Learning Theory and Generalization:

Neural Networks and other Supervised Learning Techniques

Peter Bartlett

Australian National University

Summary

- The supervised learning problem
- Pattern recognition
  - VC theory
  - Implications for neural networks, decision trees
- Real prediction
  - Scale-sensitive dimensions
  - Implications for neural networks
- Large margin classification
- Conclusions

Supervised Learning

Pattern x, label y.
Want to predict y from x, learn the relationship from examples ('training data'):
$(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$.

Labels might be

- in $\{-1, 1\}$ (two-class classification).
Example: predict whether a customer buys a certain product.
- in $\{1, 2, \ldots, k\}$ (classification).
Example: decide if a web page falls in the appropriate category.
- in $\mathbb{R}$ (regression).
Example: predict the risk of connecting a mobile phone call.

Typically don't have a good model of the underlying relationship.

Supervised Learning

- Choose a class of functions ($F \subseteq Y^X$)
- Gather data ($(x_1, y_1), \ldots, (x_m, y_m)$)
- Choose a function from F.

Three key issues in supervised learning:

1. Approximation
2. Estimation
3. Computation

Three key issues in supervised learning

Approximation
How well can the best function model the relationship?

$$\min_{f \in F} \text{error}(f)$$

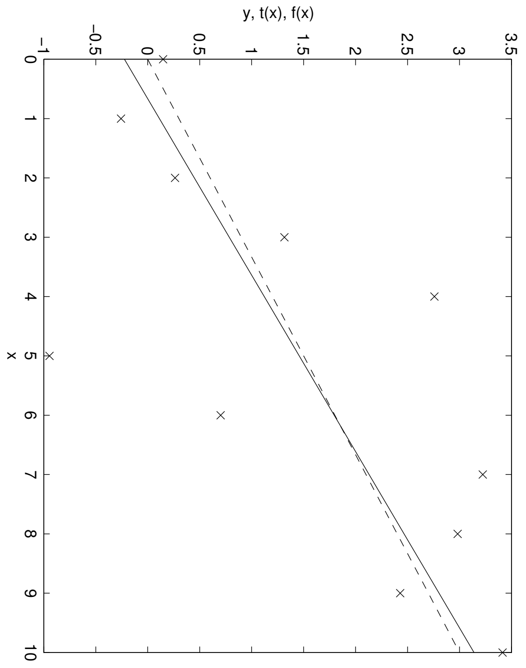
eg: $\text{error}(f) = E(f(x) - y)^2$, $(x, y) \sim P$.

Estimation
How much additional error can we expect? (Arises because learning algorithm sees only a finite amount of data.)

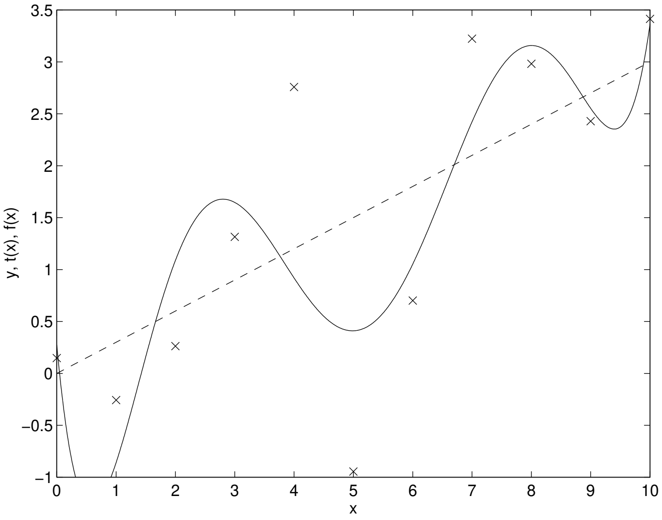
$$\text{error}\left(A(x_1, y_1, \ldots, x_m, y_m)\right) - \min_{f \in F} \text{error}(f)$$

Computation
How can we efficiently choose a suitable function?

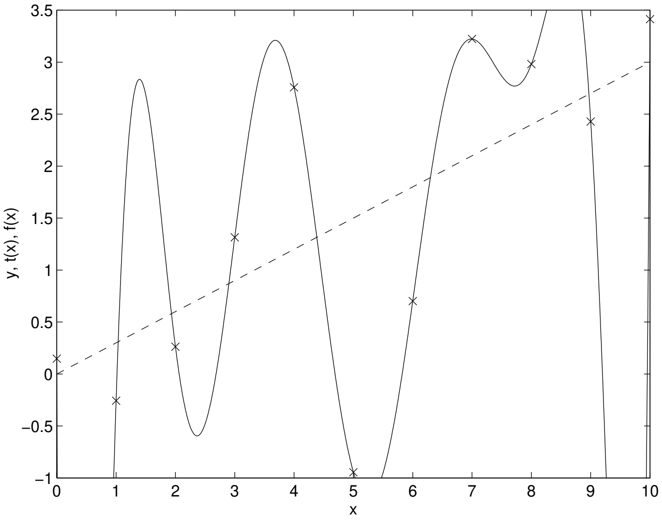
A Simple Estimate (linear)



Overfitting with a Complex Estimate (degree 6 polynomial)



Overfitting with a Complex Estimate (degree 10 polynomial)



Estimation

What determines accuracy in classification and regression?

