Adaboost in ASR

集成学习

论文分享

实验步骤

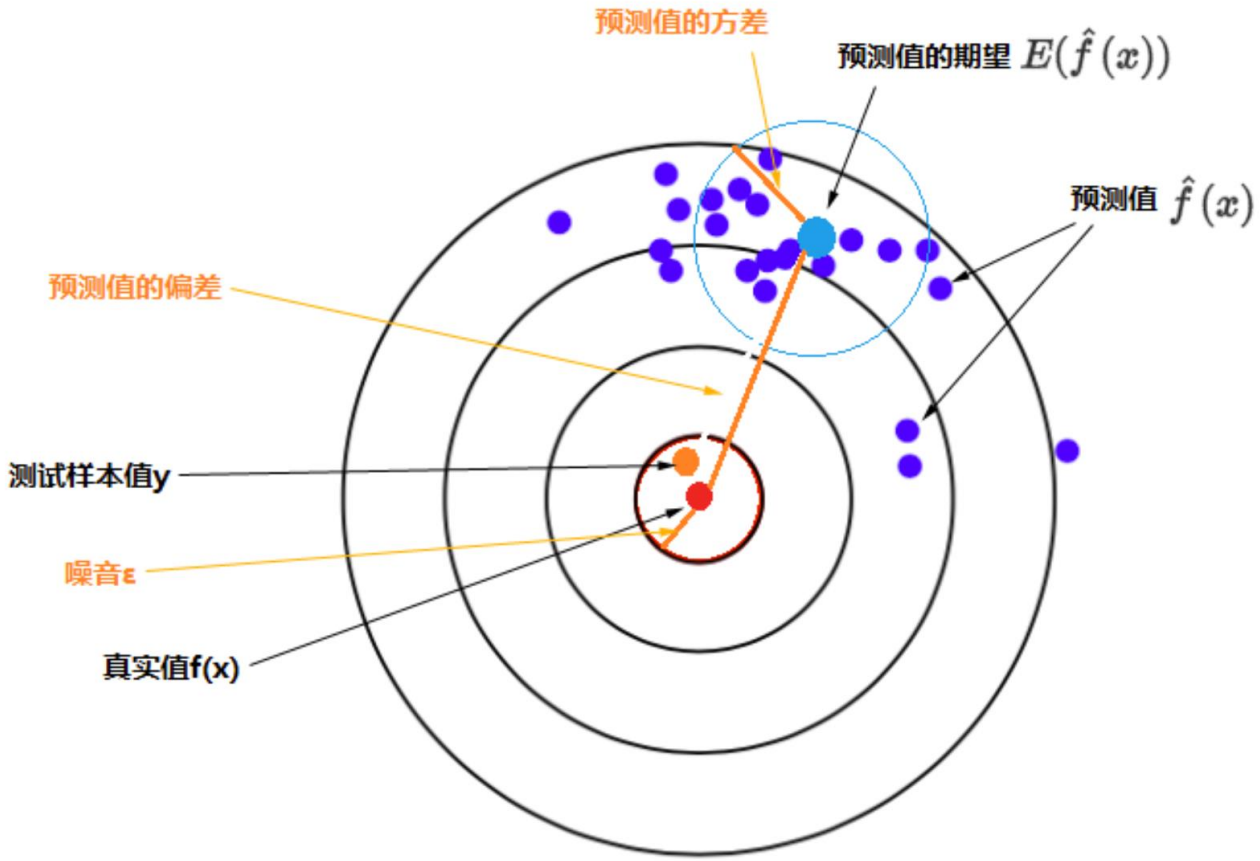
集成学习:

- 通过构建并结合多个学习器来完成学习任务。
- 一组基学习器$f_1(x), f_2(x), f_3(x), f_4(x)$ ……
- 用好的方法将分类器集合（aggregation）。

Bagging：个体学习器不存在强依赖关系，分类器容易overfitting。Eg: decision tree
Boosting：个体学习器存在强依赖关系，分类器较弱，ensemble却能得到较好的结果。

