The Learning Problem

- We are stuck with:
  1. Input Space $\mathcal{X}$ (say $\Re^n$)
  2. Output Space $\mathcal{Y}$ (say $\Re$ or $\{-1, 1\}$)
  3. Input Distribution $P_{\mathcal{X}}(x)$
  4. Target Density $F_{\mathcal{Y}}(y|x)$
- We are given a *training set* $(x_1, y_1), \ldots, (x_\ell, y_\ell)$ of $\mathbf{i.i.d}$ examples from $P(x, y) = P(x) \cdot F(y|x)$
- Goal: approximate the target density $F(y|x)$ "as well as possible" based only on training set
- We propose a *Learning Machine*:
  1. Set of functions $f_\alpha(x)$ parameterized by $\alpha \in \Lambda$ (the Hypothesis Space) (α may not $\in \Re^n$)
  2. Learning algorithm $LA : \{\mathcal{X} \times \mathcal{Y}\}^\ell \to \Lambda$ which takes the training set and produces a "guess" $f_\alpha(x)$ that we hope "best approximates" the target conditional distribution
- Key Desire:
  Succeed even when the hypothesis space is very big and the training set is very limited.

What does "Best" mean ?

- Define a *loss function* $L(y, \hat{y}, x)$ which says how bad it is to guess $\hat{y}$ when the true input and output are x and y
- Now "Best" $\Leftrightarrow$ minimize the expected "risk":
$$\min_{\alpha \in \Lambda} R(\alpha) = \int L(y, f_\alpha(x), x)\, dP(x, y)$$
- New Goal: find α_{opt} that minimizes $R(\alpha)$ given that $P(x)$ and $F(y|x)$ are *unknown* and we have only training data $(x_1, y_1), \ldots, (x_\ell, y_\ell)$
- Note 1: "Test data" ,i.e. expectation of $R(\alpha)$, is taken over the same distribution as training data
- Note 2: $F(y|x)$ includes learning functions with and without noise
- Note 3: $L(\cdot)$ might in general depend on x
i.e. some parts of the space might be more "sensitive" than others
- Note 4: "Best" has nothing to do with effort we put into learning. Assume machine has unlimited resources (time,memory).

Input-Output Distinction is Unnecessary

- Define a space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and consider examples $z_1, \ldots, z_\ell = (x_1, y_1), \ldots, (x_\ell, y_\ell)$ drawn i.i.d from $P_{\mathcal{Z}}(z)$.
- The loss function is now $Q_\alpha(z) = L(y, f_\alpha(x), x)$ and our goal is now:

$$\min_{\alpha \in \Lambda} R(\alpha) = \int Q_\alpha(z)\, dP(z)$$

- Some examples:
  1. *Classification*: $z = (x, y) \quad y \in \{-1, 1\}$
     $Q_\alpha(z) = 1$ if $y \neq f_\alpha(x)$ 0 otherwise
     then $R(\alpha)$ is probability of misclassification
  2. *Regression*: $z = (x, y) \quad y \in \Re$
     $Q_\alpha(z) = (y - f_\alpha(x))^2$
     then $\min R(\alpha) \to \int y\, dF(y|x)$
     (or closest from hypothesis class in L_2 metric)
  3. *Density Estimation*: $z = x$
     $Q_\alpha(z) = -\log p_\alpha(z)$
     for p_α a set of parameterized densities

Inductive Inference and Generalization

- Any solution to the learning problem is called a method of *inductive inference*. To construct a *learning algorithm* one must implement a particular method (ERM,Bayesian,MDL,etc..)
- Learning *theory* tries to analyze different methods of inference and recommend one that generalizes best for a fixed training set size. In order to do this it must address the question of generalization.
- The key idea of generalization:
  Given any particular attributes or conditions that our algorithm requires our best guess to satisfy there may be many values of α (i.e. many functions $Q_\alpha(z)$ or $f_\alpha(x)$) that are "compatible" with those restrictions.
- In order to estimate the generalization we can either: *average* over all compatible α's or take the *worst case* α that is compatible.
- Surprisingly, certain methods of inference can only be analyzed in one of these ways.

How Should We Do Learning ?

