# "Machine Learning Research: Four Current Directions"

Dietterich, Thomas G., (1997). "Machine Learning Research: Four Current Directions", *AI Magazine*. 18(4), 97–136.
`ftp://ftp.cs.orst.edu/pub/tgd/papers/aimag-survey.ps.gz`

- Lots of activity in Machine Learning (ML)...

- Interactions between
  - ⋆ symbolic machine learning
  - ⋆ computational learning theory
  - ⋆ neural networks
  - ⋆ statistics
  - ⋆ pattern recognition

- New applications for ML techniques
  - ⋆ knowledge discovery in databases
  - ⋆ language processing
  - ⋆ robot control
  - ⋆ combinatorial optimization

  (+ traditional problems:
  speech recognition, face recognition,
  handwriting recognition, medical data analysis,
  game playing, ...)

# Hot Topics

1. Improving accuracy by learning ensembles of classifiers

    - Subsample Training Samples (**Cross-Validated Committees; Bagging; Boosting**)
    - Manipulate Input Features
    - Manipulate Output Targets (**ErrorCorrectingOutputCode**)
    - Inject Randomness (NN: initial weights, noisy inputs; DT: splitting; **MCMC (Model Averaging)**)
    - Algorithm Specific methods (**Diversity (NN)**; "OptionTrees" (DT))

    \+ How to combine classifiers? (Unweighted; Weighted [Var, ModelAverage]; Gating; Stacking)

    \+ Why they work? (**Sample Complexity; Computational Complexity; Expressiveness**)

# Hot Topics - 2,3,4

2. Scaling up supervised learning algorithms
   - Large Training Sets
     (**Subsampling**; DataStructures;
     **Ensemble (diff subset)**; Threshold; Ripper)
   - Many Features (select/weight features)
     Preprocess [**MutualInfo; Relief-F**]
     Wrapper, LOOCV/NN
     Integrate Weighting in Learner (VSM, **Winnow**)

3. Reinforcement learning
   - Intro Dynamic Programming
   - $TD(\lambda)$
     applic: Backgammon, Job-shop scheduling, ...
   - $Q$-learning (model free)

4. Learning complex stochastic models
   - NaiveBayes, BeliefNets
     Hierarchical Mixture of Experts
     Hidden Markov Model
     Dynamic Probabilistic Network
   - Learn parameters (known struct, complete)
   - Learn parameters (known struct, incomplete)
     **Gradient Descent; EM; Gibbs Sampling**
   - Learning Structure

# Classification Task

- Target function $f: \mathcal{X} \mapsto \mathcal{Y}$

  each $\mathbf{x}^j \in \mathcal{X} = \langle x_1, \ldots, x_n \rangle$
  where $x_i \in \Re$, or discrete

  $\mathcal{Y} = \{1, ...K\}$ for CLASSIFICATION
  ($\mathcal{Y} = \Re$ for regression)

- Data $\mathbf{x} \in \mathcal{X}$ drawn from distr'n $P$,
  labeled by $f(\mathbf{x})$ (perhaps + noise)

  Error of hypothesis $h$
  *err*( $h$ ) $= P(\, \mathbf{x}$ s.t. $f(\mathbf{x}) \neq h(\mathbf{x})\,)$

Task:

| |
|---|
| Given $S = \{\langle \mathbf{x}^j,\ f(\mathbf{x}^j)\rangle\}_{i=1}^m$ |
| find (good approx to) $f$ |
| . . . $h$ s.t. *err*( $h$ ) is small (probably) |

# Comments on Classification

- Typically:

