Machine learning: lecture 10

Tommi S. Jaakkola
MIT AI Lab
tommi@ai.mit.edu

Topics

- Combination of classifiers: boosting
  - modularity, reweighting
  - AdaBoost, examples, generalization
- Complexity and model selection
  - shattering, Vapnik-Chervonenkis dimension

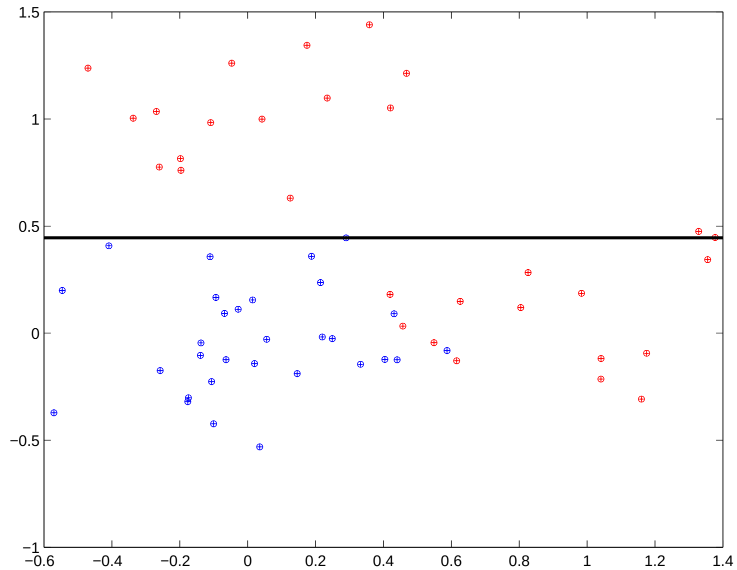
Combination of classifiers

- We wish to generate a set of simple "weak" classification methods and combine them into a single "strong" method
- The simple classifiers in our case are *decision stumps*:

$$h(\mathbf{x};\theta) = \text{sign}(\, w_1\, x_k - w_0\,)$$

where $\theta = \{k, w_1, w_0\}$.

Each decision stump pays attention to only a single component of the input vector



Combination of classifiers con'd

- We'd like to combine the simple classifiers additively so that the final classifier is the sign of

$$\hat{h}_m(\mathbf{x}) = \hat{\alpha}_1\, h(\mathbf{x}; \hat{\theta}_1) + \ldots + \hat{\alpha}_m\, h(\mathbf{x}; \hat{\theta}_m)$$

  where the "votes" α emphasize component classifiers that make more reliable predictions than others

- Important issues:
  - what is the criterion that we are optimizing? (measure of loss)
  - we would like to estimate each new component classifier in the same manner (modularity)

Combination of classifiers con'd

- One possible measure of empirical loss is

$$
\begin{aligned}
&\sum_{i=1}^{n} \exp\{\, -y_i \hat{h}_m(\mathbf{x}_i)\,\} \\
&= \sum_{i=1}^{n} \exp\{\, -y_i \hat{h}_{m-1}(\mathbf{x}_i) - y_i \hat{\alpha}_m h(\mathbf{x}_i;\hat{\theta}_m)\,\} \\
&= \sum_{i=1}^{n} \underbrace{\exp\{\, -y_i \hat{h}_{m-1}(\mathbf{x}_i)\}}_{\text{fixed at stage } m} \exp\{-y_i \hat{\alpha}_m h(\mathbf{x}_i;\hat{\theta}_m)\,\} \\
&= \sum_{i=1}^{n} W_i^{(m-1)} \exp\{-y_i \hat{\alpha}_m h(\mathbf{x}_i;\hat{\theta}_m)\,\}
\end{aligned}
$$

The combined classifier based on $m-1$ iterations defines a weighted loss criterion for the next simple classifier to add

Combination cont'd

- We can simplify a bit the estimation criterion for the new component classifiers

