}.

An Overview of Learning Theory

The main purpose of Learning Theory:

- Give formulae and techniques to predict/estimate the generalization error.
- Give well-principled methods to minimize the generalization error.
- Most formulae for the expected risk (generalization error) are of the form:

$$E_{\text{expected}} = E_{\text{train}} + c\frac{h}{P^{\alpha}}$$

Where c is a constant, P is the number of training samples, $alpha$ is a constant between 0.5 and 1.0, and h is a measure of *capacity* of the family of functions implemented by the model.

Estimating the Expected Risk

- PROBLEM 1: In fact, many formulae are probabilistic upper bounds rather than equalities: "with probability $1 - \delta$":

$$E_{\text{expected}} < E_{\text{train}} + c(\delta)\frac{h}{P^{\alpha}}$$

- PROBLEM 2: The bound formulae derived from general learning theories are generally extremely loose, so you can't just plug in numbers and hope that the formula will give you useful information. That is because general formulae cannot take into account the peculiarities of the problem at hand.
- PROBLEM 3: The "capacity" term h is generally impossible to know for a complex model family. Intuitively, h is akin to some sort of *effective number of free parameters* (which may or may not reflect the actual number of free parameters).

Bayesian Formula for Model Selection

- Bayesian Information Criterion (BIC):

$$E_{\text{expected}} = E_{\text{train}} + \frac{log(P)}{2} \frac{h}{P}$$

- can be derived from a Bayesian inference formula with a so-called "Laplace" approximation (2nd order taylor expansion of the log likelihood around the MLE estimate).

Structural Risk Minimization

- Structural Risk Minimization: these are very general bounds derived from first principles. The bad news is that they are distribution free, and therefore very loose.
- SRM bound for classification: with probability $1 - \delta$:

$$E_{\mathrm{expected}} \leq E_{\mathrm{train}} + \frac{\epsilon}{2}\left(1 + \sqrt{1 + \frac{4E_{\mathrm{train}}}{\epsilon}}\right)$$

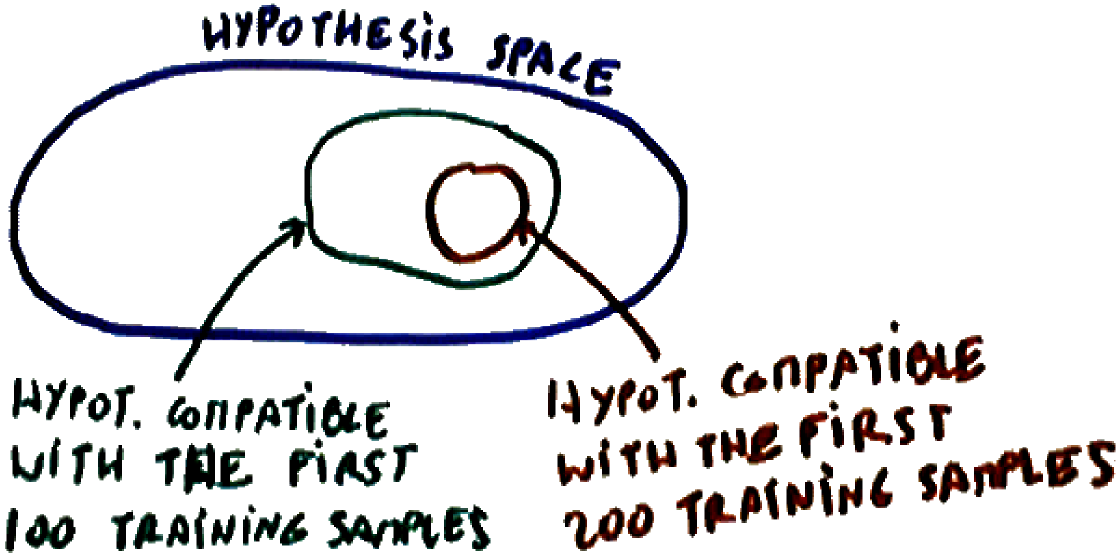
with $\epsilon = a_1 \frac{h[log(a_2 P/h)+1] - log(\delta/4)}{P}$, where a_1, a_2 are constant not furnished by the theory, and h is the *Vapnik-Chervonenkis Dimension* of the family of function.

- SRM bound for regression

$$E_{\mathrm{expected}} \leq \frac{E_{\mathrm{train}}}{(1 - c\sqrt{(}\epsilon))_+}$$

with the same ϵ as above.