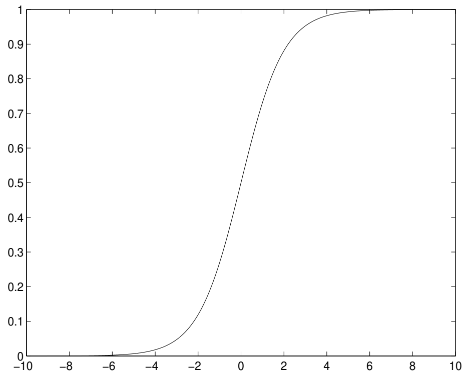
- Neural networks
- Decision trees
- Voting methods (boosting)
- Support vector machines
- ...

Two-layer Sigmoid Networks

$$f(x) = \sum_{i=1}^{k} w_i \sigma(v_i \cdot x + v_{i0}) + w_0.$$

$x \in \mathbb{R}^n$ input
$w_i \in \mathbb{R}$ output weights
$v_i \in \mathbb{R}^n, v_{i0} \in \mathbb{R}$ hidden unit weights
$\sigma : \mathbb{R} \to \mathbb{R}$ "sigmoidal" (squashing) function, such as

$$\sigma(\alpha) = \frac{1}{1 + e^{-\alpha}}.$$

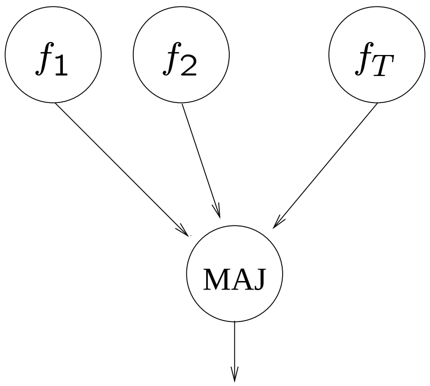


(Linear combination of squashed linear functions)

Weighted Majority of Decision Trees

$$f(x) = \text{sign}\left(\sum_{j=1}^{k} w_j f_j(x)\right)$$

x input
$w_j \in \mathbb{R}$ coefficients
$f_j : X \to \mathbb{R}$ decision tree.



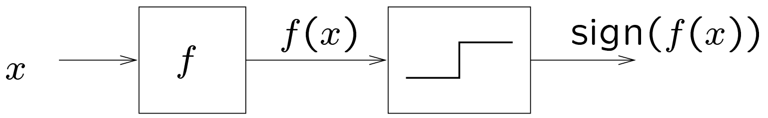
Summary

- The supervised learning problem
- Pattern recognition
  - VC theory
  - Implications for neural networks, decision trees
- Real prediction
  - Scale-sensitive dimensions
  - Implications for neural networks
- Large margin classification
- Conclusions

Pattern Recognition: VC Theory

- Learning system uses a class H of $\{-1,1\}$-valued functions.

  eg: 2-layer sigmoid networks with 12 units, thresholded output:



- Labelled examples generated independently by a probability distribution P on $X \times \{-1,1\}$.

  (models patterns and expert's labels, for training and testing)

- Want learning algorithm to choose h in H with error, $\mathrm{er}_P(h) = P(h(x) \neq y)$ small. (Ideally, close to $\mathrm{opt}_P(H) = \inf_{h \in H} \mathrm{er}_P(h)$.)

A general-purpose strategy: minimizing sample error

For a sample $z = ((x_1, y_1), \ldots, (x_m, y_m))$, define the sample error

$$\widehat{\mathrm{er}}_z(h) = \frac{1}{m} \left| \{i : h(x_i) \neq y_i\} \right|.$$

Let $L(z) = h$ with

$$\widehat{\mathrm{er}}_z(h) = \min_{g \in H} \widehat{\mathrm{er}}_z(g).$$

(So L chooses h to minimize disagreements with the training data.)

When is this an effective learning algorithm?

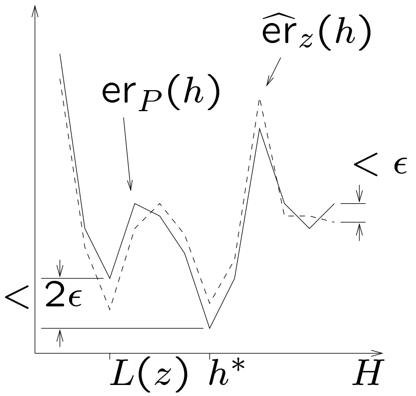
A general-purpose strategy: minimizing sample error

If $\widehat{\text{er}}_z(h)$ and $\text{er}_P(h)$ are close for all $h \in H$:

$$\sup_{h\in H} |\widehat{\text{er}}_z(h) - \text{er}_P(h)| \leq \epsilon,$$

then

$$\begin{aligned} \text{er}_P(L(z)) &\leq \inf_{h\in H} \text{er}_P(h) + 2\epsilon \\ &= \text{opt}_P(H) + 2\epsilon. \end{aligned}$$



Convergence of $\widehat{\text{er}}_z(h)$ to $\text{er}_P(h)$

Consider a **single function** $h : X \to \{-1, 1\}$.

$$\text{er}_P(h) = P(h(x) \neq y) = p.$$

$$\widehat{\text{er}}_z(h) = \frac{1}{m} |\{i : h(x_i) \neq y_i\}|$$

Want $\widehat{\text{er}}_z(h) \to \text{er}_P(h)$ as $m \to \infty$. (Law of large numbers.)

Binomial distribution:

$\Pr(\widehat{\text{er}}_z(h) = k/m)$ = $\Pr(k$ heads in m tosses of a biassed coin; $\Pr(\text{head}) = p)$.