  When $\alpha_m \approx 0$ (low confidence votes)

  $$\exp\{-y_i\alpha_m h(\mathbf{x}_i;\theta_m)\} \approx 1 - y_i\alpha_m h(\mathbf{x}_i;\theta_m)$$

  and our empirical loss criterion reduces to

  $$\approx \sum_{i=1}^n W_i^{(m-1)} \left(1 - y_i\alpha_m h(\mathbf{x}_i;\theta_m)\right) =$$

  $$= \sum_{i=1}^n W_i^{(m-1)} - \alpha_m \left(\sum_{i=1}^n W_i^{(m-1)} y_i h(\mathbf{x}_i;\theta_m) \right)$$

  We could choose each new component classifier to optimize a weighted agreement

Possible algorithm

- At stage m we find $\hat{\theta}_m$ that maximize (or at least give a sufficiently high) weighted agreement

$$\sum_{i=1}^{n} W_i^{(m-1)} y_i h(\mathbf{x}_i; \theta_m)$$

where the weights $W_i^{(m-1)}$ summarize the effect from the previously combined $m-1$ classifiers.

- We find the "votes" $\hat{\alpha}_m$ associated with the new classifier by minimizing the weighted loss

$$\sum_{i=1}^{n} W_i^{(m-1)} \exp\{-y_i \alpha_m h(\mathbf{x}_i; \hat{\theta}_m)\}$$

Boosting

- We have basically derived a Boosting algorithm that sequentially adds new component classifiers by reweighting training examples
  - each component classifier is presented with a slightly different problem
- AdaBoost preliminaries:
  - we work with *normalized* weights $\tilde{W}_i$ on the training examples, initially uniform ($\tilde{W}_i = 1/n$)

The AdaBoost algorithm

1: At the k^{th} iteration we find (any) classifier $h(\mathbf{x};\hat{\theta}_k)$ for which the *weighted classification error* ϵ_k

$$\epsilon_k = 0.5 - \frac{1}{2}\left(\sum_{i=1}^{n} \tilde{W}_i^{(k-1)} y_i h(\mathbf{x}_i;\hat{\theta}_k)\right)$$

is better than chance.

2: Determine how many "votes" to assign to the new component classifier: $\hat{\alpha}_k = 0.5\log(\,(1-\epsilon_k)/\epsilon_k\,)$ (decorrelation)

3: Update the weights on the training examples:

$$\tilde{W}_i^{(k)} = \tilde{W}_i^{(k-1)} \cdot \exp\{\,-y_i\hat{\alpha}_k h(\mathbf{x}_i;\hat{\theta}_k)\,\}$$

and renormalize the new weights to one.

