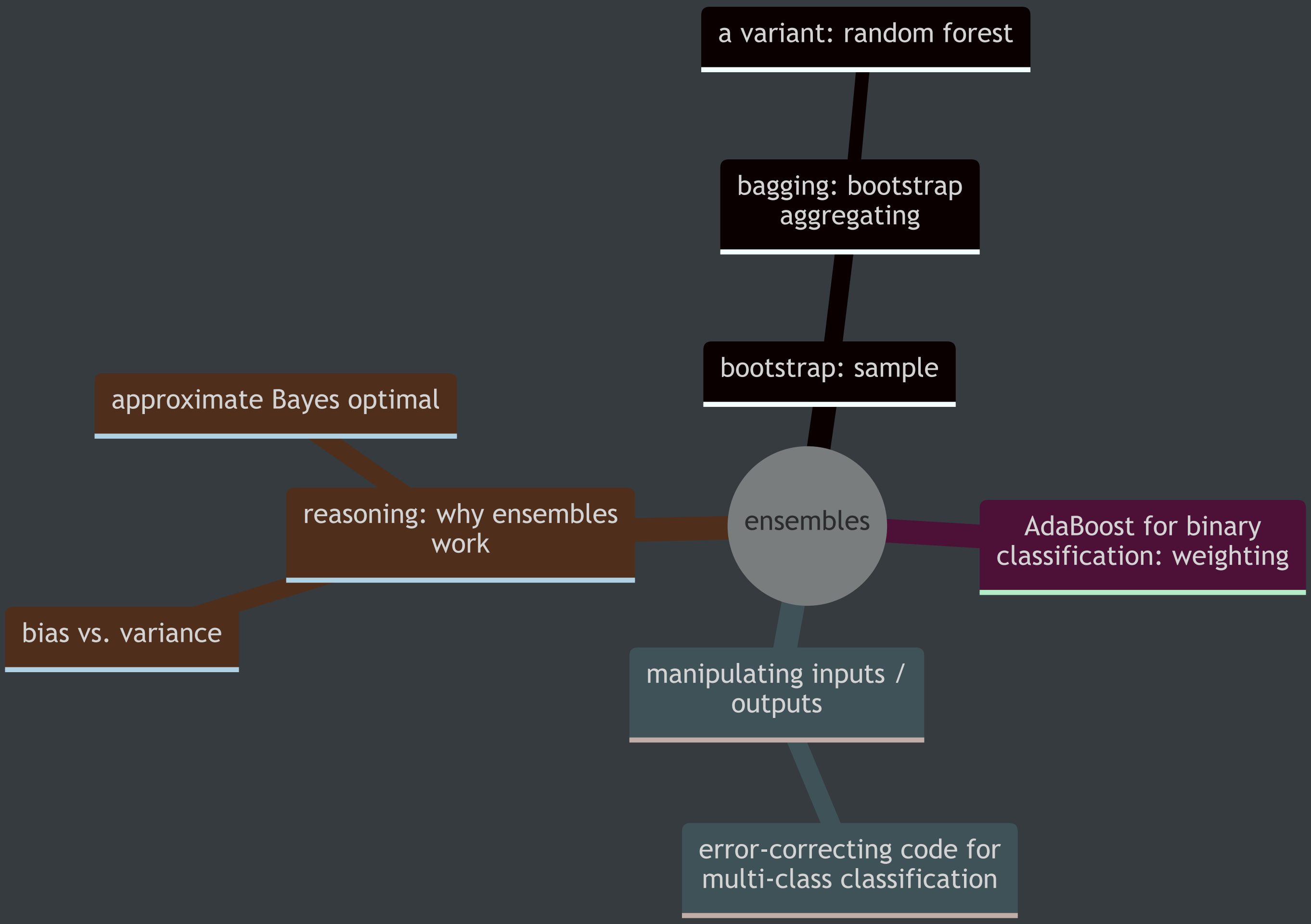
ensemble methods

motivation & overview

can learning more than one model improve performance?



a variant: random forest

bagging: bootstrap aggregating

bootstrap: sample

ensembles

approximate Bayes optimal

reasoning: why ensembles work

bias vs. variance

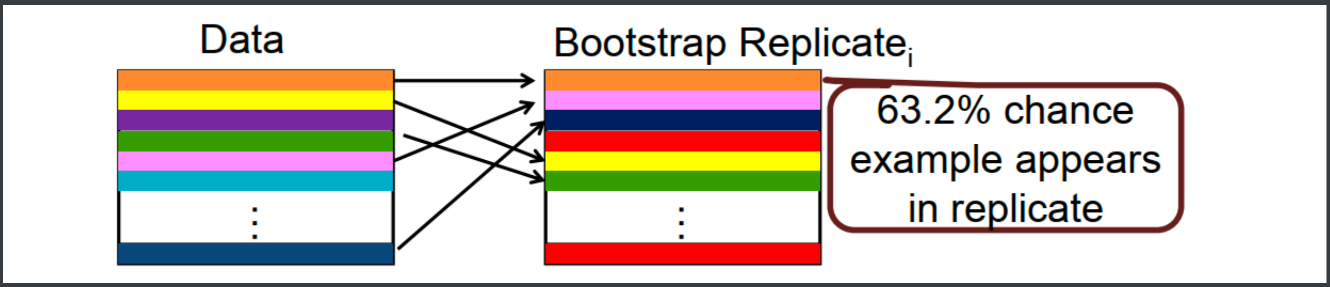
AdaBoost for binary classification: weighting