Convergence of $\widehat{\text{er}}_z(h)$ to $\text{er}_P(h)$

Standard (Chernoff) bounds on tails of the binomial distribution show that for all h

$$\Pr\left(|\text{er}_P(h) - \widehat{\text{er}}_z(h)| > \epsilon\right) \leq 2\exp(-2\epsilon^2 m).$$

We want this uniformly over $h \in H$.
Suppose H is **finite**:

$$\begin{aligned} & \Pr\left(\text{some } h \in H \text{ has } |\text{er}_P(h) - \widehat{\text{er}}_z(h)| > \epsilon\right) \\ \leq & \sum_{h \in H} \Pr\left(|\text{er}_P(h) - \widehat{\text{er}}_z(h)| > \epsilon\right) \\ \leq & |H| \cdot 2\exp(-2\epsilon^2 m). \end{aligned}$$

(**Uniform convergence** result.)

Uniform convergence and learning

So if $L(z)$ minimizes sample error, with probability $1 - \delta$ it has

$$\text{er}_P(L(z)) \leq \text{opt}_P(H) + \left(\frac{2}{m}\ln(2|H|/\delta)\right)^{1/2}.$$

Uniform convergence and learning

Example $H =$ decision trees on $X = \{-1, 1\}^n$ with N decision nodes.

We can specify the tree by listing, in breadth-first order, either the variable on which the node splits or the label of the leaf. This list has $2N+1$ items, each from a set of size $n+2$, so

$$|H| \leq (n+2)^{2N+1}.$$

So if L minimizes sample error,

$$\mathrm{er}_P(L(z)) \leq \mathrm{opt}_P(H) + \left(\frac{2}{m}\left((2N+1)\ln(n+2) + \ln\left(\frac{2}{\delta}\right)\right)\right)^{1/2}.$$

Uniform convergence and learning

Notes:

- Minimizing sample error seems very simple-minded. We'll see it's nearly optimal (in terms of rate of convergence).

- Finiteness is a significant restriction. (What about functions specified by real parameters?)

Vapnik-Chervonenkis dimension

What if H is infinite?

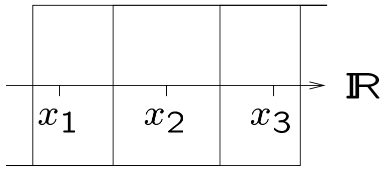
Suppose
H is a class of functions from X to $\{-1, 1\}$,
and $x = (x_1, \ldots, x_k) \in X^k$.
Then we define $H_{|x}$ as

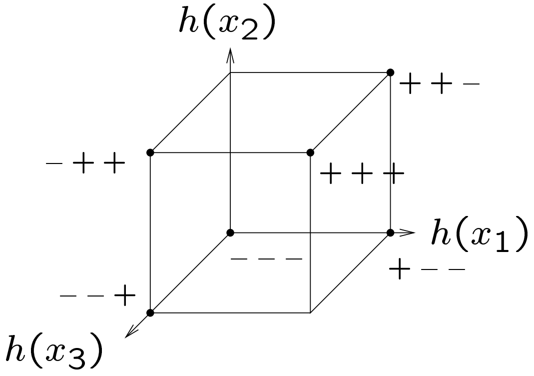
$$H_{|x} = \{(h(x_1), \ldots, h(x_k)) : h \in H\} \subseteq \{-1, 1\}^k.$$

Crucial property: $\left|H_{|x}\right|$ versus 2^k.

Example:
H = threshold function on $\mathbb{R}$,
$x = (x_1, x_2, x_3)$:

x_1	x_2	x_3
$-$	$-$	$-$
$+$	$-$	$-$
$+$	$+$	$-$
$+$	$+$	$+$
$-$	$+$	$+$
$-$	$-$	$+$





Vapnik-Chervonenkis dimension

If $\left|H_{|x}\right| = 2^k$, H computes **all** dichotomies of $\{x_1, \ldots, x_k\}$.
In that case, we say that H **shatters** x.

The Vapnik-Chervonenkis dimension (VC-dimension) of H is

$$\text{VCdim}(H) = \max\left\{k : \text{ some } x \in X^k \text{ is shattered by } H\right\}.$$

Example:
H =threshold function on $\mathbb{R}$,
$\text{VCdim}(H) = 2$.

Uniform Convergence and VC Classes

Key result ("Sauer's Lemma"):
If $\text{VCdim}(H) \leq d$ then

$$\left|H_{|x}\right| \leq \sum_{i=0}^{d} \binom{m}{i} \leq (2m)^d,$$

for $x \in X^m$.

ie: Number of distinct classifications of m points grows only polynomially with m.

Uniform Convergence and VC Classes

Theorem For a sample error minimizer L, with probability at least $1-\delta$,

$$\mathrm{er}_P(L(z)) \leq \mathrm{opt}_P(H) + \left(\frac{c}{m}\left(d + \ln\left(\frac{1}{\delta}\right)\right)\right)^{1/2},$$

for $d = \mathrm{VCdim}(H)$.

- Simple strategy (minimize errors)
- Very general result
- Many approximations, but nearly optimal:

Converse result

Theorem
For any learning algorithm L, there is a probability distribution P on $X \times \{-1,1\}$ such that with probability at least 1/200,

$$\mathrm{er}_P(L(z)) \geq \mathrm{opt}_P(H) + c\left(\frac{\mathrm{VCdim}(H)}{m}\right)^{1/2}.$$

VC Theory

The worst-case difference between $\mathrm{er}_P(L(z))$ and $\mathrm{opt}_P(H)$ grows as

$$\left(\frac{\mathrm{VCdim}(H)}{m}\right)^{1/2}$$

- VC-dimension is the crucial quantity: VCdim finite $\Leftrightarrow$ can estimate from a finite sample.
- Simple minded strategy (minimize sample error) has nearly optimal estimation rate.