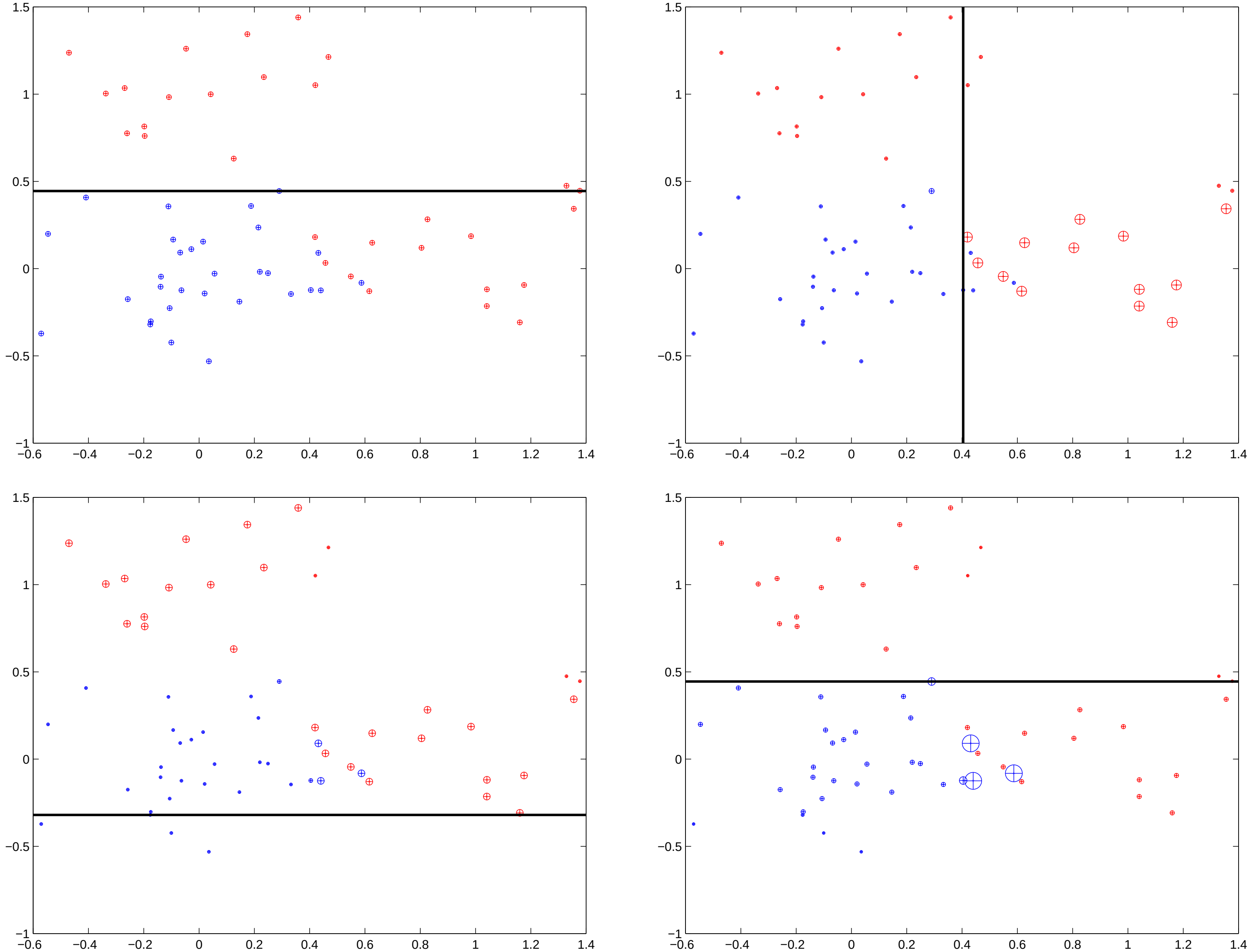
The AdaBoost algorithm cont'd

- The final classifier after m boosting iterations is given by the sign of

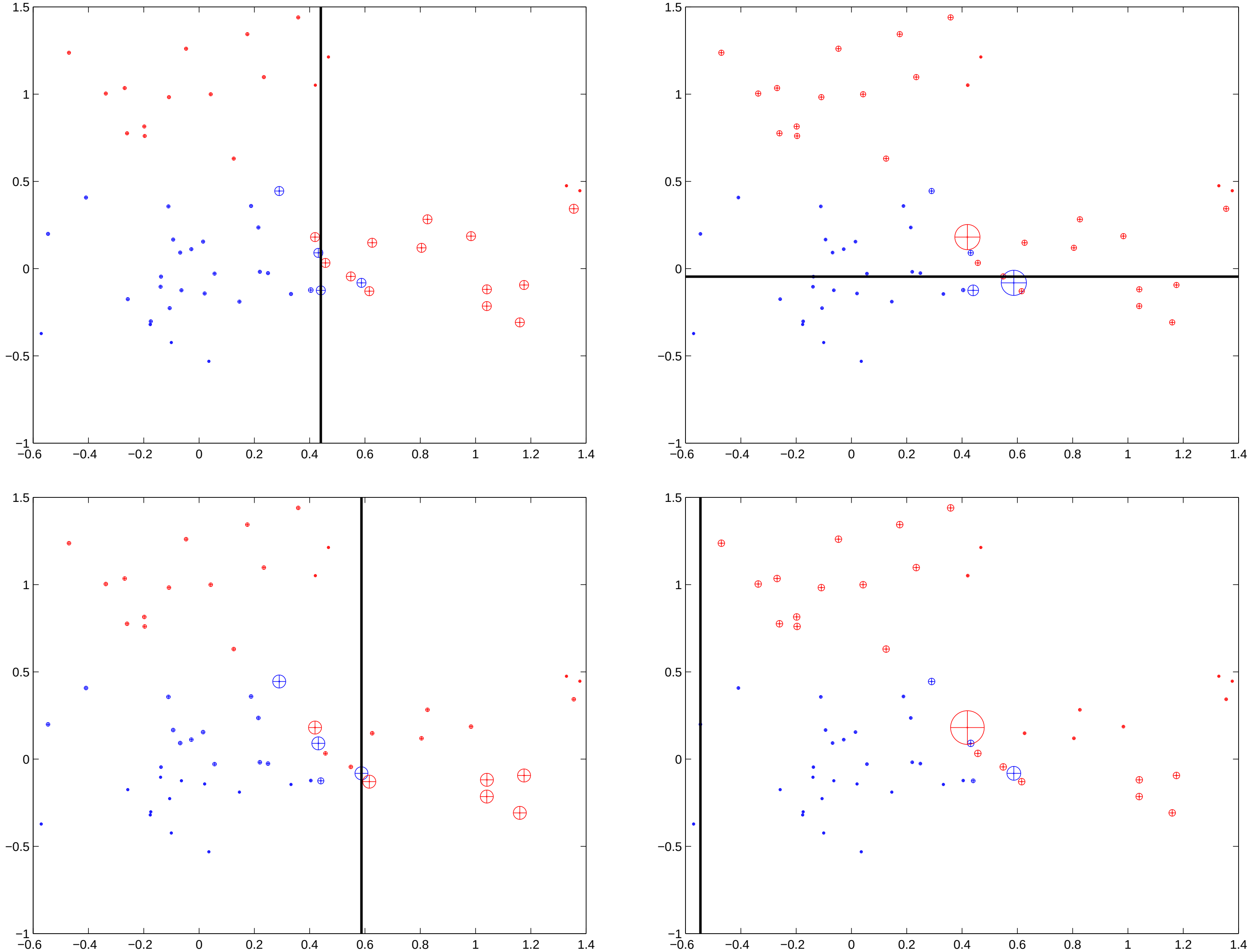
$$\hat{h}(\mathbf{x}) = \frac{\hat{\alpha}_1 h(\mathbf{x};\hat{\theta}_1) + \ldots + \hat{\alpha}_m h(\mathbf{x};\hat{\theta}_m)}{\hat{\alpha}_1 + \ldots + \hat{\alpha}_m}$$

(the votes here are normalized for convenience)