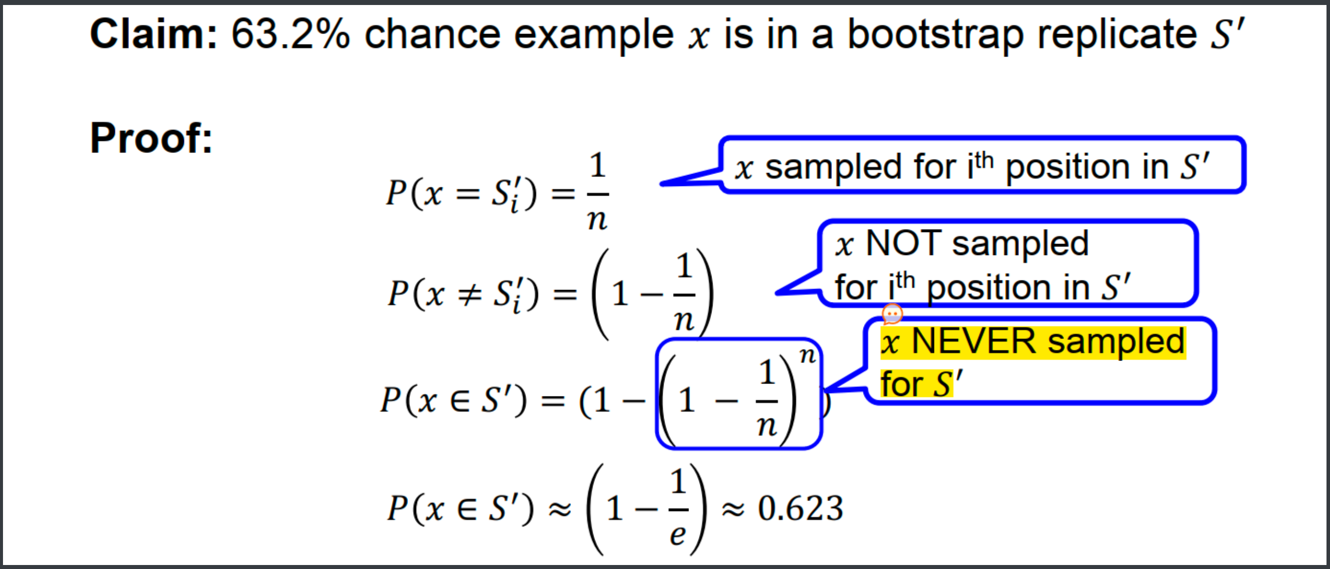
manipulating inputs / outputs

error-correcting code for multi-class classification

bagging & random forests

- bootstrap





Claim: 63.2% chance example x is in a bootstrap replicate S'

Proof:

$$P(x = S_i') = \frac{1}{n}$$

x sampled for ith position in S'

$$P(x \neq S_i') = \left(1 - \frac{1}{n}\right)$$

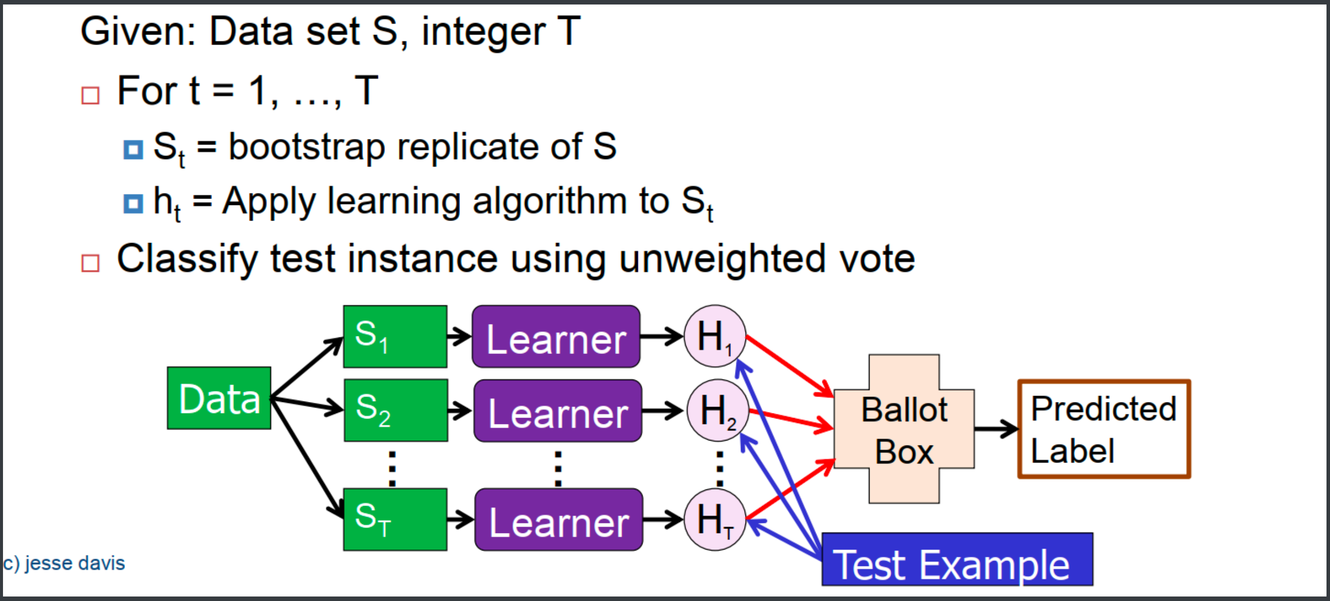
x NOT sampled for ith position in S'

$$P(x \in S') = (1 - \boxed{\left(1 - \frac{1}{n}\right)^n})$$

x NEVER sampled for S'

$$P(x \in S') \approx \left(1 - \frac{1}{e}\right) \approx 0.623$$

- Bagging: **B**ootstrap **Agg**regating



Given: Data set S, integer T

- □ For t = 1, …, T
  - ◘ S_t = bootstrap replicate of S
  - ◘ h_t = Apply learning algorithm to S_t
- □ Classify test instance using unweighted vote

c) jesse davis

- Bagging works best for unstable learners

unstable learners:

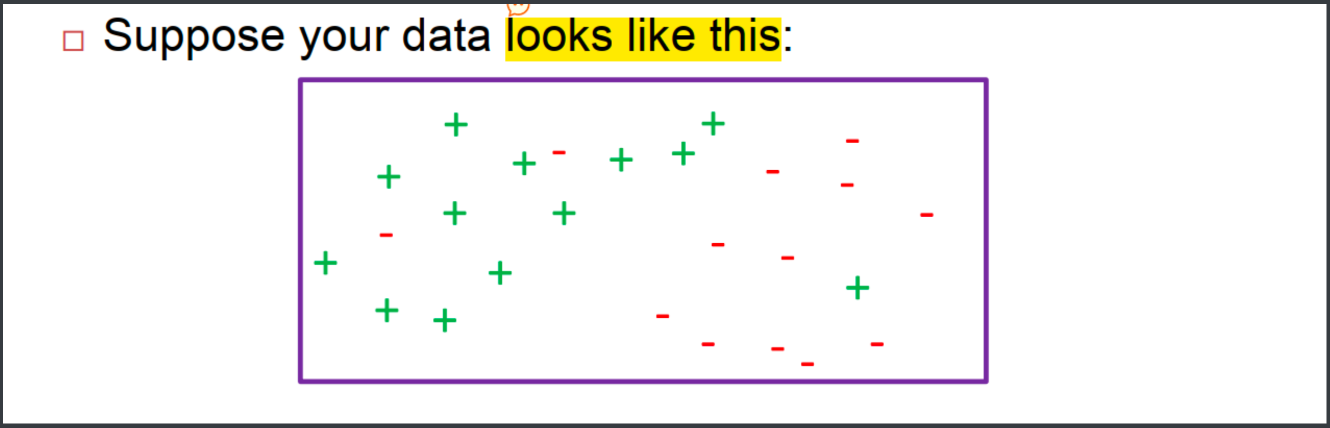
minor variations in training data -> major changes in classifier output

e.g.

unstable: decision tree, neural network, rule learning

stable: linear regression, nearest neighbor, linear threshold algorithms etc.

- intuition: why does bagging work? suppose for a noisy training data

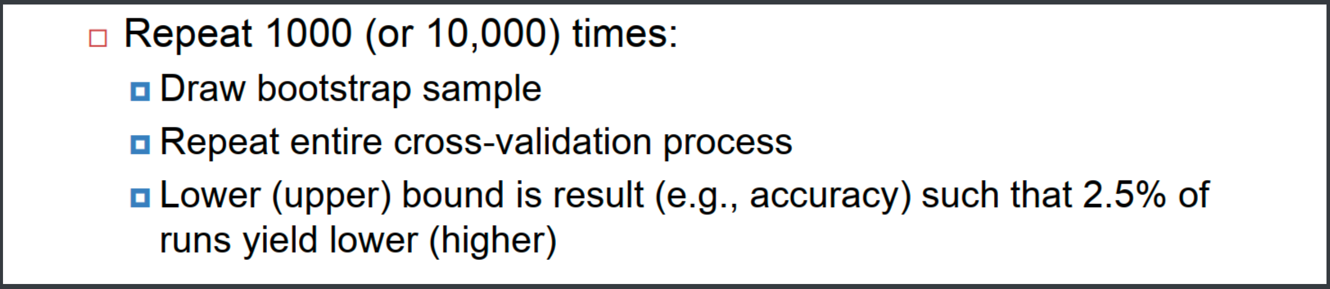


□ Suppose your data looks like this:

noisy samples might not appear in the samples used for training

(although kNN is stable learning)