VC Theory

For a function class H, the following statements are equivalent:

- A sample error minimizer gives near-optimal error.
- $\widehat{\mathrm{er}}_z$ converges to er_P uniformly over H, and the rate is uniform over all distributions P.
- H has finite VC-dimension.
- For $x \in X^m$, the number of classifications, $\left|H_{|x}\right|$ is bounded by a polynomial in m.

VC-dimension calculations: Neural networks

$\boxed{\text{VC-dimension} \approx \text{number of parameters}}$

- For linear threshold functions on $\mathbb{R}^n$,

  $$\text{VCdim} = n + 1,$$

  (number of parameters is $n+1$)

- For linear threshold networks, and fixed depth networks with piecewise polynomial squashing functions,

  $$c_1 W \leq \text{VCdim} \leq c_2 W \log W,$$

  where W is number of weights in network.

- Some threshold networks have VCdim$\geq cW \log W$.

- VCdim(sigmoid net) $\leq cW^4$.

VC-dimension calculations

General results:

Any function class H that can be computed by a program that takes a real input vector x and k real parameters and involves no more than t of the following operations:
- $+, -, \times, /$ on real numbers
- $>, \geq, <, \leq, =, \neq$ on real numbers
- output -1 or 1

has VC-dimension $O(kt)$.

Any function class H that can be computed by a program that takes a real input vector x and k real parameters and involves no more than t of the following operations:
- $+, -, \times, /, e^{\alpha}$ on real numbers
- $>, \geq, <, \leq, =, \neq$ on real numbers
- output -1 or 1

has VC-dimension $O(k^2 t^2)$.

(eg: sigmoid nets, RBF nets, mixture of experts)

VC-dimension calculations

For neural networks and decision trees:

$$\boxed{\text{VC-dimension} \approx \text{size.}}$$

Hence, **misclassification probability** is no more than

$$\text{training error} + \left(\frac{\text{size}}{m}\right)^{1/2}.$$

This suggests that the amount of training data should grow roughly linearly with size. And for any algorithm, there is a classification problem (probability distribution) for which the amount of training data *must* grow linearly with size, or the algorithm cannot return a near-optimal classifier.

Model Selection

Structural Risk Minimization:

> Choose complexity parameter d
> (eg: number of hidden units in neural network, size of decision tree)
> and function $h \in H_d$ to minimize
>
> $$\widehat{\text{er}}_z(h) + c\left(\frac{\text{VCdim}(H_d)}{m}\right)^{1/2}.$$

(Minimize upper bound on error.)

Summary

- The supervised learning problem
- Pattern recognition
  - VC theory
  - Implications for neural networks, decision trees
- Real prediction
  - Scale-sensitive dimensions
  - Implications for neural networks
- Large margin classification
- Conclusions

Supervised learning for real-valued functions

- Learning system uses a class F of real-valued functions. (Assume $F \subset [0,1]^X$.)
- Labelled examples generated independently by a probability distribution P on $X \times Y$. Special cases:
  - Function, $f : X \to Y$, $y_i = f(x_i)$.
  - Noisy functions, $f : X \to Y$, $y_i = f(x_i) + \eta_i$ (η_i iid)
  - Noise depends on x_i
- Want learning algorithm to choose f in F with error, $\mathrm{er}_P(f) = \mathbf{E}(f(x) - y)^2$ small. (Ideally, close to $\mathrm{opt}_P(F) = \inf_{f \in F} \mathrm{er}_P(f)$.)

Minimizing sample error

Define sample error,

$$\widehat{\text{er}}_z(f) = \frac{1}{m}\sum_{i=1}^{m}\left(f(x_i) - y_i\right)^2.$$

Choose $L(z) = f$ with

$$\widehat{\text{er}}_z(f) = \min_{g \in F} \widehat{\text{er}}_z(g).$$

Uniform convergence

If

$$\max_{f \in F} |\widehat{\text{er}}_z(f) - \text{er}_P(f)| \leq \epsilon$$

then

$$\text{er}_P(L(z)) \leq \text{opt}_P(F) + 2\epsilon.$$

Consider a single function $f : X \to [0,1]$.

$$\begin{aligned}\text{er}_P(f) &= \int_{X \times Y} (f(x) - y)^2 \, dP(x,y) \\ \widehat{\text{er}}_z(f) &= \frac{1}{m}\sum_{i=1}^{m}(f(x_i) - y_i)^2\end{aligned}$$

Want $\widehat{\text{er}}_z(f) \to \text{er}_P(f)$ as $m \to \infty$.
(Law of large numbers)

Hoeffding's inequality gives the rate.

Uniform convergence

For any $f : X \to [0,1]$, we have

$$\Pr\left(|\mathrm{er}_P(f) - \widehat{\mathrm{er}}_z(f)| > \epsilon\right) \le 2\exp(-2\epsilon^2 m)$$

For a finite class, this implies

$$\Pr\left(\sup_{f\in F} |\mathrm{er}_P(f) - \widehat{\mathrm{er}}_z(f)| > \epsilon\right) \le 2|F|\exp(-2\epsilon^2 m),$$

so, with probability $1-\delta$, if L minimizes sample error it has

$$\mathrm{er}_P(L(z)) \le \mathrm{opt}_P(F) + \left(\frac{2}{m}\ln\left(\frac{2|F|}{\delta}\right)\right)^{1/2}.$$

Uniform convergence

Example
$F =$ neural network with one output unit, W parameters, each represented with k bits.
Then $|F| \le 2^{kW}$, so

$$\mathrm{er}_P(L(z)) \le \mathrm{opt}_P(F) + \left(\frac{2}{m}\left(kW\ln 2 + \ln(2/\delta)\right)\right)^{1/2}.$$

(Also: decision tree of finite size, splits on finite number of values, finite number of labels.)