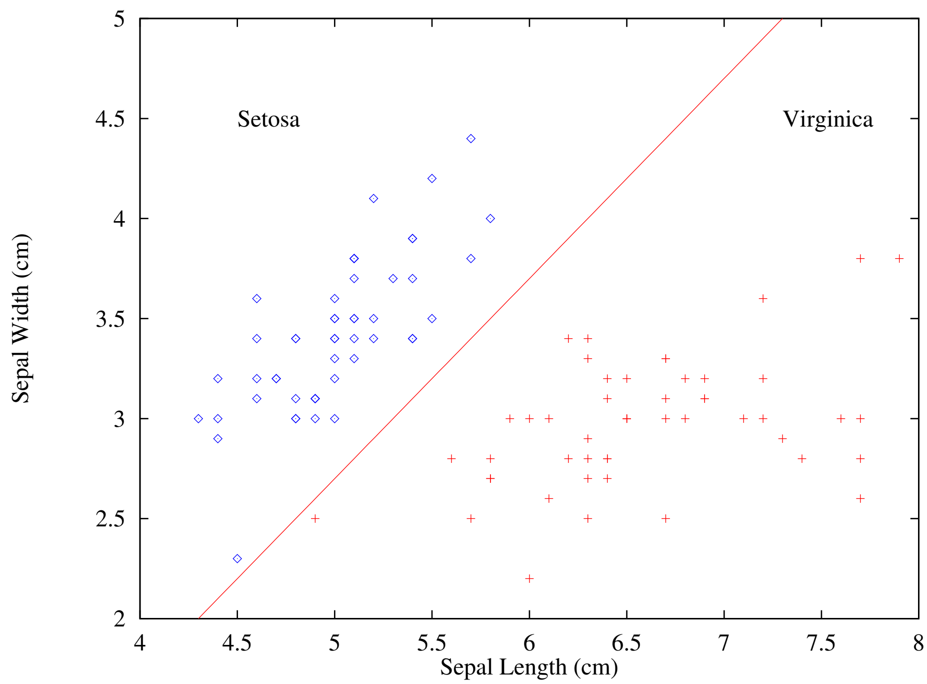
| | |
|---|---|
| **Given:** | Set of **training examples**: |
| | $\{(x_1, f(x_1)), \ldots, (x_m, f(x_m))\}$ |
| | Space of **hypotheses** $H$ |
| **Find:** | Hypothesis $h \in H$ that is good approx'n to $f$ |
| | (ie, s.t. *err*( $h$ ) small |
| | $h(x) \approx f(x)$ for most $x$ in space) |

*Note:* $f : \mathcal{X} \mapsto \mathcal{Y}$ not known
($f$ need not be in $H$)

Want $h$ that works well throughout "instance space" $\mathcal{X}$
... training examples only small subset

- Typical Hypothesis spaces:
  Decision Tree (DT) [C4.5, Cart]
  Neural Nets (NN) [Backprop, ...]
    Perceptron, MLP, RBF, ...
  Nearest Neighbor
  Belief Nets
  ... LogicPrograms, ParameterSettings, ...

# Discrete-Valued Functions: Classification




- **Unknown function:** maps from flower measurements to species of flower
- **Examples:** 100 flowers measured and classified by R.A. Fisher
- **Hypothesis Space:** All linear discriminators of form

$$h(x) = \begin{cases} Setosa & \text{if } w_0 + w_1 \cdot x.SepalWidth + w_2 \cdot x.SepalLength > 0 \\ Virginica & \text{otherwise} \end{cases}$$

## Improve Classification Accuracy by learning ensembles of classifiers

Q: Why not use $h = $ majority$\{h_1,\ h_2,\ h_3\}$ ?

$\forall x \quad h(x) = \text{majority}\{h_1(x),\ h_2(x),\ h_3(x)\}$

If $h_i$ make *INDEPENDENT mistakes*,
$h$ is more accurate!

Eg: If *err*$(\,h_i\,) = \epsilon$, then *err*$(\,h\,) = 3\epsilon^2$
(0.01 $\mapsto$ 0.0003)
If majority of $2k-1$ hyp, then $\boxed{\textit{err}(\,h\,) \approx \binom{2k-1}{k}\epsilon^k}$

- Ideas:
  - + Subsampling Training Sample (Boosting, Bagging, ...)
  - + Manipulate Input Features
  - + Manipulate Output Targets (ECOC)
  - + Injecting Randomness
  - + Algorithm Specific methods
- & How to combine classifiers
- & Why they work?