- for Empirical Confidence Bounds

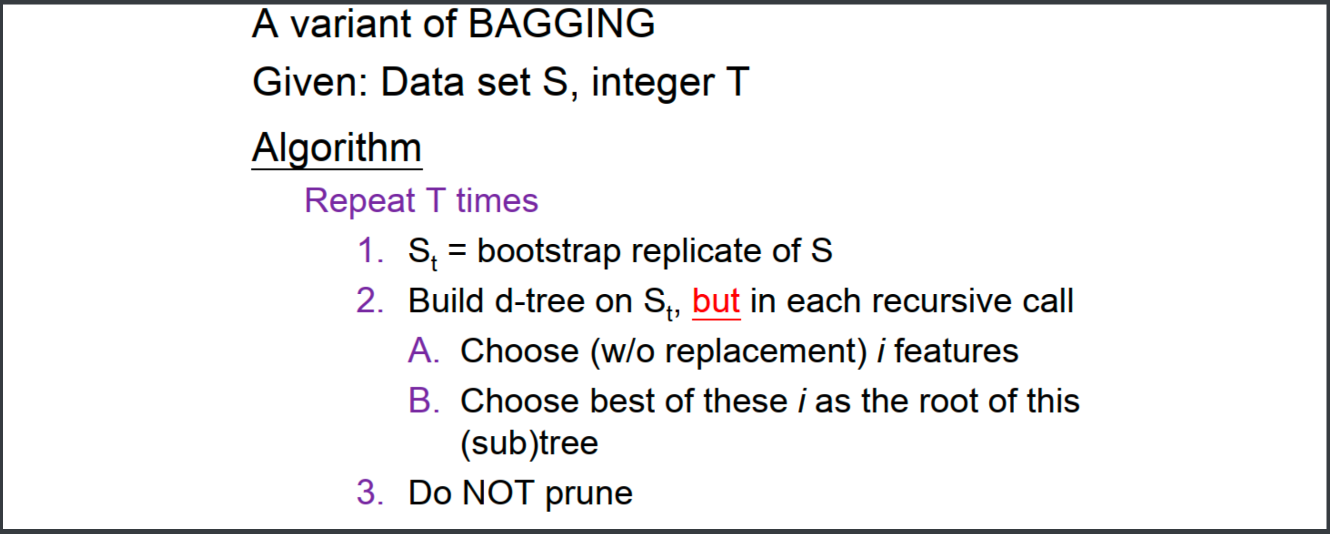


□ Repeat 1000 (or 10,000) times:

- Draw bootstrap sample
- Repeat entire cross-validation process
- Lower (upper) bound is result (e.g., accuracy) such that 2.5% of runs yield lower (higher)

then confidence level = 95%

- Random Forests



A variant of BAGGING

Given: Data set S, integer T

Algorithm

```
Repeat T times
    1. $S_t$ = bootstrap replicate of S
    2. Build d-tree on $S_t$, but in each recursive call
        A. Choose (w/o replacement) *i* features
        B. Choose best of these *i* as the root of this (sub)tree
    3. Do NOT prune
```

1. about choosing the features, increasing i,
   - increases correlation among individual trees (BAD)
   - accuracy of individual trees (GOOD)

   therefore,

   - use a tuning set to pick a good value of *i*

        (note: tuning set is a set for picking the best hyperparameters after training)

2. overall advantages:

   very fast, works for a large number of features, reduces overfitting

3. extreme RF:

   main differences:

   - choose random split points, rather than based on IG etc.
   - test time: unweighted vote

AdaBoost

- Boosting:

  deal with weak learners,

  focuses on the current model **correcting** incorrect predictions

- loss

  error: measure the deviation between the true and predicted values

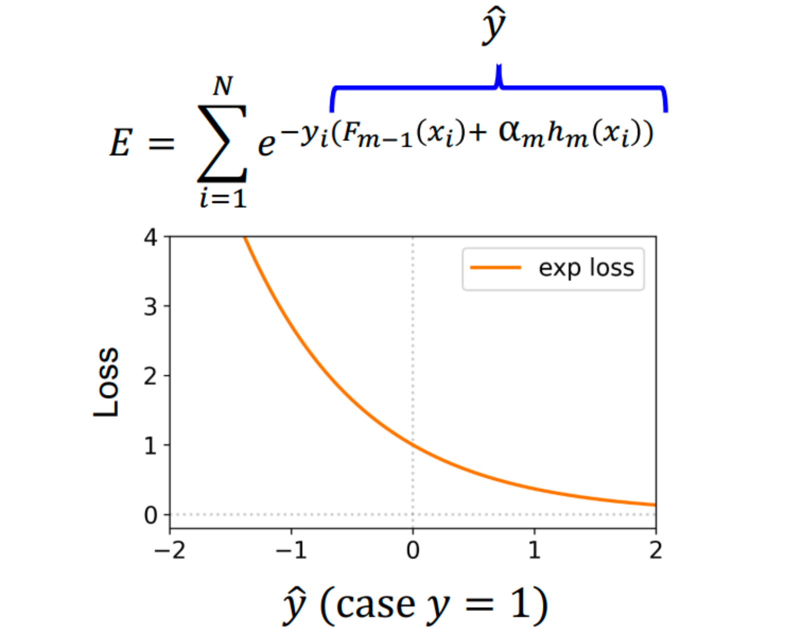
  loss: quantify the impact or consequence of an error

  both always > 0, loss drops quickly

  (note: the error and loss can be identical, e.g. linear regression)

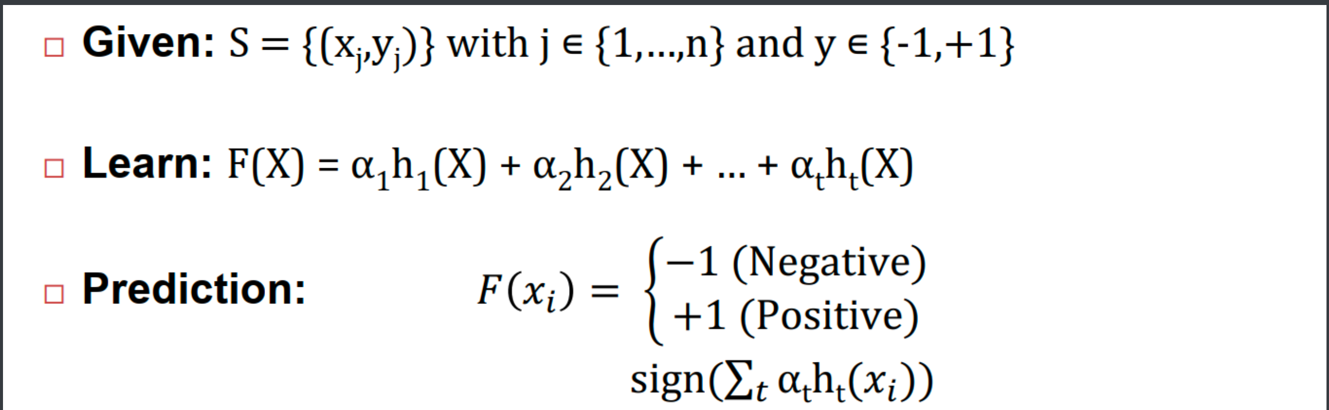
  in AdaBoost,

  loss is small if predicted and true label have same sign (binary classification, only the sign makes a difference)