Bias & variance generalization

误差的期望值 = 噪音的方差 + 模型预测值的方差 + 预测值相对真实值的偏差的平方

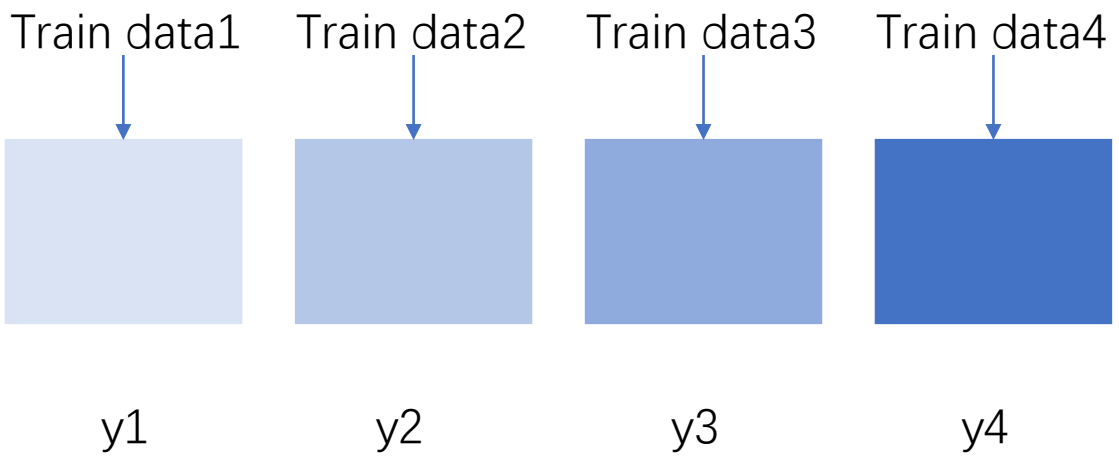


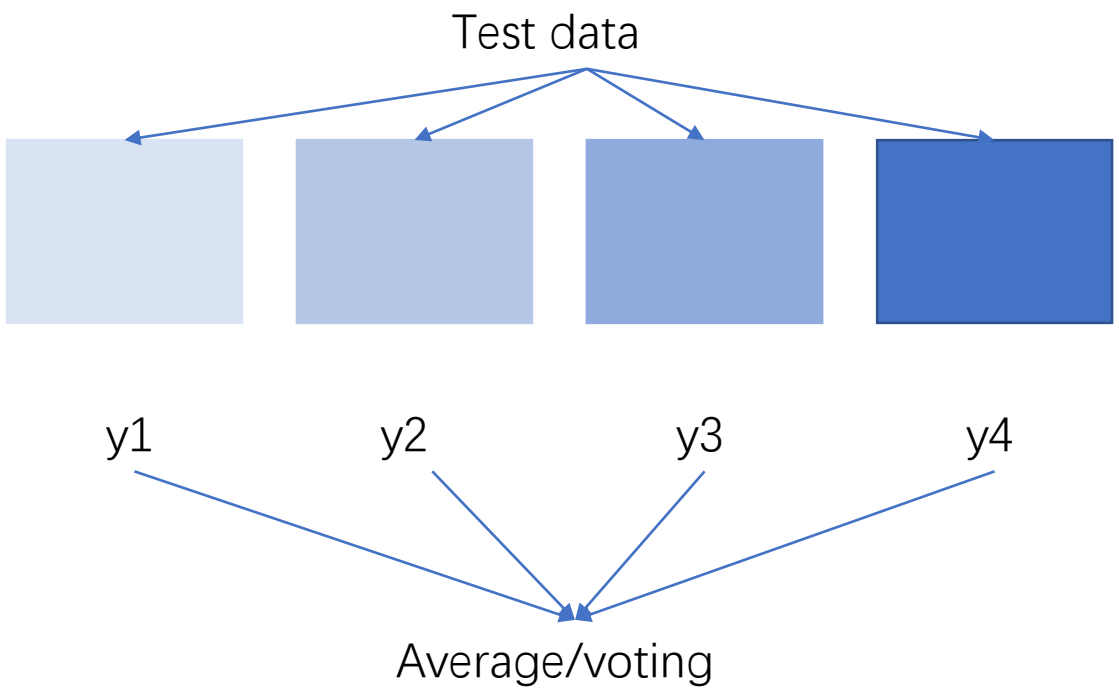
偏差：训练出的模型在训练集上的准确度

方差：方差越大的模型越容易过拟合

[模型的偏差和方差]