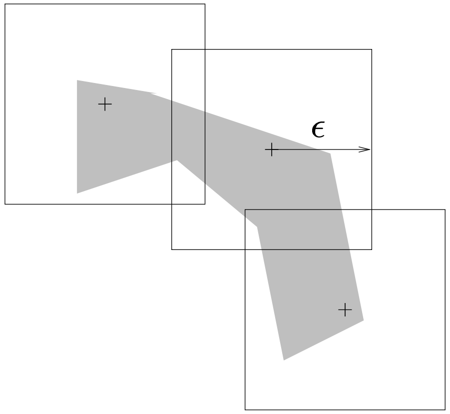
Uniform convergence over infinite classes: Covering numbers

For $f : X \to [0,1]$,
$f_{|x} = (f(x_1), \ldots, f(x_m)) \in [0,1]^m$.
Typically, $\left|F_{|x}\right| = \infty$.

But we only need to *approximate* F...

Covering number = size of an approximating set:

$$\mathcal{N}\left(F_{|x}, \epsilon\right) = \min\left\{|\hat{F}| : \text{ for all } f \in F \text{ some } \hat{f} \in \hat{F} \text{ has } \left\|f_{|x} - \hat{f}_{|x}\right\| < \epsilon\right\}.$$



Uniform convergence over infinite classes

$$P^m\left\{\exists f \in F : |\mathrm{er}_P(f) - \widehat{\mathrm{er}}_z(f)| \geq \epsilon\right\} \leq 4 \max_{x \in X^{2m}} \mathcal{N}\left(F_{|x}, \epsilon/16\right) e^{-\epsilon^2 m/32}.$$

If $\boxed{\max_{x \in X^{2m}} \mathcal{N}\left(F_{|x}, \epsilon/16\right)}$ grows too quickly with m, the probability does not converge to zero.

(How much does $F_{|x}$ ‘fill up’ the cube $[0,1]^m$?)

Covering numbers

Example:
$F =$ analytic functions from $\mathbb{R}$ to $[0,1]$.
For any $x \in X^m$, $F_{|x} = [0,1]^m$. So $\mathcal{N} \sim \left(\frac{1}{\epsilon}\right)^m$.

Example: $F \subseteq$ finite-dimensional linear space.
For any $x \in X^m$, $F_{|x}$ is a subset of a d-dimensional subspace of $\mathbb{R}^m$.

$$\mathcal{N} \leq \left(\frac{m}{\epsilon}\right)^d$$

and hence

$$\mathrm{er}_P(L(z)) \leq \mathrm{opt}_P(F) + O\left(\left(\frac{1}{m}\left(d \log^2 m + \log \frac{1}{\delta}\right)\right)^{1/2}\right).$$

Example: $F =$ non-decreasing functions from $\mathbb{R}$ to $[0,1]$.

$$\mathcal{N} \leq m^{1/\epsilon}$$

Hence,

$$\mathrm{er}_P(L(z)) \leq \mathrm{opt}_P(F) + O\left(\left(\frac{1}{m}\log(m/\delta)\right)^{1/3}\right).$$

Covering numbers and scale-sensitive dimensions

Theorem

$$c_1 \mathrm{fat}_F(16\epsilon) \leq \log \mathcal{N} \leq c_2 \mathrm{fat}_F\left(\frac{\epsilon}{4}\right) \log^2\left(\frac{m}{\epsilon}\right).$$

Here, fat_F is a *scale-sensitive dimension*, the *fat-shattering dimension*.

(cf: Sauer's Lemma)

Uniform convergence over infinite classes

We only need to measure the complexity of F at an appropriate scale:

With probability $1-\delta$, if L minimizes sample error it has

$$\mathrm{er}_P(L(z)) \leq \mathrm{opt}_P(F) + \epsilon$$

for

$$\epsilon \geq \left(\frac{c_1}{m}\left(\mathrm{fat}_F(c_2\epsilon) + \log\left(\frac{1}{\delta}\right)\right)\right)^{1/2}.$$

(+ converse result)

Covering numbers and scale-sensitive dimensions

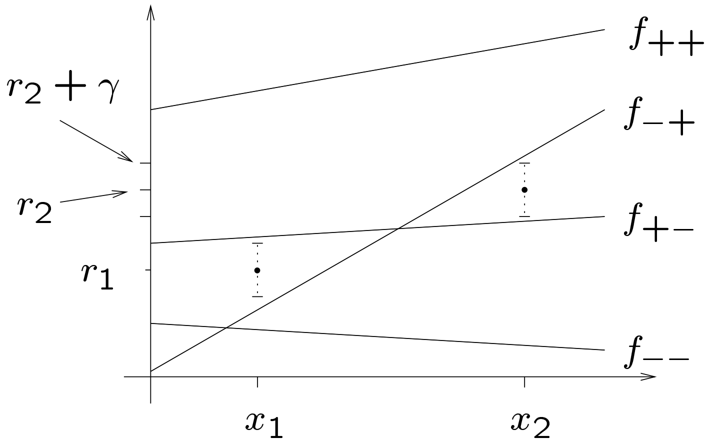
Recall: For $H \subseteq \{-1,1\}^X$, H shatters x if for all $b = (b_1, \ldots, b_m) \in \{-1,1\}^k$, there is a function $h_b \in H$ with

$$\begin{aligned} h_b(x_i) &= 1 \quad \text{if } b_i = 1 \\ h_b(x_i) &= -1 \quad \text{if } b_i = -1. \end{aligned}$$