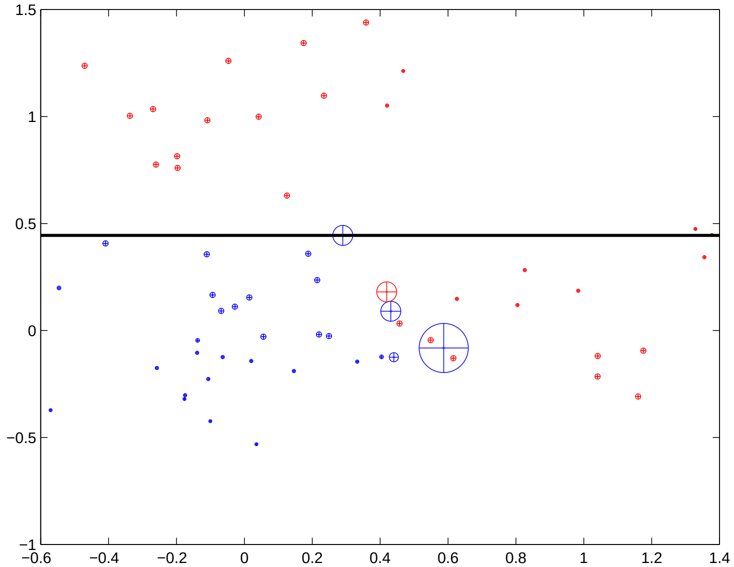
Boosting: example

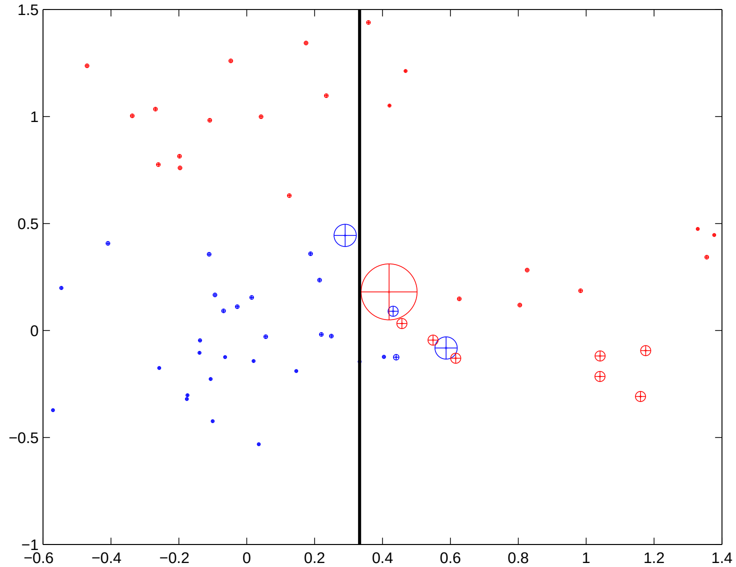


Boosting: example cont'd



Boosting: example cont'd

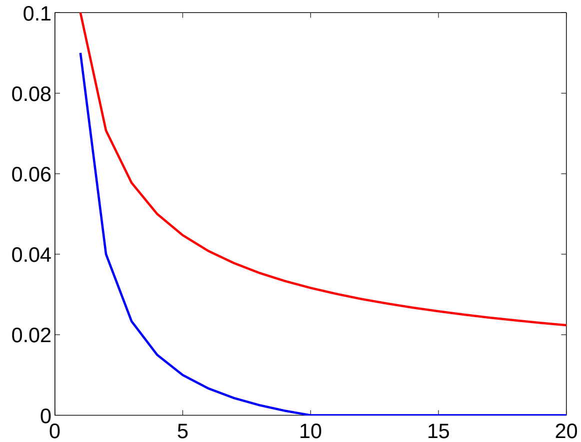




Boosting performance

- Training/test errors for the *combined classifier*

$$\hat{h}(\mathbf{x}) = \frac{\hat{\alpha}_1 h(\mathbf{x}; \hat{\theta}_1) + \ldots + \hat{\alpha}_m h(\mathbf{x}; \hat{\theta}_m)}{\hat{\alpha}_1 + \ldots + \hat{\alpha}_m}$$



What about the error rate of the component classifiers (decision stumps)?