An Example of How to get a Bound



- The hypothesis space of our model is the family of all possible functions indexed by our parameter W:

$$\mathcal{H} : \{F(Y, X, W), \forall W\}$$

  That set may have an infinite (possibly uncountable) cardinality.
- One view of learning: each training example eliminates elements of the hypothesis space that disagree with it.

An Example of How to get a Bound

- Let's assume that the number of hypotheses is finite and equal to k: $W_1, W_2, ...W_k$.
- Let's assume the loss is binary: 0 if correct, 1 if error.
- Let's define the expected error for a particular hypothesis $E_{exp}(W_i)$, the error that W_i over an infinite training set.
- Let's define the empirical error for P examples: $E^P_{emp}(W_i)$, the error that W_i over a particular training set of size P.
- The Hoeffding/Chernoff bound tells us:

$$P(|E_{exp}(W_i) - E^P_{emp}(W_i)| > \epsilon) < 2\exp(-2\epsilon^2 P)$$

- The Hoeffding/Chernoff bound is a wonderful formula that tells us how fast the average of a variable computed over a finite set of sample converges to the true expectation (average over an inifinte number of samples) as we increase the number of samples.

Bound on the Expected Error

Since the probability of the union of a set of events is bounded above by the sum of the individual event probabilities, we can write:

$$P(\exists W_i, |E_{exp}(W_i) - E^P_{emp}(W_i)| > \epsilon) \leq \sum_{i=1}^{k} 2\exp(-2\epsilon^2 P) = 2k\exp(-2\epsilon^2 P)$$

this can be rewritten as

$$P(\forall W_i, |E_{exp}(W_i) - E^P_{emp}(W_i)| \leq \epsilon) > 1 - 2k\exp(-2\epsilon^2 P)$$

This is a uniform convergence bound, because it holds for all hypotheses.

Bound on the Expected Error

Let's define $\delta = 2k \exp(-2\epsilon^2 P)$. Suppose we hold P and δ, and solve for ϵ, we get with probability $1 - \delta$:

$$E_{exp}(W) \leq E_{emp}^P(W) + \sqrt{\frac{1}{2P} \log \frac{2k}{\delta}}$$

In particular, the previous inequality is true for the hypothesis W_{train} that minimizes the training set error:

$$E_{exp}(W_{\mathrm{train}}) \leq E_{emp}^P(W_{\mathrm{train}}) + \sqrt{\frac{1}{2P} \log \frac{2k}{\delta}}$$

VC-dimension

- The previous bound assumed the space of hypotheses was finite (with k hypotheses).
- The Vapnik-Chervonenkis approach derives similar results for infinite hypotheses spaces.
- key idea: as far as we are concerned, two hypotheses are identical if they produce the same classification on our dataset: identical hypotheses are put into equivalence classes.
- The formulae we obtain are the SRM formulae shown a few slides back where h is the VC-dimension of the family of functions
- The VC-dim is defined as the largest number of points (in any position) that our family of function could classify in every possible ways.
- EXAMPLE: The VC-dim of linear classifier with N inputs is $N + 1$: in dimension 2, there is a set of 3 points on which all 2^3 dichotomies are linearly separable. There is no such set with 4 points.

VC-dimension, continued

- The VC-dim and the SRM formulae are very interesting conceptually, because they are derived with a minimal set of assumptions.
- The VC theory allows us to “derive” and quantify the intuitive notion of Occam’s Razor from *first principles*.
- Occam’s Razor is the idea that simple models are preferable to complex one.
- Until the VC theory, Occam’s Razor was assumed to be a good idea, or was derived from rather contrived sets of assumptions.

How Can we get the Constants?

Recall that a good general form for the generalization error is:

$$E_{\text{expected}} = E_{\text{train}} + c\frac{h}{P^\alpha}$$

The constants c depends on the task and the learning algorithm used, h depends on the model, and $0.5 \leq \alpha \leq 1$ depends on the type of task (e.g. regression versus classification).
We can *measure* the constants c and α for a particular task by running a learning machine with a known h on the task (e.g. a linear classifier for which $h = N + 1$) with various size of the training set.
Then, if we assume c is constant from model to model, we can measure h for unknown learning machine by running on several size of training set and fitting the curve.
This process is rather long and expensive, and rarely worth it.