$$E = \sum_{i=1}^{N} e^{-y_i \overbrace{(F_{m-1}(x_i) + \alpha_m h_m(x_i))}^{\hat{y}}}$$

- goal

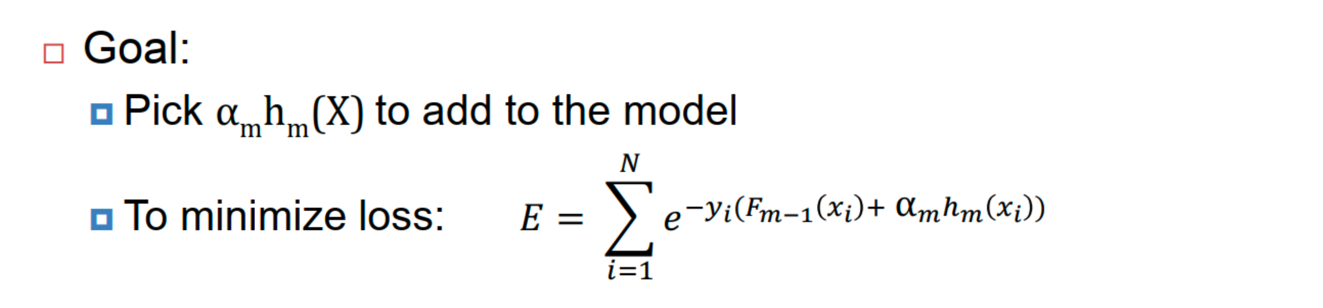


□ **Given:** S = {(x_j,y_j)} with j ∈ {1,…,n} and y ∈ {-1,+1}

□ **Learn:** F(X) = $\alpha_1 h_1$(X) + $\alpha_2 h_2$(X) + … + $\alpha_t h_t$(X)

□ **Prediction:** $F(x_i) = \begin{cases} -1 \text{ (Negative)} \\ +1 \text{ (Positive)} \end{cases}$

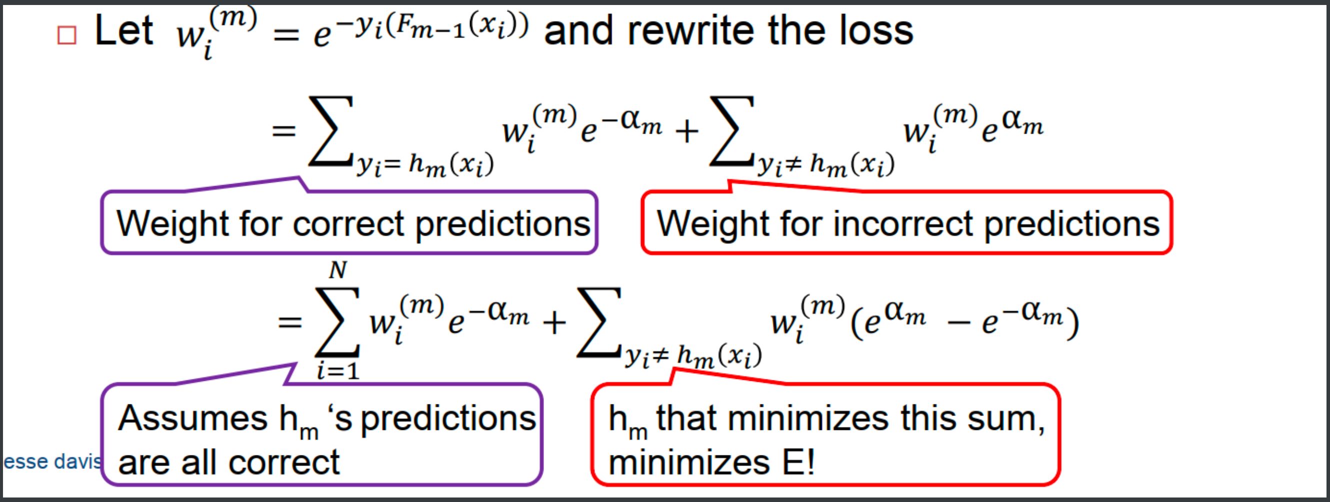
sign($\sum_t \alpha_t h_t(x_i)$)



□ Goal:

- Pick $\alpha_m h_m$(X) to add to the model
- To minimize loss: $E = \sum_{i=1}^{N} e^{-y_i(F_{m-1}(x_i) + \alpha_m h_m(x_i))}$

- how to pick classifier: h_m



□ Let $w_i^{(m)} = e^{-y_i(F_{m-1}(x_i))}$ and rewrite the loss

$$= \sum_{y_i = h_m(x_i)} w_i^{(m)} e^{-\alpha_m} + \sum_{y_i \neq h_m(x_i)} w_i^{(m)} e^{\alpha_m}$$