Definition (γ-shattering)
For $F \subseteq \mathbb{R}^X$, $x \in X^k$, and $\gamma > 0$
x is γ-**shattered by** F if
there are real numbers $r_1, \ldots, r_k$ such that
for all $b = (b_1, \ldots, b_k) \in \{-1,1\}^k$,
there is a function $f_b \in F$ with

$$\begin{aligned} f_b(x_i) &\geq r_i + \gamma \quad \text{if } b_i = 1 \\ f_b(x_i) &\leq r_i - \gamma \quad \text{if } b_i = -1. \end{aligned}$$

Scale-sensitive dimensions



Covering numbers and scale-sensitive dimensions

$\text{VCdim}(H) = \max\{m : H \text{ shatters some } x \in X^m\}$,

The **fat-shattering dimension** of a class $F \subseteq \mathbb{R}^X$ is

$\text{fat}_F(\gamma) = \max\Big\{k : \text{ some } x \in X^k \text{ is } \gamma\text{-shattered by } F\Big\}$.

In learning real-valued functions, it is only complexity at the appropriate **scale** that matters.

Properties of $\mathrm{fat}_F(\gamma)$

- Think of $\mathrm{fat}_F(\gamma)$ as VCdim at scale γ.

- $\mathrm{fat}_F(\gamma)$ is a non-increasing function of γ. (F is more complex when examined on a smaller scale.)

- For many natural function classes,

$$\lim_{\gamma \to 0} \mathrm{fat}_F(\gamma) \approx \mathrm{VCdim}(\mathrm{sign}(F)).$$

  Hence, VCdim results also apply when learning real functions.

- Some function classes F have

$$\begin{aligned} \mathrm{VCdim}(\mathrm{sign}(F)) &= \infty \\ \mathrm{fat}_F(\gamma) &< \infty \qquad \forall \gamma > 0. \end{aligned}$$

  (eg: arbitrarily large two-layer networks with small output weights.)

Fat-shattering dimension calculations

Using VC-dimension results for neural networks, decision trees, etc,

$$\text{expected error} \leq \text{training error} + \left(\frac{\text{size}}{m}\right)^{1/2}.$$

But this can be improved significantly in some cases. (Eg: large networks, small parameters, little data.)

Sigmoid Networks with Small Weights

$$f(x) = \sum_{j=1}^{k} w_j \sigma(v_j \cdot x + v_{j0}) + w_0,$$

with $\sum_{i=1}^{k} |w_i| \leq V$, $x \in \mathbb{R}^d$.
For these networks (with k arbitrary),

$$\mathrm{fat}_F(\epsilon) \leq c \frac{V^2 d}{\epsilon^2} \log\left(\frac{Vd}{\epsilon}\right).$$

> With high probability,
>
> $$\mathrm{er}_P(L(z)) \leq \mathrm{opt}_P(F) + \left(\frac{cV^2 d}{m}\right)^{1/3}.$$

In the case of large networks, the amount of training data should grow with the size of the parameters.

Model Selection

Structural Risk Minimization
(minimize upper bound on error):

> Choose complexity parameter V
> (upper bound on size of output parameters in network)
> and function $f \in H_V$ to minimize
>
> $$\widehat{\mathrm{er}}_z(f) + c\left(\frac{V^2}{m}\right)^{1/3}.$$

- Penalty term involving size of parameters.

Summary

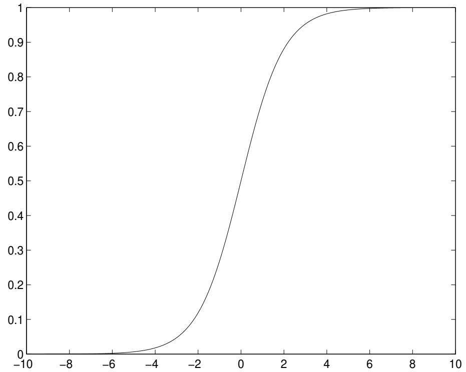
- The supervised learning problem
- Pattern recognition
  - VC theory
  - Implications for neural networks, decision trees
- Real prediction
  - Scale-sensitive dimensions
  - Implications for neural networks
- Large margin classification
- Conclusions

A Surprising Consequence of the VC Theory

Two-layer sigmoid network: $(x \in \mathbb{R}^n)$

$$f(x) = \sum_{j=1}^{k} w_j \sigma(v_j \cdot x + v_{j0}) + w_0.$$

$\sigma(\alpha)$



α

If v_j, w_j "small", network is approximately linear

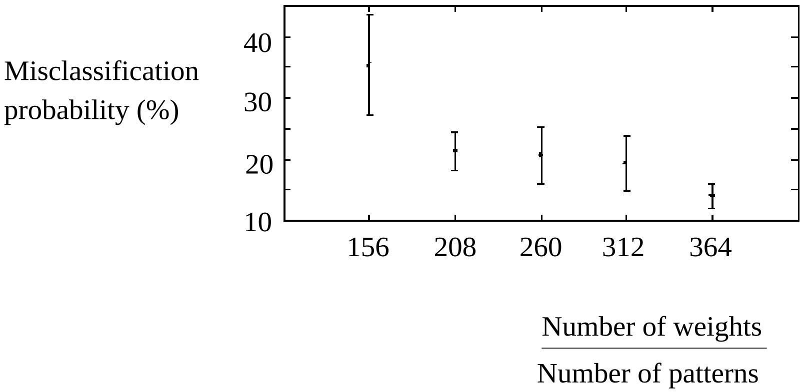
Linear:	VCdim $= n + 1$
Arbitrarily small weights:	VCdim $\geq$ number of parameters ($\approx nk$).