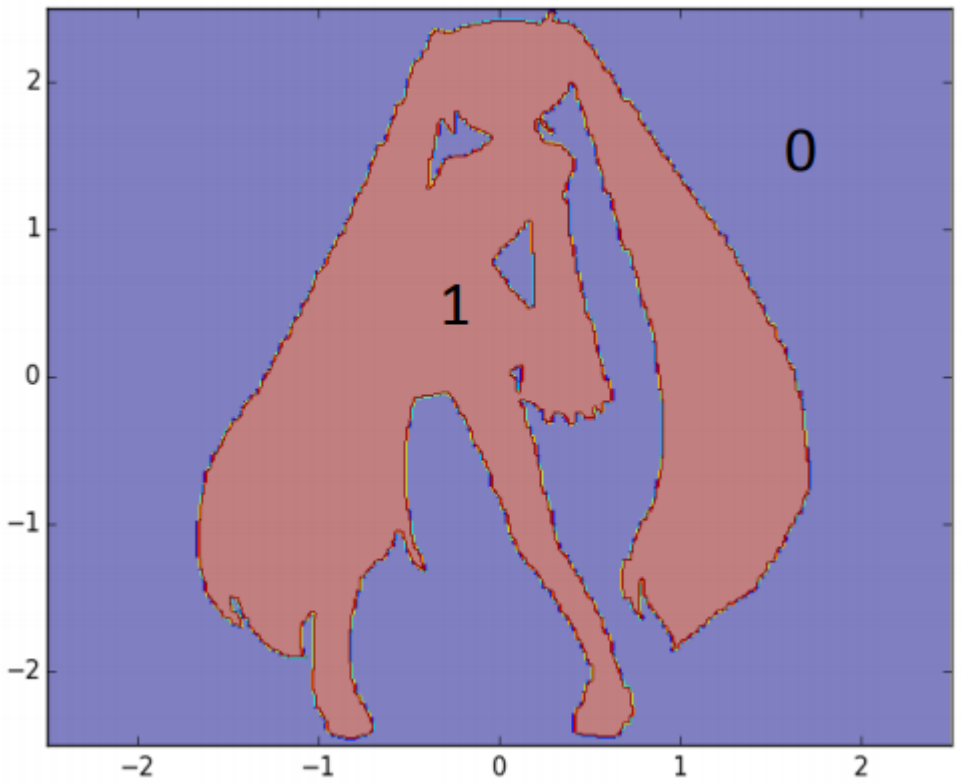
Bagging





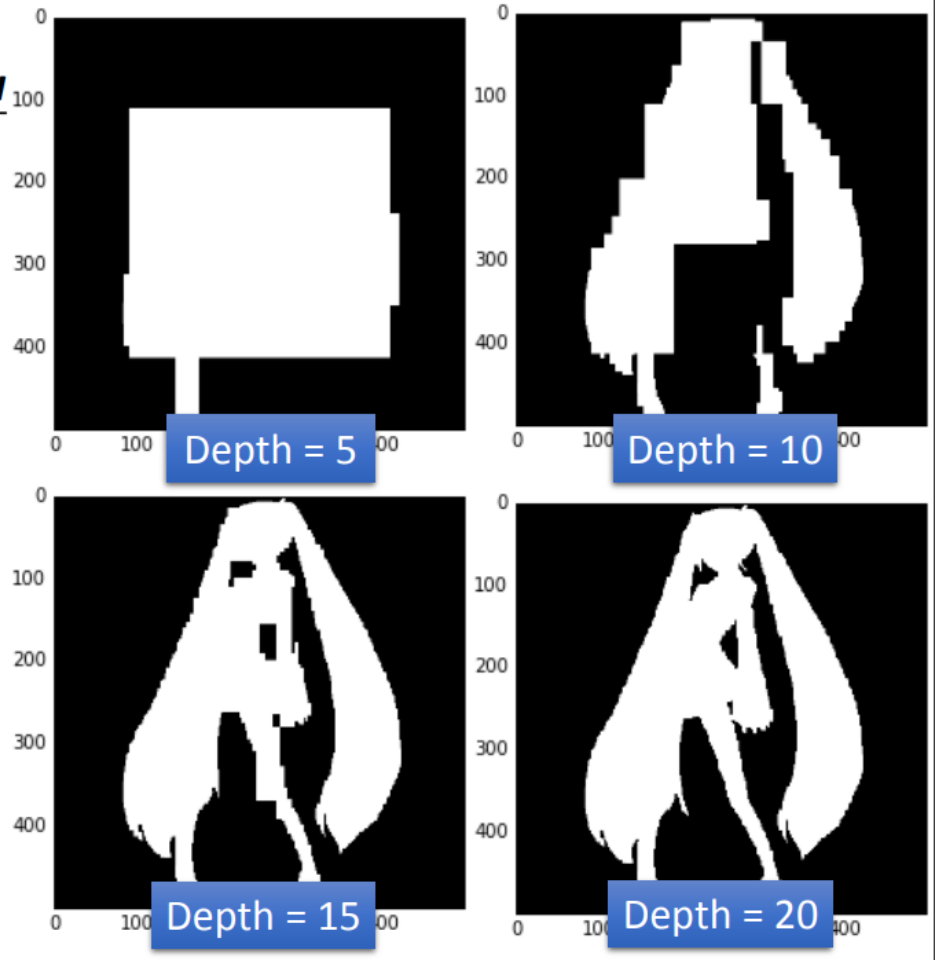
Bagging improves generalization performance due to a reduction in variance while maintaining or only slightly increasing bias.

Bagging



Experiment:
Function of Miku

Single
Decision
Tree



Bagging

- Model is complex → large variance, small bias

整体模型的期望近似于基模型的期望（偏差近似）

整体模型的方差小于基模型的方差

Boosting

- “关注错误分类样本”

强可学习

如果存在一个多项式的学习算法能够学习它，并且正确率很高，那么这个概念（类）是强可学习的

弱可学习

如果存在一个多项式的学习算法能够学习它，学习的正确率仅比随机猜测略好，那么这个概念（类）是弱可学习的

在概率近似正确（PAC）学习框架中，强可学习与弱可学习是等价的。

Adaboost

- How to obtain different classifiers?

If x^n is misclassified $u_{t+1}^n = u_t^n \times d_t$
If x^n is misclassified $u_{t+1}^n = u_t^n \div d_t$
那么f_1在re-weighting数据集上的准确率为50%

$$\varepsilon_t = \frac{\sum_n u_t^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_t}$$

$$Z_t = \sum_n u_t^n$$

$$d_t = \sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$u_{t+1}^n = u_t^n \times d_t = u_t^n \times \exp(\alpha_t)$$
$$u_{t+1}^n = u_t^n \times d_t = u_t^n \times \exp(-\alpha_t)$$

$$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

Adaboost

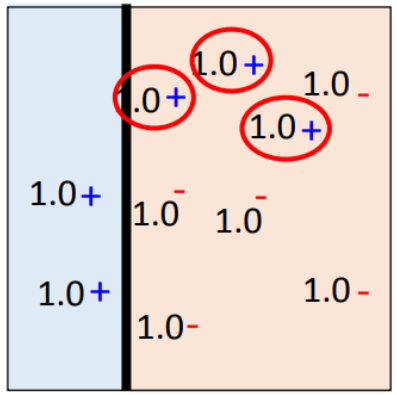
$$\boldsymbol{\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}}$$

- How to aggregate these classifiers?