Weight for correct predictions

Weight for incorrect predictions

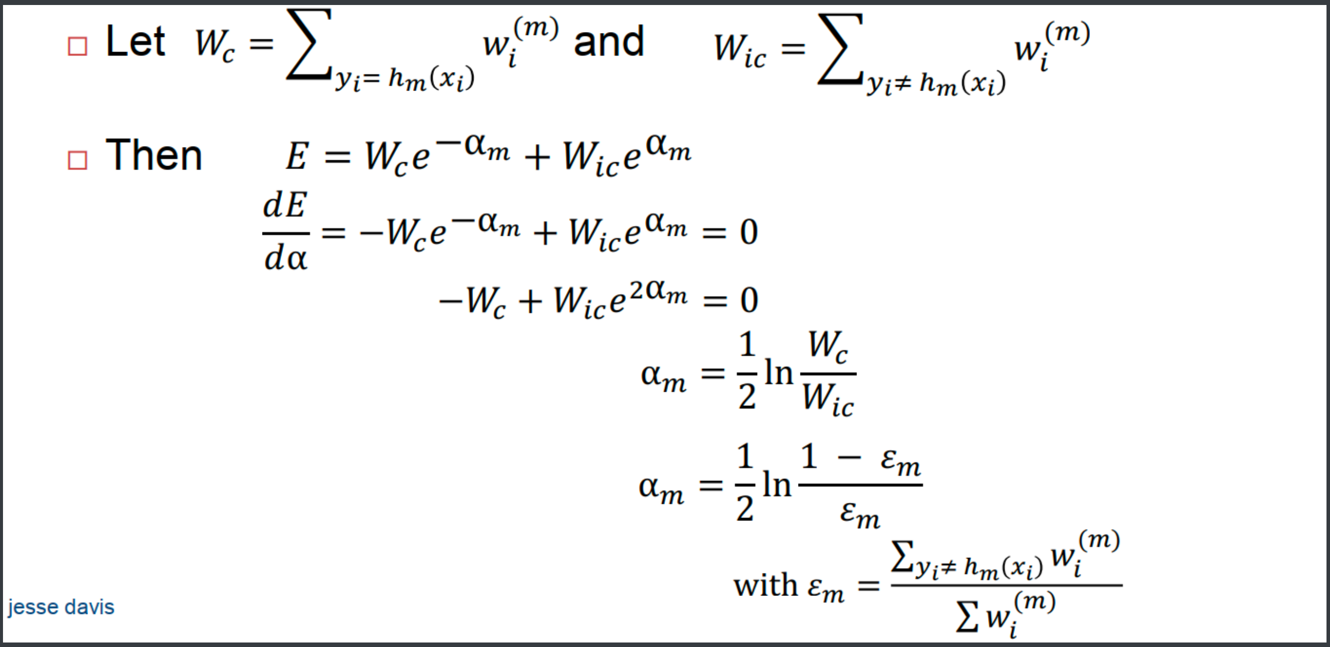
$$= \sum_{i=1}^{N} w_i^{(m)} e^{-\alpha_m} + \sum_{y_i \neq h_m(x_i)} w_i^{(m)} \left(e^{\alpha_m} - e^{-\alpha_m}\right)$$

Assumes h_m 's predictions are all correct

h_m that minimizes this sum, minimizes E!

esse davis

- how to pick weight: α_m



□ Let $W_c = \sum_{y_i = h_m(x_i)} w_i^{(m)}$ and $W_{ic} = \sum_{y_i \neq h_m(x_i)} w_i^{(m)}$

□ Then $E = W_c e^{-\alpha_m} + W_{ic} e^{\alpha_m}$

$$\frac{dE}{d\alpha} = -W_c e^{-\alpha_m} + W_{ic} e^{\alpha_m} = 0$$

$$-W_c + W_{ic} e^{2\alpha_m} = 0$$

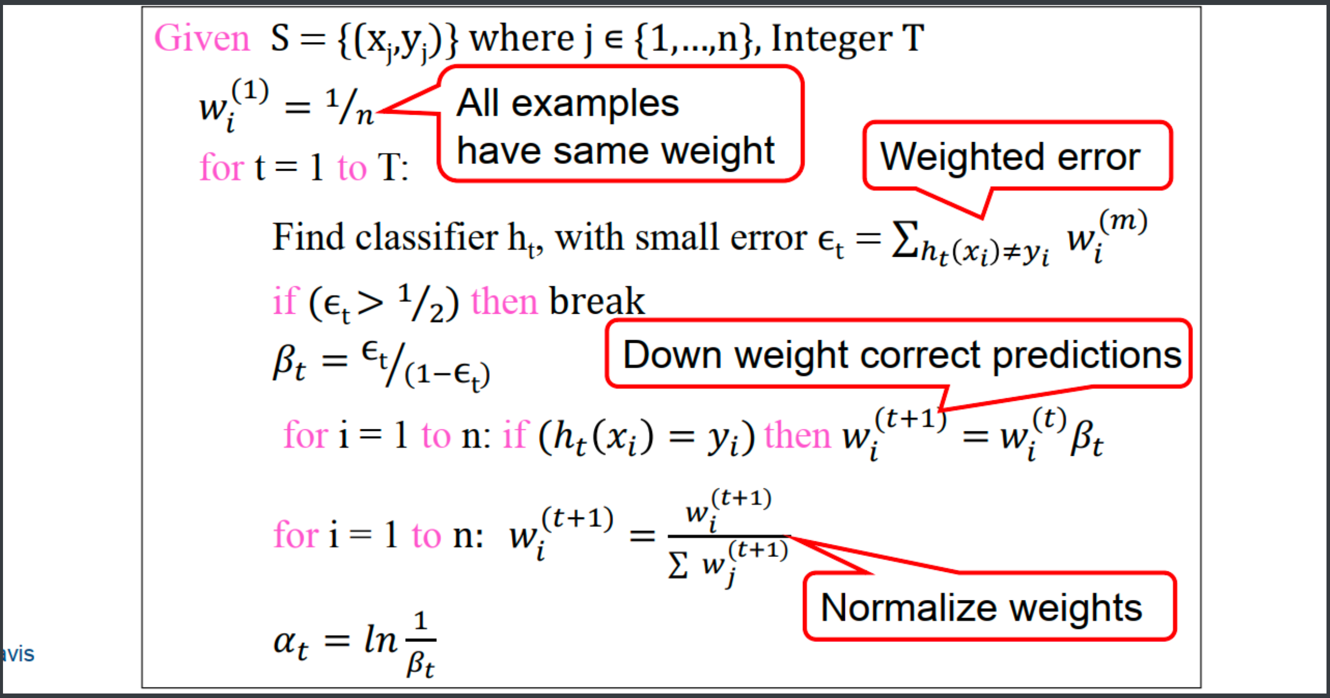
$$\alpha_m = \frac{1}{2} \ln \frac{W_c}{W_{ic}}$$