Experimental Results

Neural networks with many parameters, trained on small data sets, sometimes generalize well.

Eg: Face recognition (Lawrence *et al*, 1996)

$m = 50$ training patterns.



Voting Methods

$$f(x) = \text{sign}\left(\sum_{j=1}^{T} \alpha_j f_j(x)\right).$$

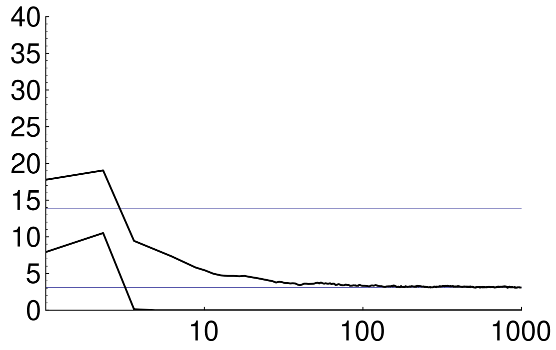
Freund and Schapire's `AdaBoost` algorithm:

- Iteratively adds classifiers f_j.
- For each j,
  - (multiplicatively) reweights the training data to emphasize mistakes,
  - uses a learning algorithm to choose f_j.
  - Chooses the voting weight α_j according to the accuracy of f_j.

Very successful for real-world classification problems. (eg: UCI repository)

Empirical Results

Boosting C4.5 for the (UCI) "*letter*" dataset. Training and test errors, versus number of boosting rounds:



- Boosting improves performance (decreases test error from 13.8% to 3.1%)

- Increasing size of combined function does not harm generalization!

(Observed by a number of researchers: Drucker and Cortes, Quinlan, Breiman)

VC-theory.

- $\{-1, 1\}$-valued functions.
- Learning algorithm minimizes number of disagreements on training set.

Neural network practice.

- Real-valued functions.
- Learning algorithm minimizes squared error on training set. (pushes network outputs towards -1 and 1)

Margins

Training data $(x_1, y_1), \ldots, (x_m, y_m)$
(patterns x_i, labels $y_i \in \{-1, 1\}$).

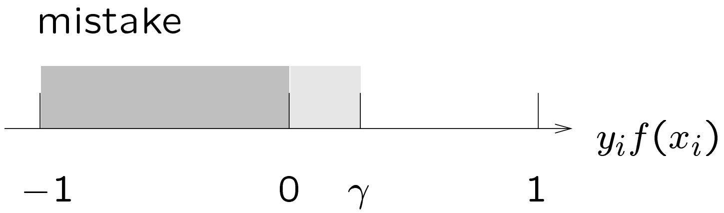
Instead of

$$\frac{1}{m}\left|\{i : \mathrm{sign}(f(x_i)) \neq y_i\}\right|,$$

consider

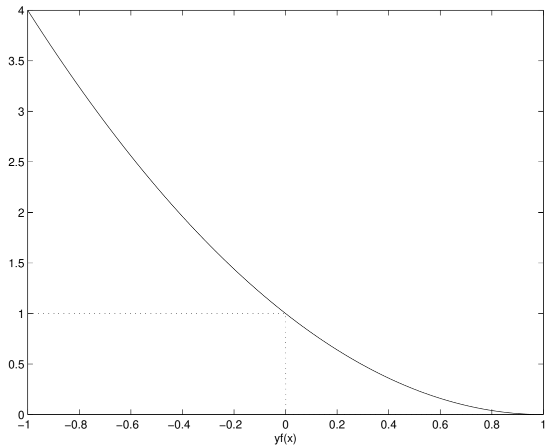
$$\frac{1}{m}\left|\{i : y_i f(x_i) < \gamma\}\right|.$$

Margin of (x_i, y_i) is $y_i f(x_i)$.



Neural Network Margin Cost Function

Quadratic



Generalization Behaviour and Margins

Theorem With high probability over m i.i.d. training examples, every function $f : X \to [-1, 1]$ in a class F has misclassification probability bounded by

$$\Pr(\mathrm{sign}(f(x)) \neq y) \leq \underbrace{\frac{1}{m}|\{i : y_i f(x_i) < \gamma\}|}_{\text{training error}} + \left(\frac{\mathrm{fat}_F(\gamma)}{m}\right)^{1/2}.$$

Since $\mathrm{fat}_F(\gamma)$ is a non-increasing function of γ, and $\lim_{\gamma\to 0} \mathrm{fat}_F(\gamma) \approx \mathrm{VCdim}(\mathrm{sign}(F))$, the result "improves" on the VC bounds.

Sigmoid Networks with Small Weights

$$f(x) = \sum_{j=1}^{k} w_j \sigma(v_j \cdot x + v_{j0}) + w_0.$$

Theorem With high probability over m i.i.d. training examples, every sigmoid network has misclassification probability bounded by

$$\Pr(\mathrm{sign}(f(x)) \neq y) \leq \frac{1}{m}|\{i : y_i f(x_i) < \gamma\}| + \frac{\|w\|_1}{\gamma}\left(\frac{n}{m}\right)^{1/2}$$

$$\text{where } \|w\|_1 = \sum_{j=1}^{k} |w_j|.$$

If w_j "small": network is approximately linear.