## $\boxed{\textbf{1a. Subsample training sample}}$

Given: learner $L(\ \{\langle \mathbf{x}^j, f(\mathbf{x}^j)\rangle\}\ ) \quad = \quad$ classifier

Def'n:

| Learner is **UNSTABLE** if<br>its output classifier undergoes major changes<br>in response to small changes in training data |
| --- |

Eg: Decision-tree, neural network, rule learning alg's

(Stable: Linear regression, nearest neighbor, linear threshold algorithms)

- Subsampling is best for *unstable learners*
- Techniques:
  - Cross-Validated Committees
  - Bagging
  - Boosting

# Simple Subsampling

Given sample $S$ with $m$ instances
learner $L$
constant $K$, . . .

- "Cross-validation committee" [Parmanto/Munro/Doyle'96

```
Divide S into K disjoint sets:    S = ⋃_i s_i
For i=1..K
   Let S_i  =  S − s_i
   Let h_i  =  L(S_i)
Return  h(x)  ≜  majority{h_i(x)}
```

- BAGging = **B**oostrap **AG**gregation [Brieman'96]

```
For i=1..K
   Produce S_i by drawing m instances uniformly
     with replacement

       % |S_i| ≈ 0.632 = (1 − 1/e) of |S|,
       % with many duplicates

   Let h_i = L( S_i )
Return  h(x)  ≜  majority{ h_i(x) }
```

# 1a, iii: Boosting

- Focus effort on problematic instances

- Get classifier $h_i$ on iteration $i$

  For iteration $i+1$
  Give more weight to instances that $h_i$ got wrong

  Final classifier is weighted average of $h_i$'s
  weighted by $h_i$'s error (wrt its distr'n)

- PROVABLY BOOSTS weak learner,
  to produce arbitrarily good one! [Shapire]

- Empirical comparison [Freund/Schapire'96]
  raw C4.5, vs
  C4.5 + BAGging, vs
  C4.5 + Boosting:

  Boosting seems best (UCI Datasets)

- ...but problems w/noisy data [Quinlan'96]

# AdaBoost.M1 Algorithm

**AdaBoost.M1** `algorithm`

```
Input:  labeled examples:  S = {⟨x_i; y_i⟩}_{i=1}^m
        Learn (a learning algorithm)
        a constant L ∈ 𝒩
```

`for all` $i$:

$w_1(i) := 1/m$ *% initialize weights*

`for` $\ell = 1..L$ `do`

`for all` $i$: $p_\ell(i) := \frac{w_\ell(i)}{\sum_i w_\ell(i)}$ *% normalize weights*

$h_\ell :=$ Learn$(p_\ell)$ *% call* Learn *on weights*

$\epsilon_\ell := \sum_i p_\ell(i)\ [[h_\ell(\mathbf{x}_i) \neq y_i]]$ *% calculate* $h_\ell$*'s error*

`if` $\epsilon_\ell > \frac{1}{2}$ `then`

`L :=` $\ell - 1$

`break` *% Exit from this "for" loop*

`end if`

$\beta_\ell := \frac{\epsilon_\ell}{(1-\epsilon_\ell)}$

`for all` $i$:

$$w_{\ell+1}(i) := \begin{cases} w_\ell(i) & \texttt{if } h_\ell(x_i) \neq y_i \\ w_\ell(i)\,\beta_\ell & \texttt{otherwise} \end{cases}$$

*% Increase* $x_i$ *weight*
*% if* $h_\ell$ *got it wrong!*

`end for`

`Output:` $$h_f(\mathbf{x}) := \underset{y\in Y}{\text{argmax}} \sum_{t=1}^{L} \left(\log \frac{1}{\beta_t}\right) [[h_t(\mathbf{x}) = y]]$$

---

[[E]] is 1 if E is true and 0 otherwise

# 1b: Manipulate INPUT FEATURES

- Different learners see different subsets of features
  (of each of the training instances)

Eg: $\exists$ 119 features for classifing volcanoes on Venus
Divide into 8 disjoint subsets (by hand)...
and use 4 networks for each
$\Rightarrow$ 32 NN classifiers

Did VERY well [Cherkauer'96]

- Tried w/sonar dataset – 25 input
  Did NOT work [Tumer/Ghost'96]