- Q: Hopeless since we do not know $P_{\mathcal{Z}}(z)$?
  A: No, because we have $z_1, \ldots, z_\ell$ that are drawn i.i.d from $P_{\mathcal{Z}}(z)$.
- One approach [Fischer '52, Parzen '62]:
  1. Use $z_1, \ldots, z_\ell$ to estimate $P_{\mathcal{Z}}(z)$
  2. Minimize $R(\alpha)$ directly with this approximation
  3. Very hard because (1.) needs lots of data and (2.) needs lots of effort
- The Bayesian approach:
  1. Consider several "sensible" hypothesis classes.
  2. For each class, find the posterior distribution over α and "guess" the weighted average.
  3. Compare hypothesis classes by their evidence.
- Many others:
  - MDL Principle [Rissenan '78]
  - Regularization Theory [Tikhonov '63, others]
  - Stochastic Approximation [Robbins & Monroe '51, Amari '67]
  - Algorithmic Complexity [Kolmogorov '65, Chaitin '66, Solmonoff '60]

Empirical Risk Minimization

- Vapnik & Chervonenkis ('74) proposed the method of *structural risk minimization* (SRM) motivated by a detailed analysis of the common *empirical risk minimization* (ERM) method:

$$\min_{\alpha \in \Lambda} R_{emp}(\alpha) = \sum_{i=1}^{\ell} Q_\alpha(z_i)$$

  1. This gives α_ℓ "near" α_{opt} (we hope).
  2. Use $Q_{\alpha_\ell}(z)$ or $f_{\alpha_\ell}(x)$ as our "best guess"

- The method of ERM is not new:
  Least Squares is ERM for Regression loss fn.
  Maximum Likelihood is ERM for Density loss fn.
  Minimizing the number of training errors (e.g. in ANN) is ERM for Classification loss fn.

- Key Questions about ERM:
  1. Does it work asymptotically, i.e. does $\alpha_\ell \to \alpha_{opt}$ as $\ell \to \infty$ [consistency,bias]?
  2. How fast does $\alpha_\ell \to \alpha_{opt}$ as a fn. of ℓ [convergence,variance]?
  3. When is it possible to ensure (1) and to make (2) fast?

Question

- Why should we prefer empirical risk minimization and the resulting structural risk minimization over other methods of inference (e.g. Bayesian)?
- Are there provable conditions under which ERM & SRM will be superior to or equivalent to other methods?
- Does the mere existence of detailed theoretical analyses for these methods make them more attractive to pursue?
- See [Vapnik '95] p.32 for an unsatisfactory discussion of these questions.

CONSISTENCY

- Empirical Risk Minimization is *consistent* for a hypothesis class $Q_\alpha(z)\ \alpha \in \Lambda$ and an input distribution $P_{\mathcal{Z}}(z)$ if:
  $\alpha_\ell \to \alpha_{opt}$ as $\ell \to \infty$ AND
  $R_{emp}(\alpha_\ell) \to R(\alpha_{opt})$ as $\ell \to \infty$
  in other words, we guess the *right* function AND we get the *right* estimate of the true risk
- KEY THEOREM [Vapnik & Chervonenkis '89]: Consistency $\Leftrightarrow$ Uniform convergence
- Uniform convergence means that each $R_{emp}(\alpha)$ converges to the true $R(\alpha)$ uniformly over all $\alpha \in \Lambda$ as $\ell \to \infty$ formally,
  ERM is consistent for $Q_\alpha(z)\ \alpha \in \Lambda, P_{\mathcal{Z}}(z)$ iff:

$$\lim_{\ell \to \infty} P\left\{\sup_{\alpha \in \Lambda} |R(\alpha) - R_{emp}(\alpha)| > \epsilon\right\} = 0 \quad \forall \epsilon > 0$$

- Why key ? Tells us that when considering method of ERM only the *worst case* deviations matter. No *average case* theory of consistency is possible.

Why Uniform Convergence ?

- Intuition: eventually we will play a game like this: "Take ℓ large enough so that $R_{emp}(\alpha)$ is close to $R(\alpha)$ and then choose α_ℓ that minimizes empirical risk. This should give us good actual risk as well since the two are close."
  However: if all the members of the hypothesis class do not converge uniformly then we may be tricked into selecting one with a low empirical risk that actually has a poor true risk if it is a slower converger than others.
- We need to be able to say: "No matter which hypothesis we consider, with so many training examples its empirical risk is (almost) sure to be (very) close to its true risk." Then we can proceed to compare hypotheses without fear and we can assert that our training error will be a good reflection of test error.
- This is so called PAC Learning [Valiant '84] ("probably approximately correct") except for computational considerations.

Uniform Convergence Conditions

- If our hypothesis class has only a single element or even any finite number of elements in it then we can find conditions on when uniform convergence will occur and also bounds on the rate with respect to the number of samples in the training set based on the Law of Large Numbers from classical probability.
- But most often our hypothesis class is infinite. When can infinite classes be learned from a finite number of samples ?
- Vapnik and Chervonenkis answered this question for binary valued function classes in 1968 and for real valued functions in 1981.
- They demonstrated necessary and sufficient conditions in the form of an input-distribution dependent quantities known as the *VC Entropy* and *VC Annealed Entropy* of function classes.
- However, their most exciting results were their input-distribution *independent* sufficient conditions based on a quantity called the *Growth Function* of a function class.