Sigmoid Networks

- Complexity of network

$$\approx \frac{\textbf{size of parameters}}{\textbf{margin}}$$

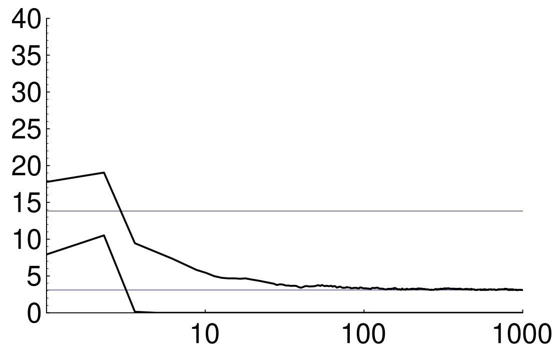
- Supports the use of certain popular heuristics (weight decay, early stopping).

- Consistent with experimental evidence.

- Suggests new algorithms:
  explicitly increase the margin, while controlling the size of parameters.

Recall: Empirical Results

Boosting C4.5 for the (UCI) "*letter*" dataset. Training and test errors, versus number of boosting rounds:



- Increasing size of combined function does not harm generalization!

Another Formulation of Boosting

(Breiman, Frean and Downs, Friedman *et al*)

At boosting round t, `Adaboost` chooses f_t and $w_t > 0$ to minimize

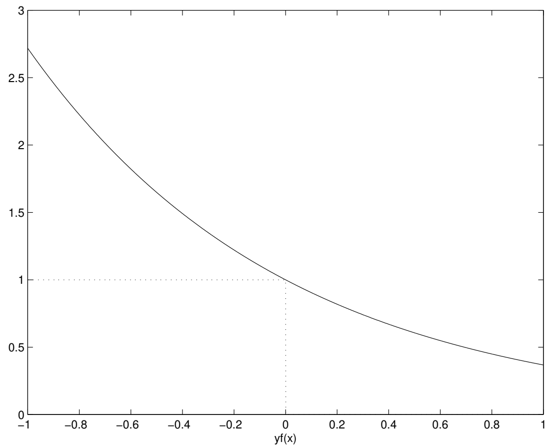
$$\frac{1}{m}\sum_{i=1}^{m} \exp\left(-\frac{1}{2}y_i\left(\sum_{j=1}^{t-1} w_j f_j(x_i) + w_t f_t(x_i)\right)\right).$$

Simple manipulations give

- a classification problem to choose the base classifier f_t,
  ($\Rightarrow$ multiplicative reweighting of training data)
- a closed-form expression for the corresponding weight w_t.

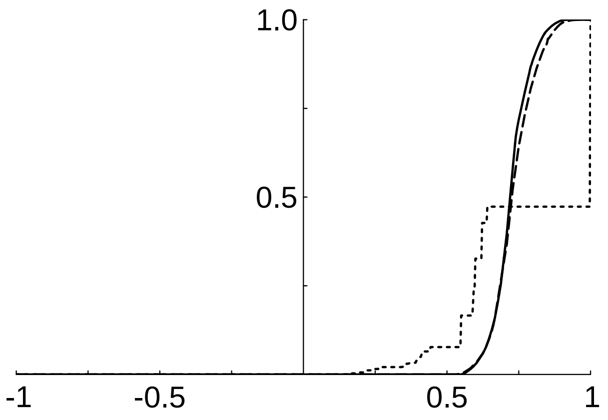
`Adaboost` Margin Cost Function

Exponential



AdaBoost increases the margin

Margin distribution
(= proportion of training data with margin $< \alpha$)
for *letter* dataset after 5, 100, and 1000 rounds of boosting:



Boosting and Margins:

AdaBoost increases the margin

Theorem If each f_j (base classifier produced in round j) predicts slightly better (say, by γ) than random guessing, then a sufficiently large combined classifier satisfies

$$y_i f(x_i) = y_i \sum_{j=1}^{T} \alpha_j f_j(x_i) \geq \gamma$$

for every training example (x_i, y_i).

(Recall:
patterns x_i, labels $y_i \in \{-1, 1\}$, $f : X \to [-1, 1]$)

Boosting and Margins:

Large margin = low complexity

Theorem With high probability over m i.i.d. training examples, every convex combination of classifiers from a class F has misclassification probability bounded by

$$\Pr(\mathrm{sign}(f(x)) \neq y) \leq \frac{1}{m}|\{i : y_i f(x_i) < \gamma\}| + \left(\frac{\mathrm{VCdim}(F)}{\gamma^2 m}\right)^{1/2}.$$

Support Vector Machines

$$F = \{x \mapsto w \cdot x : \|w\| \leq 1, \|x\| \leq B\}.$$

$$\mathrm{fat}_F(\gamma) \leq \frac{B^2}{\gamma^2}.$$

Theorem Every f in F with

$$y_i f(x_i) \geq \gamma$$

for $i = 1, \ldots, m$ has

$$\Pr(\mathrm{sign}(f(x)) \neq y) \leq \frac{cB^2}{\gamma^2 m}.$$

Conclusions

$$\text{error} \leq \text{error estimate} + \frac{\text{complexity}}{\text{sample size}}$$

Classification

error estimate = proportion misclassified:

$$\Rightarrow \qquad \text{complexity} \approx \text{number of parameters}$$

margin-based error estimate:

$$\Rightarrow \qquad \text{complexity} \approx \frac{\text{size of parameters}}{\text{margin}}$$

Regression

error estimate = average squared error:

$$\Rightarrow \qquad \text{complexity} \approx \text{size, number of parameters}$$

Structural risk minimization:
minimize upper bound on error
(complexity-based regularization).