$$\alpha_m = \frac{1}{2} \ln \frac{1 - \varepsilon_m}{\varepsilon_m}$$

$$\text{with } \varepsilon_m = \frac{\sum_{y_i \neq h_m(x_i)} w_i^{(m)}}{\sum w_i^{(m)}}$$

jesse davis

- complete algorithm



```
Given S = {(x_j, y_j)} where j ∈ {1,…,n}, Integer T
    $w_i^{(1)} = 1/n$
    for t = 1 to T:
        Find classifier $h_t$, with small error $\epsilon_t = \sum_{h_t(x_i) \neq y_i} w_i^{(m)}$
        if ($\epsilon_t > 1/2$) then break
        $\beta_t = \epsilon_t / (1-\epsilon_t)$
        for i = 1 to n: if ($h_t(x_i) = y_i$) then $w_i^{(t+1)} = w_i^{(t)} \beta_t$
        for i = 1 to n: $w_i^{(t+1)} = \frac{w_i^{(t+1)}}{\sum w_j^{(t+1)}}$
        $\alpha_t = ln \frac{1}{\beta_t}$
```

All examples have same weight

Weighted error

Down weight correct predictions

Normalize weights

avis

There will be T classifiers.

- important practical details

1. typically use depth bounded decision tree
2. for weighted instances:

   solution 1:

   sample a larger set of unweighted instances,

   according to the **weight distribution**,

   and run learner on this new dataset

   solution 2:

   adapt learner,

   e.g. weighted counts for split criteria for decision trees

   基尼指数的加权版本

   基尼指数公式：

   $$G = \sum_i p(i)(1 - p(i))$$

   在引入样本权重后，每个样本的贡献变为：

   $$G_{weighted} = \sum_i w_i \cdot p(i) \cdot (1 - p(i))$$

3. training rounds & margins

margin = (正类的加权票数)–(负类的加权票数)，表示信心（置信度）；

AdaBoost 不容易过拟合，更多轮的训练会增加 margin。大 margin 意味着弱分类器之间更一致，预测结果更有信心。模型泛化能力强；

4. AdaBoost (only binary classification) variations

   variations for multiclass tasks & regression

Manipulating inputs / outputs

input

different learners see different subsets of features

but does not work that well

output

- multiclass problem

  solution 1:

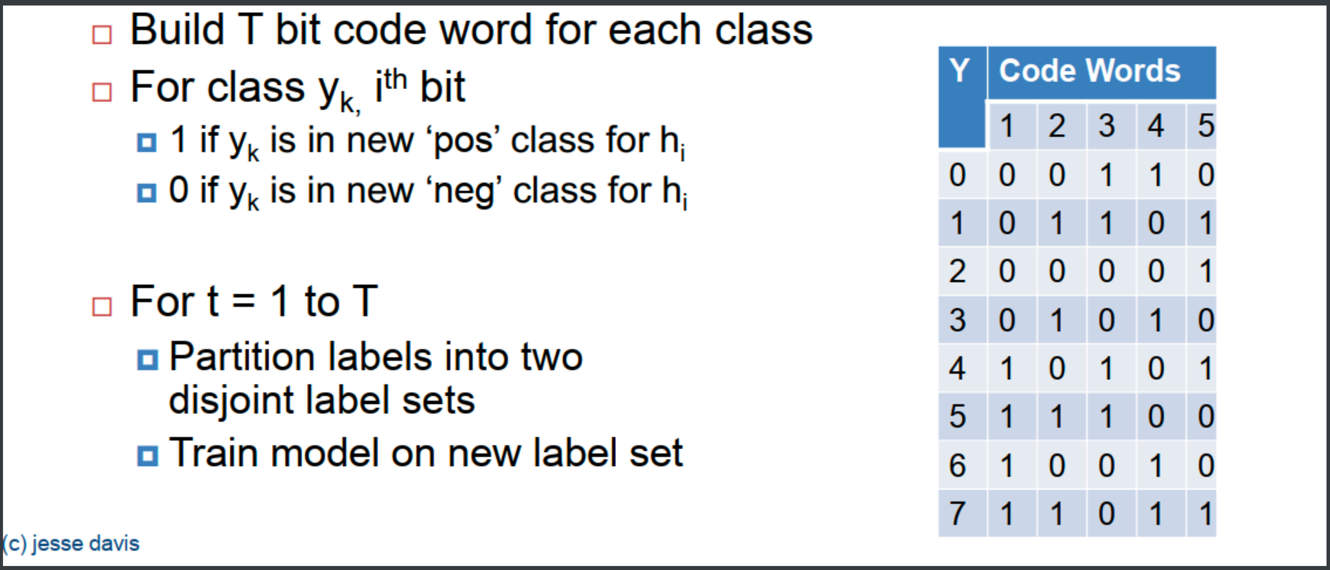
  one-vs-rest strategy: learn k binary classifiers