Growth Function

- Consider a class of binary valued functions $Q_\alpha(z)$ parameterized by $\alpha \in \Lambda$ and a finite set of data samples $z_1, \ldots, z_\ell$.
- Construct the binary ℓ-vector $q_\alpha = [Q_\alpha(z_1), \ldots, Q_\alpha(z_\ell)]$.
- As the parameter α is moved throughout Λ, q_α changes, visiting various corners of the ℓ-dimensional hypercube. Count the number of corners it visited, call this number $N_\Lambda(z_1, \ldots, z_\ell)$.
- N_Λ is the number of *dichotomies induced* by the function class $Q_\alpha(z)\ \alpha \in \Lambda$ on the dataset. Clearly, $N_\Lambda \leq 2^\ell$.
- The maximum value of N_Λ over all possible datasets of size ℓ depends only on the function class and on ℓ. Call it $m_\Lambda(\ell)$.

$$m_\Lambda(\ell) = \max_{\mathcal{Z}^\ell} N_\Lambda(z_1, \ldots, z_\ell)$$

- This is the so-called *growth function* $m_\Lambda(\ell)$ for the function class. [Different in Vapnik '95.]

VC Dimension

- Vapnik and Chervonenkis ('71) proved the surprising and very powerful result:
$$P\left\{\sup_{\alpha\in\Lambda} |R(\alpha) - R_{emp}(\alpha)| > \epsilon\right\} \leq 4m_\Lambda(2\ell)e^{-\epsilon^2\ell}$$
independent of the input distribution $P_{\mathcal{Z}}(z)$!
- It has the nice equivalent form which say that:
$$|R(\alpha) - R_{emp}(\alpha)| \leq \frac{\log[4m_\Lambda(2\ell)/\eta]}{\ell}$$
with probability at least $(1-\eta)$ *simultaneously* for all functions $Q_\alpha(z)$ $\alpha\in\Lambda$.
- It also has the useful corollary:
A sufficient condition for uniform convergence and thus for consistency of ERM is that there exists a finite d such that $m_\Lambda(\ell) \leq \ell^d + 1$.
- Finally, they noted the amazing fact [Cover '65]:
Either $m_\Lambda(\ell)$ is identically equal to 2^ℓ or it is bounded above by $m_\Lambda \leq \ell^d + 1$.
- The single scalar d above characterizes the complexity or capacity of the entire function class. It is called the *VC Dimension* of the function class.

Some Notes on VC Dimension

- If the growth function is equal to 2^ℓ then the function class has *infinite* VC dimension.
- The VC dimension depends only on the function class and is not necessarily equal to the number of free parameters in α. It may be equal to, more than, or less than this number.
- The convergence guaranteed by the corollary is *fast* (exponential). However finite VC dimension is *necessary and sufficient* for uniform convergence independent of probability distribution. (Only fast convergence exists.)
- The results can be generalized to strictly bounded and merely non-negative real valued functions.
- The growth function proceeds as 2^ℓ as long as $\ell < d$ and then begins to grow only polynomially.
- An *equivalent* definition of the VC dimension is the size of the largest dataset that can be found so that the function class *shatters* the dataset (induce all possible dichotomies).

How Many Samples are Needed to Generalize Well ?

- Say that we did not have the VC result but we had established (somehow) consistency and thus uniform convergence. The question would still remain: how fast ?
- Consider just one hypothesis α_*. It has some true risk which (for example in classification) is its prob. of error. Its empirical error can be considered the outcome of ℓ independent Bernoulli trials with prob. of success $1 - R(\alpha_*)$ The std. deviation goes as $\sqrt{\ell}$ and so the empirical risk converges to the true risk as $1/\sqrt{\ell}$. So if we had just a single hypothesis to consider we could easily find how large ℓ should be.
- But we have a huge number of hypotheses and we want them to *all* have empirical risks close to their true risks. We could still write this down for a finite size hypothesis class (ℓ would be huge) but what about infinite hypothesis classes ?
- The VC result answers this question and gives bounds for the finite cases as well.

Question

- Still missing is an answer to the question of what changes if we remove the second condition of consistency (i.e. we sacrifice the ability to estimate the risk well).
- Can we do better if we say "I don't care if my training error has nothing to do with my test error so long as you promise me that minimizing training error gets me the right hypothesis."?
- What can be said here ?

Complexity Control for Small Sample Size

- The main VC result independent of the input distribution is the bound:

$$R(\alpha_\ell) \le R_{emp}(\alpha_\ell) + \Phi\left(\frac{\ell}{d}, \frac{-\log \eta}{\ell}\right)$$

  which is true with probability at least $(1-\eta)$.