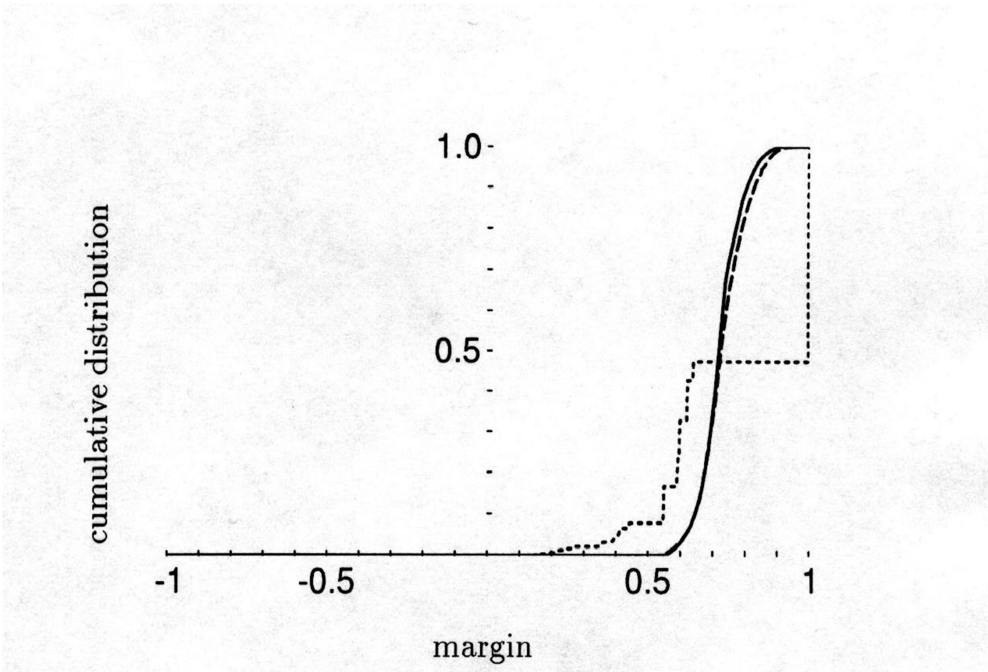
- Even after the training error of the combined classifier goes to zero, boosting iterations can still improve the generalization error!

Boosting and margin

- Successive boosting iterations improve the majority vote or *margin* for the training examples

$$\text{margin for example } i \;=\; y_i \left[\frac{\hat{\alpha}_1 h(\mathbf{x};\hat{\theta}_1) + \ldots + \hat{\alpha}_m h(\mathbf{x};\hat{\theta}_m)}{\hat{\alpha}_1 + \ldots + \hat{\alpha}_m} \right]$$

The margin lies in $[-1, 1]$ and is negative for all misclassified examples.



Topics

- Complexity and model selection
  - shattering, Vapnik-Chervonenkis dimension
  - structural risk minimization (next lecture)

Measures of complexity

- "Complexity" is a measure of a set of classifiers, not any specific (fixed) classifier
- Many possible measures
  - degrees of freedom
  - description length
  - Vapnik-Chervonenkis dimension

  etc.
- There are many reasons for introducing a measure of complexity
  - generalization error guarantees
  - selection among competing families of classifiers

VC-dimension: preliminaries

- **A set of classifiers F:**
  For example, this could be the set of all possible linear separators, where $h \in F$ means that

$$h(\mathbf{x}) = \mathrm{sign}\left(w_0 + \mathbf{w}^T \mathbf{x} \right)$$

  for some values of the parameters $\mathbf{w}, w_0$.

VC-dimension: preliminaries

- **Complexity:** how many different ways can we label n training points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with classifiers $h \in F$?

  In other words, how many distinct binary vectors

  $$[h(\mathbf{x}_1)\; h(\mathbf{x}_2)\; \ldots\; h(\mathbf{x}_n)]$$

  do we get by trying each $h \in F$ in turn?

  $$\begin{array}{cl} [\;\; \text{-1} \quad 1 \quad \ldots \quad 1 \;\;] & h_1 \\ [\;\;\; 1 \quad \text{-1} \quad \ldots \quad 1 \;\;] & h_2 \\ & \ldots \end{array}$$