  solution 2:

  log k models

- **error-correcting codes**:

  construct new labels for each class



- □ Build T bit code word for each class
- □ For class $y_{k,}$ i^{th} bit
  - ◘ 1 if y_k is in new ‘pos’ class for h_i
  - ◘ 0 if y_k is in new ‘neg’ class for h_i
- □ For t = 1 to T
  - ◘ Partition labels into two disjoint label sets
  - ◘ Train model on new label set

Y	Code Words				
	1	2	3	4	5
0	0	0	1	1	0
1	0	1	1	0	1
2	0	0	0	0	1
3	0	1	0	1	0
4	1	0	1	0	1
5	1	1	1	0	0
6	1	0	0	1	0
7	1	1	0	1	1

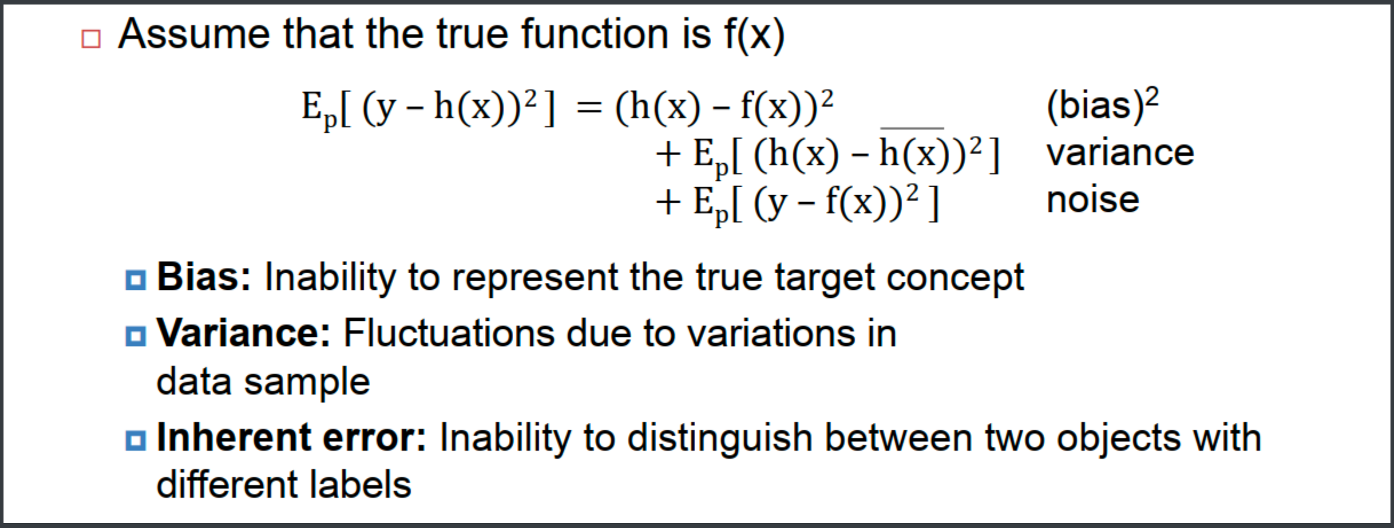
(c) jesse davis

for a test result (a T-bit vector), use hamming distance to find the closest class

why ensembles work

bias / variance explanation

the expected error:



□ Assume that the true function is f(x)

$$
\begin{aligned}
E_p[\,(y - h(x))^2\,] &= (h(x) - f(x))^2 && \text{(bias)}^2\\
&\quad + E_p[\,(h(x) - \overline{h(x)})^2\,] && \text{variance}\\
&\quad + E_p[\,(y - f(x))^2\,] && \text{noise}
\end{aligned}
$$

- **Bias:** Inability to represent the true target concept
- **Variance:** Fluctuations due to variations in data sample
- **Inherent error:** Inability to distinguish between two objects with different labels

- bias: underfitting the data

  incorrect assumptions,

  inability to represent certain decision boundaries (wrong model),

  "too global" classifiers (e.g. small decision tree)

- variance: overfitting the data

  "too local" classifiers (knn),

  decision made on small subsets of data,

  randomization in the learning algorithm,

  unstable learning algorithm

- for ensembles
  - bagging (sample)

    theoritically, reduces variance

    actually, resolves both

  - boosting (weight)

early iterations -> bias

later iterations -> variance

statistical explanation

- 贝叶斯后验估计
- bayes optimal classifier:

AdaBoost、Random Forest 等方法通过将多个弱分类器的预测结果进行加权投票，从而模拟贝叶斯分类器的后验概率推断过程。

- ensembles 近似 bayes optimal.

representational explanation

线性加权平均能够更好地逼近真实函数

computational explanation

avoid local minima -> repeat the search many times with random restarts

Applications