- Technique works best when
  input features highly redundant

# 1c: Manipulate OUTPUT Targets

- Spse $K$ outputs $Y = \{y_1, \ldots y_K\}$

a. Could learn 1 classifier, into $Y$ ($|Y|$ values)

b. Or could learn $K$ binary classifiers:

$y_1$ vs $Y - y_1$;
$y_2$ vs $Y - y_2$;
...
then vote.

c. Build $\ln K$ binary classifiers

$h_i$ specifies $i^{th}$ bit of index $\in \{0, 1, \ldots, K-1\}$

Each $h_i$ sub-classifier splits output-values into 2 subsets (*e.g.*, $h_0(\mathbf{x})$ is $\begin{cases} 1 & \text{if ``}y_0, \ldots, y_7\text{''} \\ 0 & \text{if ``}y_8, \ldots, y_{15}\text{''} \end{cases}$

$h_1(\mathbf{x})$ is $\begin{cases} 1 & \text{if ``}\{y_0 - y_3,\ y_8 - y_{11}\}\text{''} \\ 0 & \text{otherwise} \end{cases}$

$h_2(\mathbf{x})$ is $\begin{cases} 1 & \text{if ``}\{y_0, y_1,\ y_4, y_5,\ y_8, y_9,\ y_{12}, y_{13}\}\text{''} \\ 0 & \text{otherwise} \end{cases}$

...)

## Error Correcting Output Code

- Why not $> \ln K$ binary classifiers ...
  "Error-Correcting Codes" (some redundancy)
  [Dietterich/Bakiri'95]

- Each $h_i(\mathbf{x})$ "votes" for some output-values
  Eg, $h_0(\mathbf{x})$ gives 1 to each of $y_8, y_9, \ldots, y_{15}$
  (0 for other values)
  $h_1(\mathbf{x})$ gives 1 to each of $y_0, y_1, \ldots, y_8, y_9, \ldots$
  ...

  Return $y_i$ with most votes

- Or... view $\langle h_0(\mathbf{x}), \ldots h_m(\mathbf{x})\rangle$ as code-word;
  take $y_i$ with nearest codeword

- Can combine with AdaBoost [Schapire'97]
  gets better!

# 1d: Injecting Randomness

- For Neural Nets:

1. Different random initial values of weights

   But really independent?

   Empirical test: [Pamanto, Munro, Doyle 1996]
   Cross-validated committees BEST,
   then Bagging, then Random initial weights

2. Add 0-mean Gaussian noise to
   input features [Raviv/Intrator'96]

   Draw w/replacement from original data,
   but add noise

   (Large improvement on
   + synthetic benchmark;
   + medical Dx)

# Randomness – w/ C4.5

- C4.5 uses Info Gain to decide which attribute to split on

  (Issues wrt REAL values)

  Why not consider top 20 attributes; choose one at random?

  ⇒ Produce 200 classifiers (same data)
  To classify new instance: Vote.

  Empirical test: [Dietterich/Kong 1995]
  Random better than bagging, than single C4.5

- FOIL (for learning Prolog-like rules)

  Chose any test whose info gain within 80% of top

  Ensemble of 11
  STATISTICALLY BETTER
  than 1 run of FOIL [Ali/Pazzani'96]

# Model Averaging

- Why have SINGLE hypothesis?

  Why not use SEVERAL HYPOTHESES $\{h_i\}$
  combined with posterior prob?

Given: $\star$ data $S = \{\langle x_j,\ f(x_j)\rangle\}$
$\star$ unlabeled instance $\mathbf{x}$
$\star$ (PRIOR DISTR'N over hyp $P(\,h_i\,)$)

Compute $P(\,y\,|\,\mathbf{x},\ S\,)$

$$\ldots = \textstyle\sum_i P(\,y,\ H = h_i\,|\,\mathbf{x},\ S\,)$$
$$= \textstyle\sum_i P(\,H = h_i\,|\,\mathbf{x}, S\,)\ P(\,y\,|\,H = h_i,\ \mathbf{x},\ S\,)$$
$$= \textstyle\sum_i P(\,H = h_i\,|\,S\,)\ P(\,y\,|\,H = h_i,\ \mathbf{x}\,)$$
$$= \frac{1}{P(\,S\,)} \sum_i P(\,S\,|\,h_i\,)\ P(\,h_i\,)\ P(\,h_i(\mathbf{x}) = y\,)$$

$P(\,S\,|\,h_i\,)$ is prob of data, given $h_i$
$P(\,h_i\,)$ is prior prob of $h_i$

# Monte Carlo Markov Chain

Challenge: How to get set of $h_i$'s ?