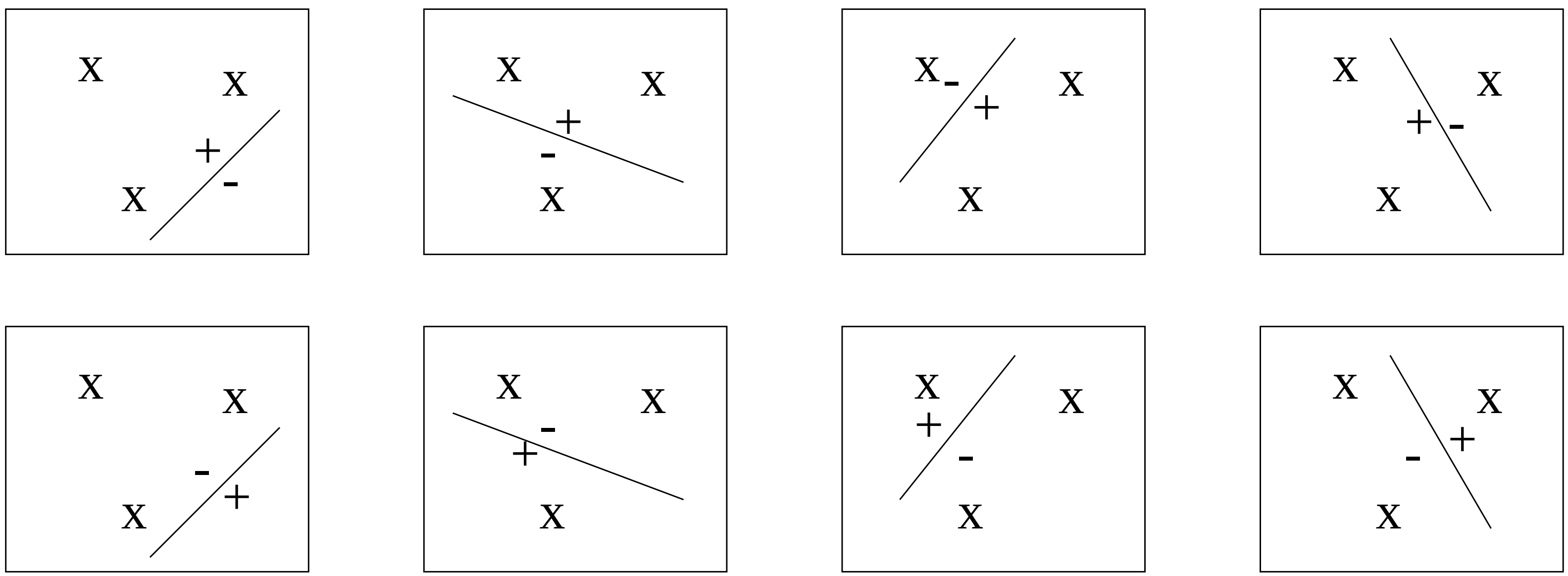
VC-dimension: shattering

- A set of classifiers F *shatters* n points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ if

$$[h(\mathbf{x}_1)\ h(\mathbf{x}_2)\ \ldots\ h(\mathbf{x}_n)], \quad h \in F$$

generates all 2^n distinct labelings.

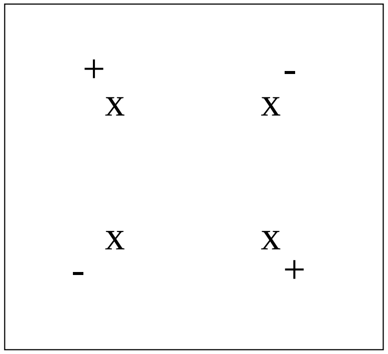
- Example: linear decision boundaries shatter (any) 3 points in 2D



but not any 4 points...

VC-dimension: shattering cont'd

- We cannot shatter 4 points in 2D with linear separators
  For example, the following labeling



  cannot be produced with any linear separator

- More generally: the set of all d-dimensional linear separators can shatter exactly $d + 1$ points

VC-dimension

- *The VC-dimension d_{VC} of a set of classifiers F is the largest number of points that F can shatter*
- This is a combinatorial concept and doesn't depend on what type of classifier we use, only how "flexible" the set of classifiers is

  Example: Let F be a set of classifiers defined in terms of linear combinations of m **fixed** basis functions

  $$h(\mathbf{x}) = \text{sign}\left(w_0 + w_1\phi_1(\mathbf{x}) + \ldots + w_m\phi_m(\mathbf{x}) \right)$$

  d_{VC} is at most $m+1$ regardless of the form of the fixed basis functions.