1. $H(x) = sign(\boldsymbol{\alpha_1} f_1(x) + \boldsymbol{\alpha_2} f_2(x) + \boldsymbol{\alpha_3} f_3(x) \dots\dots)$

错误率低的classifier权重高，错误率高的classifier权重低

https://zhuanlan.zhihu.com/p/54483252



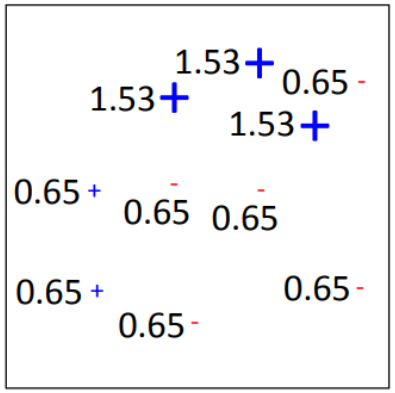
$f_1(x)$

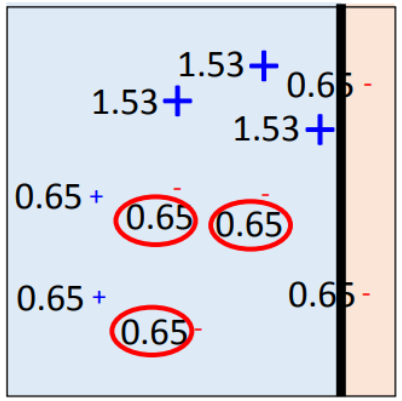
1

$\varepsilon_1 = 0.30$

$d_1 = 1.53$

$\alpha_1 = 0.42$





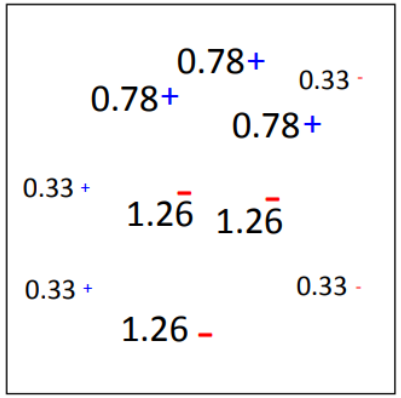
$f_2(x)$

2

$\varepsilon_2 = 0.21$

$d_2 = 1.94$

$\alpha_2 = 0.66$



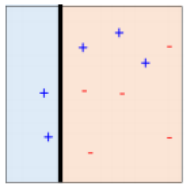
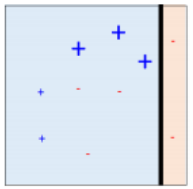
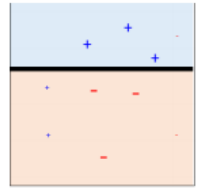
3