- Monte Carlo Markov Chain [Neal'93; MacKay'92]

  Start with (random) $h_0$,

  produce new $h_{i+1}$ by randomly modifying $h_i$

  (In NN: perhaps adjust on weight;
  for DT, perhaps interchange parent and child, or
  replace one node with another)

  Eventually, get representative set of $\{h_i\}$
  ... drawn from $P( h_i \mid S )$

- Compute, for each $y$:
  $$P( y \mid \mathbf{x},\ S ) = \sum_i P( h_i \mid S )\, P( h_i(\mathbf{x}) = y )$$

  Return **argmax**

# Why this PAC-Learning Model?

- PAC-Learning Framework:
  + Initial Hypothesis Space

    $$\mathcal{H}_0 \;=\; \underline{\hspace{6cm}}$$

  + Given evidence...

    $$\mathcal{H}_D \;=\; \underline{\hspace{6cm}}$$

  + Which hypothesis?

    $$h^* \in \mathcal{H}_D \; \underline{\hspace{6cm}}$$

  + Use, to classify unlabeled example $x$?

    $$c \;=\; h^*(\mathbf{x})$$

- Issues:

  Q1: Why this initial hypothesis?

  Why "discrete": $h_1 \in \mathcal{H}$, $h_2 \notin \mathcal{H}$?

  Q2: Is this best use of training instances?

  Consistency problematic, if noisy data!

  Q3: Why select only 1 (consistent) hypothesis?

# Why Select 1 hypothesis *(Q3)*?

- Perhaps keep *"Version Space"*
  $\equiv$ ALL consistent hypotheses

$$\mathcal{H}_D \;=\; \mathcal{H}(\{\langle x_i, c_i \rangle\}) \;=\; \{\, h \in \mathcal{H} \,|\, h(x_i) = c_i \,\}$$

Let: $Hyp(x, c) \;=\; \{\, h \in \mathcal{H}_D \,|\, h(\mathbf{x}) = c \,\}$

Use:

Set $\quad Class(\mathbf{x}) \;=\; \underset{c}{\text{argmax}}\{\, |\, Hyp(x, c) \,|\, \}$

Q3': Why "1 hypothesis, 1 vote" ?

Q3'': What if hypothesis has doubts

$$h(\mathbf{x}) \;= \begin{cases} 0.3 & \text{w prob } 1/2 \\ 0.82 & \text{w prob } 1/2 \end{cases}$$

- *Why not really include probabilities?*

# $\boxed{\text{Bayesian Approach}}$

- Bayesian Framework:
  + Initial Hypothesis Space
  $$\mathcal{H}_0 = \underline{\hspace{6cm}}$$
  + Given evidence...
  $$\mathcal{H}_D = \underline{\hspace{6cm}}$$
  + Which hypothesis? [Below]
  + Use, to classify unlabeled example $x$?

$$\boxed{P(\,class(\mathbf{x}) \;=\; c \,|\, D\,) = \sum_h P(\,h(\mathbf{x}) = c\,) \cdot P(\,h \,|\, D\,)}$$

- **Model Averaging!**

Notes: Allows "stochastic" $h$'s

Simpler if $h$ is function, ...

If $h(\mathbf{x}) \in \Re$, can return $c(\mathbf{x}) \in \Re$:

$$\boxed{c(\mathbf{x}) \;=\; E_h[h(\mathbf{x})] \;=\; \sum_h c \cdot P(\,h(\mathbf{x}) = c\,) \cdot P(\,h \,|\, D\,)}$$

or [mean, variance]; or tails; or ...