- If the sample size is large compared to the VC dimension then ℓ/d will be large and so the second term (confidence interval) will be small. If we can get a low value of empirical risk we can guarantee small actual risk.

- However is the sample size is small compared to the VC dimension (say $\ell/d < 20$) then the confidence interval will be large and the bound will be poor – our test error may not be near our training error.

- To get the best possible bound on the true risk we minimize over *both* terms in the bound simultaneously; this implies that we need to introduce the VC dimension as a *control variable*.

Structural Risk Minimization

- Based on the above insight a new inference method called *structural risk minimization* was developed [VC '74]:
  Provide the hypothesis class $\mathcal{S}$ with a *structure* by defining within it successive subclasses:
  $$S_1 \subset S_2 \subset \cdots \subset S_n \quad S_k = \{Q_\alpha(z) : \alpha \in \Lambda_k\}$$
  such that the VC dimension d_k of each subclass S_k is finite and satisfies $d_1 \leq d_2 \leq \ldots \leq d_n$ (i.e. increase complexity through structure).
- Now, moving in order through the structure, find for each S_k the α_ℓ^k which minimizes $R_{emp}(\alpha)$ and compute the bound:
  $$\hat{R}_k = R_{emp}(\alpha_\ell^k) + \Phi\left(\frac{\ell}{d_k}, \frac{-\log\eta}{\ell}\right)$$
- Choose the α_ℓ^k for which $\hat{R}_k$ is smallest.
- Examples:
  1. Polynomial fitting – go up in degree by one.
  2. Neural nets – add hidden units one at a time.
  3. Neural nets – bound sum of weights by $c_1, \ldots, c_n$.

Minimizing the Risk Bound

- The usual approach to minimizing the risk bound is to choose a fixed hypothesis class (network architecture). This fixes the VC dimension and thus the confidence interval [$\Phi()$ term] of the bound. Then minimize the empirical risk (training error) in order to minimize the bound on expected risk.
- What about early stopping? Remember, we only minimize a *bound*.
- Difficulty: if the chosen class is too complex (VC dimension too large) then the confidence interval will be quite large and the bound will be bad. If it is too simple (VCD too low) then the empirical risk will be high and the bound will again suffer.
- We must redesign the machine for each different size of training data, and we must have some clever way of picking the right complexity *a priori* to avoid the above tradeoff.
- Neural network research often follows this path using many heuristics to estimate the "right" complexity.

- Another approach would be to fix the empirical risk to be small (or zero) and then to actively minimize the confidence interval [$\Phi()$ term] of the bound. This requires a learning machine in which complexity control is built in.
- *Support Vector Machines* [Vapnik '92] are an example of one learning algorithm that implements this alternate approach. They define a "weighted" template matching classifier:

$$f_{\alpha,w}(x) = sign\left[\sum_{i=1}^{m} \alpha_i K(x, w_i) \ - b\right]$$

- The function $K(\cdot, \cdot)$ defines a "distance metric" between any two members of the input space. The vectors w_i are the templates to which we match and the α_i are mixing coefficients that control the weighting between templates.
- It will turn out that almost all the coefficients α_i will be zero. The few that are nonzero define which templates are important. These important templates are called the "support vectors".

Support Vector Machines

- Hard problems to answer:
  1. How should we choose the "distance" $K(\cdot,\cdot)$?
  2. How should we pick the number (m) and locations (w_i) of the templates?
  3. How should we select mixing coefficients α_i?
  4. Even if we can answer (1),(2),(3) we still have a hard nonlinear optimization to do.
- These questions can be answered by viewing the classification rule that support vector machines implement as follows: First transform the data points to a high dimensional "straightening" feature space with a nonlinear mapping. Separate them there with a hyperplane.
- Mathematically, this comes from the observation that (almost) any distance metric $K(\cdot,\cdot)$ we choose corresponds to a dot product in some (possibly infinite) higher dimensional space.
- So now we can decide what the "optimal" thing to do is according to SRM by considering the hyperplane case.

Optimal Hyperplanes

- The other important observation is that the VC dimension of the set of hyperplanes with bounded weight vectors $\|w\|^2 \leq A$ is the minimum of $n+1$ and C^2A^2+1 for some constant C (related to the bounding sphere of the training data).
- So the "optimal" hyperplane is the one which separates the data and has the smallest VC dimension; namely the smallest weight vector.
- By selecting only amongst hyperplanes which separate the data (fix $R_{emp}(\alpha)=0$) and minimizing the norm of the hyperplane weight vector (minimize VC dimension) we can implement the structural risk inference.
- When these restrictions are translated back into the original space of data points, two surprising conditions appear:
  1. Templates must lie on training set points.
  2. The mixing coefficients are given by the solution of a quadratic optimization.