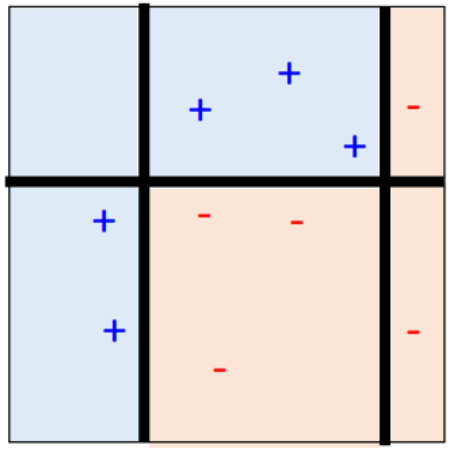
$f_3(x)$

$\varepsilon_3 = 0.13$

$d_3 = 2.59$

$\alpha_3 = 0.95$

$$sign(\ 0.42$$  $$+\ 0.66$$  $$+\ 0.95$$  $$)$$



$$d_t = \sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

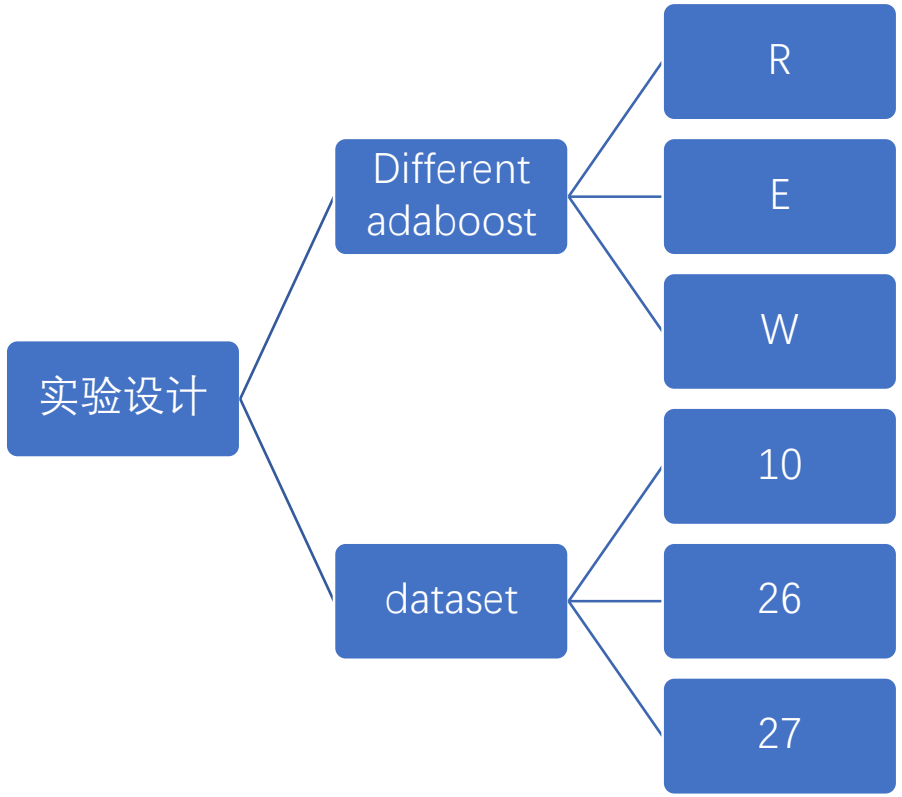
论文分析

《boosting neural network》 Schwenk, H., & Bengio, Y

- **实验目的**

Use adaboost on NN

- **实验设计**



- **算法改进**

Adaboost.M2

focuses not only on the examples that are hard to classify, but more specifically on improving the discrimination between the correct class and the incorrect class that competes with it

Pseudo-loss :

$\epsilon_t = \frac{1}{2}\sum_{(i,y)\in B} D_t(i,y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$

$\beta_t = \epsilon_t/(1-\epsilon_t)$

$D_{t+1}(i,y) = D_t(i,y)\beta_t^{\frac{1}{2}(1+h_t(x_i,y_i)-h_t(x_i,y))}$

$\alpha_t = log\frac{1}{\beta_t}$

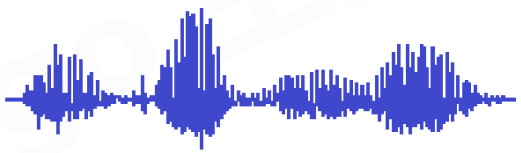
总结

1. Adaboost works well on NN
2. several hundred thousand classifiers → overfit
3. In the presence of significant amounts of noise degrades a lot
4. Neural network cannot overfit
5. Adaboost works as well for NN as decision tree
6. Adaboost is less useful for very big networks because it has low bias but high variance