MAP: If $\exists\ h^* \in \mathcal{H}$ s.t. $P(\,h^* \,|\, D\,) \approx 1$

get $P(\,class(\mathbf{x}) \;=\; c \,|\, D\,) \;\approx\; P(\,h^*(\mathbf{x}) = c\,)$

So just use $h^*$ !

$\Rightarrow$ Use $h^{MAP} \;=\; \text{argmax}_h\{P(\,h \,|\, D\,)\}$

Called *"Maximum A Posterior"*

# 1e: Algorithm Specific (NNs)

**Seek "diverse" population of NNs**

- Simultaneously train several NN's
  with penalty for correlations.

  Backprop minimizes error function =
  sum of MSE and correlations [Rosen'96]

- Use operators to build new structures
  keep $R$ "best", based on
  DIVERSITY + ACCURACY
  (like GA [Opitz/Shavlik'96])

- Give different NNs different auxiliary tasks,
  (eg, predict one input feature)
  in addition to primary task

  Backprop use BOTH in error, so produces different nets [Abu-Mostafa'90; Caruana'96]

- For each $\langle x_i, y_i \rangle$, re-train $NN_j$ with
  $\langle x_i, \langle y_i, 1 \rangle \rangle$ if $NN_j(x_i)$ closest to $y_i$
  $\langle x_i, \langle y_i, 0 \rangle \rangle$ otherwise

  (So diff NNs get different training values, to help NN learn where it performs best) [Munro/Parmanto'97]

# 1e: Algorithm Specific (NN #2)

- Person identifies which region of input space
  (Highway, 2lane-road, dirt-road, ...)

  Train $\text{NN}_i$ for $\text{region}_i$
  eg, to steer, . . .

- Each $\text{NN}_i$ also learns to reconstruct image
  Same intermediate layer!

- When "running", each $\text{NN}_i$
  proposes steering direction,
  reconstruction of image

  Take direction from $\text{NN}_i$
  with best reconstruction [Pomerleau]

- Also: train on "bad" situation,
  by distorting image, and defining correct label

# 1e: Algorithm Specific (DTs, ...)

- "Option tree":

  Decision Tree whose internal nodes have $> 1$ splits

  each producing own sub-decision-tree

  (Eval: go down each, then vote) [Buntine'90]

- Empirical: accuracy $\approx$ bagged C4.5 trees
  but MUCH more understandable

---

- Can try different modalities
  but not clear how DIVERSE they will be

  (Should check for both accuracy and diversity

  ... cross-validation)

# Combining Classifiers

- **Unweighted voting**

  bagging, ErrorCorrecting, ...

  If each $h_\ell$ produces class prob. estimates,

  $P(\, f(\mathbf{x}) = y \,|\, h_\ell \,)$

  can add these

  $$P(\, f(\mathbf{x}) = y \,) \;=\; \sum_\ell P(\, f(\mathbf{x}) = y \,|\, h_\ell \,)\, P(\, h_\ell \,)$$

  Forecasting lit. suggests this is very robust [Clemen'89]

- **Weighted voting**

  *Regression:*

  Use *least squares regression* to find weights that max accuracy on training data

  $\Rightarrow \boxed{h_\ell\text{'s weight} \quad \propto \quad 1/\mathrm{Var}(h_\ell)}$

  should also deal w/ less correlated subset

  *Classification:*

  derive weights from performance on hold-out set

  or Bayesian approach [Ali/Pazzani'96; Buntine'90]

# Combining Classifiers, II

- **Gating** [Jordan/Jacobs'94]
  Learn classifier's $\langle h_1, \ldots, h_m \rangle$
  output$(x) \;=\; \sum_\ell w_\ell \cdot h_\ell(x)$

  "soft-max": $\boxed{w_\ell \quad = \quad e^{v_\ell \cdot \mathbf{x}} / \sum_u e^{v_u \cdot \mathbf{x}}}$

  Problem: lot of parameters to learn
  $\{v_\ell\}$, as well as params for all $h_\ell$s

- **Stacking** [Wolper'92; Breiman'96]
  Given learners $\{ \; L_i(\cdot) \; \}$,
  obtain $h_i = L_i( \; S \; )$.