概念:
Generation
Variance & bias
Margin of classification

Adaboost in ASR

训练



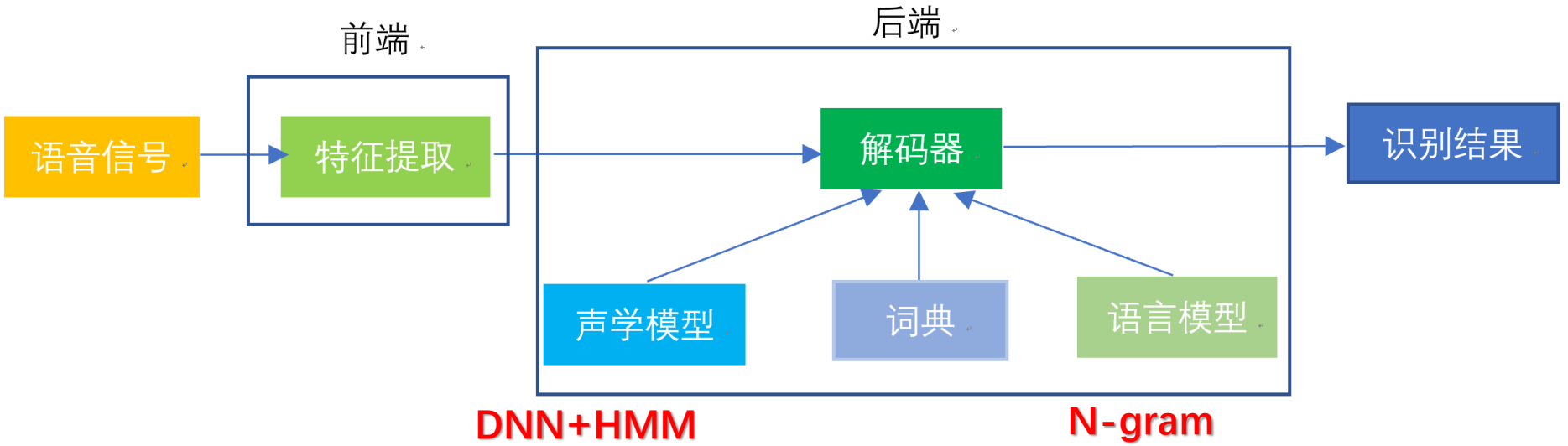
GMM+HMM

你好

n n i i i h h ao ao ao

NN

解码



Margin of classification

Note, however, there is no unique bias-variance decomposition for classification tasks.

Adaboost, on the other hand, constructs a composite classifier by sequentially training classifiers while putting more and more emphasis on certain patterns. For this, adaboost maintains a probability distribution $D_t(i)$ over the original training set. In each round t, the classifier is trained with respect to this distribution. Some learning algorithms do not allow training with respect to a weighted cost function. Examples with high probability would then occur more often than those with low probability, and some examples may not occur in the example at all, although their probability is not zero.

The probabilities are changed so that the error of the tth classfier using these new weights would be 0.5.

In general, neural network classifiers provide more information than just a class label. It can be shown that the network outputs approximate the a posterior probabilities of classes.

It can be shown that the error of the composite classifier on the training data decreases exponentially fast to zero as the number of combined classifiers is increased.

Many empirical evaluations of adaboost also provide an analysis of the so-called margin distribution. The margin is defined as the difference between the ensemble score of the correct class and the strongest ensemble score of a wrong class.

R: training the tth classifier with a fixed training set obtained by resampling with replacement once from the original training set. Sample N patterns from the original training set. Before starting training the tth network, we sample N patterns from the original training set, each time with a probability $P_t(i)$ of picking pattern i. training is performed for a fixed number of iterations always using this same resampled training set.

E: training the tth classifier using a different training set at each epoch, by resampling with replacement after each training epoch. After each epoch, a new training set is obtained by sampling from the original training set with probability $P_t(i)$. This is equivalent to sampling a new pattern from the original training set with probability $P_t(i)$ before each forward or backward pass through the neural network.

W: training the tth classifier by directly weighting the cost function (here the squared error) of the tth neural network.

E is a better approximation of the weighted cost function than R, in particular when many epochs are performed.