  Want classifier $h(x) \equiv h_*( \; h_1(\mathbf{x}), \ldots, h_L(\mathbf{x}) \; )$

  Let $h_\ell^{(-i)} = L_\ell(S - s_i)$
  so $L \times |S|$ classifiers

  Let $h_\ell^{(-i)}(\, \mathbf{x}_i \,) = \widehat{y}_i^\ell$

  Now learn $h_*$ from $\langle \; \langle \widehat{y}_i^1, \ldots, \widehat{y}_i^L \rangle, \; y_i \; \rangle_i$

# Why Ensembles Work?

Uncorrelated errors (made by indiv. classifiers)
removed by voting

**But:** 1. Why should we be able to find ensembles of classifiers that make uncorrelated errors?

2. Why not just single classifier?

**Background:** learner searches space $\mathcal{H}$ of hyp's
in gen'l, removing inconsistent $h_i$'s from $\mathcal{H}$

Let $VS(\mathcal{H}, S) \subset \mathcal{H}$ be hyp's left after $S$

Why ensembles?

# Why Ensembles?

1. **Sample complexity:**
   $|\mathcal{H}|$ is so large that $VS(\mathcal{H}, S)$ still large.
   Need to "blur" them together,
   rather than take one.

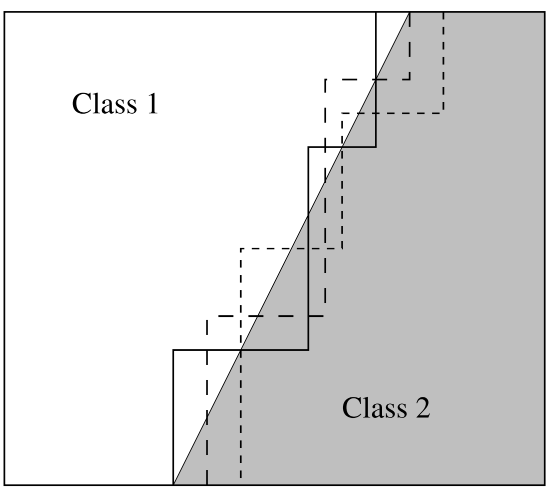
2. **Computational complexity:**
   Computing best member of $VS(\mathcal{H}, S)$ is NP-hard, so hill-climb.

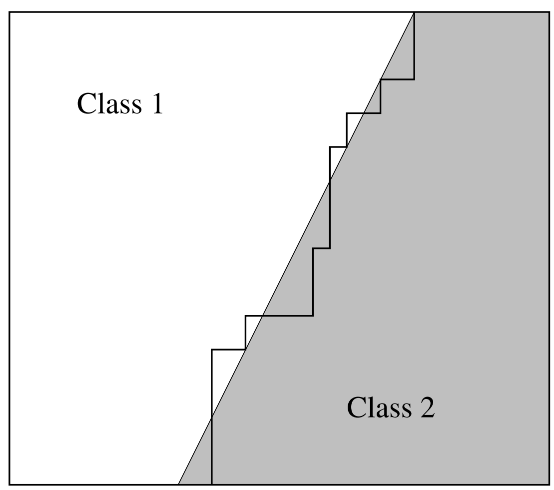
   Ensembles compensate for
   imperfect optimization

3. **Expressiveness:**
   Spse $\mathcal{H}$ does not include good approx to $f$
   ($\neg \exists h \in \mathcal{H}$ *err*( $h$ ) $\approx 0$ )

   Combinations of $h_i$ may overcome
   inadequacies in $\mathcal{H}$

# Combining DT's Boundaries



# Issues/Problems with Ensembles

- Specific Problems
  - AdaBoost is good way to construct
    ensemble of DTs

    But if data noisy:
    AdaBoost places high weight on
    incorrectly-labeled data
    $\Rightarrow$ constructs bad classifier
  - ErrorCorrected Output does not work well with local algs (like nearest neighbor)

  ? Combination of Ensemble methods

| Learning Algorithms | $\times$ | Combining Process |
|---|---|---|

- General Problem:
  - lots of memory to store ensemble
    200 DTs: 59M !
  - how to interpret
    (one DT easy to understand; but 200 of them?)
  - CPU time