

PDEEC – Machine Learning 2018/19

Lecture - Model assessment, selection and Ensemble

Jaime S. Cardoso
`jaime.cardoso@inesctec.pt`

INESC TEC and Faculdade Engenharia, Universidade do Porto

Nov. 08, 2018

Main Sources of the Slides



Aarti Singh

Model Selection

<http://www.cs.cmu.edu/~epxing/Class/10701/Lecture/lecture8.pdf>



Aarti Singh

Boosting

<http://www.cs.cmu.edu/~epxing/Class/10701/Lecture/lecture19.pdf>



Trevor Hastie and Robert Tibshirani and Jerome Friedman

The elements of statistical learning (chap 7, 10),
Springer.

True vs. Empirical Risk

True Risk: Target performance measure

- ▶ Classification - Probability of misclassification $P(f(\mathbf{x}) \neq y)$
- ▶ Regression - Mean Squared Error $E[(f(\mathbf{x}) - y)^2]$

Also known as “Generalization Error” - performance on a random test point (X, Y)

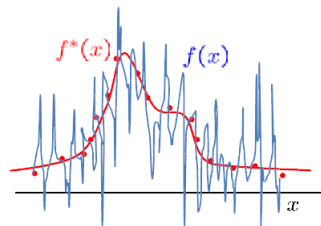
Empirical Risk: Performance on training data

- ▶ Classification - Proportion of misclassified examples $\frac{1}{N} \sum_{i=1}^N I_{f(\mathbf{x}_i) \neq y_i} P(f(\mathbf{x}) \neq y)$
- ▶ Regression - Average Squared Error $\frac{1}{N} \sum_{i=1}^N (f(\mathbf{x}_i) - y_i)^2$

Overfitting I

Is the following predictor a good one?

$$f(x) = \begin{cases} Y_i, & x = X_i \text{ for } i = 1, \dots, n \\ \text{any value,} & \text{otherwise} \end{cases}$$



What is its empirical risk? (performance on training data)

zero !

What about true risk?

> zero

Will predict very poorly on new random test point,

Large generalization error !

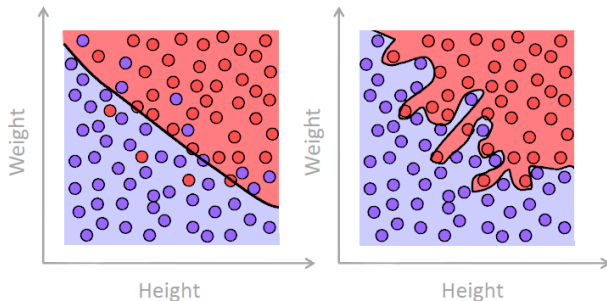
Overfitting II

If we allow very complicated predictors, we could overfit the training data.

Examples: Classification

Football player ?

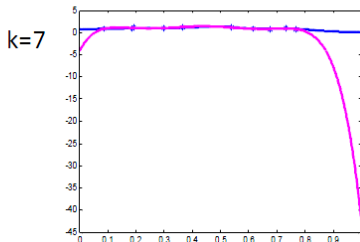
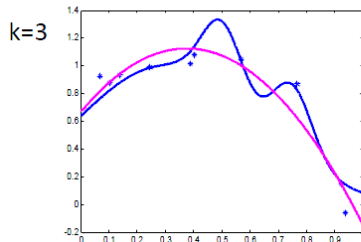
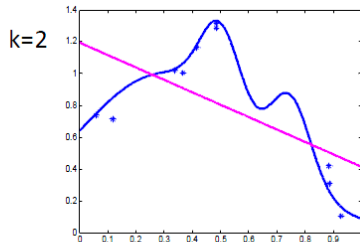
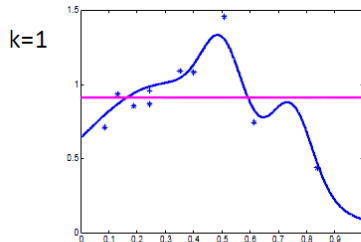
- No
- Yes



Overfitting III

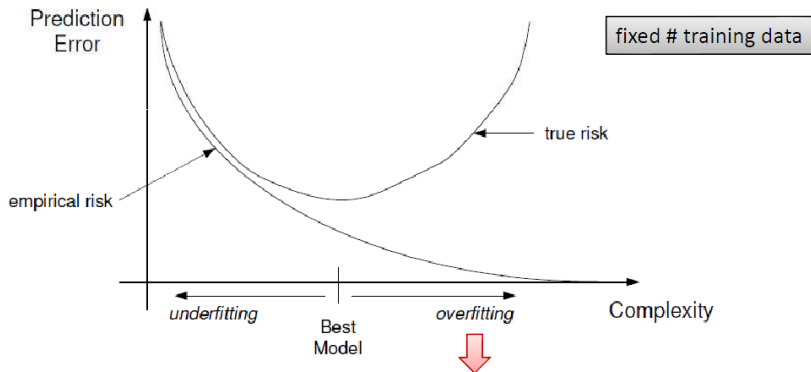
If we allow very complicated predictors, we could overfit the training data.

Examples: Regression



Effect of Model Complexity

If we allow very complicated predictors, we could overfit the training data.




Empirical risk is no longer a good indicator of true risk

Behavior of True Risk

Want predictor based on training data \hat{f}_n to be as good as optimal predictor f^*

Excess Risk $E[R(\hat{f}_n)] - R^*$

 wrt the distribution of training data

- Why is the risk of \hat{f}_n a random quantity?

$$\left. \begin{aligned} R(\hat{f}_n) &= P_{XY}(\hat{f}_n(X) \neq Y) \\ R(\hat{f}_n) &= \mathbb{E}_{XY}[(\hat{f}_n(X) - Y)^2] \end{aligned} \right\} \hat{f}_n \text{ depends on random training dataset}$$

Behavior of True Risk

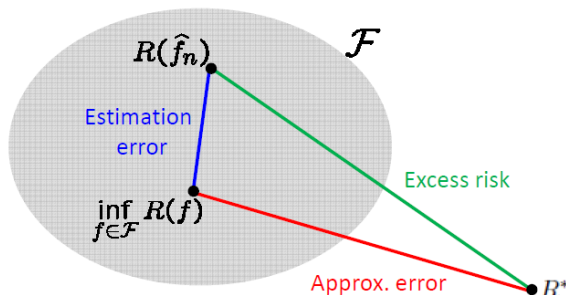
Want predictor based on training data \hat{f}_n to be as good as optimal predictor f^*

$$\text{Excess Risk } E[R(\hat{f}_n)] - R^* = \underbrace{\left(E[R(\hat{f}_n)] - \inf_{f \in \mathcal{F}} R(f) \right)}_{\text{estimation error}} + \underbrace{\left(\inf_{f \in \mathcal{F}} R(f) - R^* \right)}_{\text{approximation error}}$$

finite sample size
+ noise

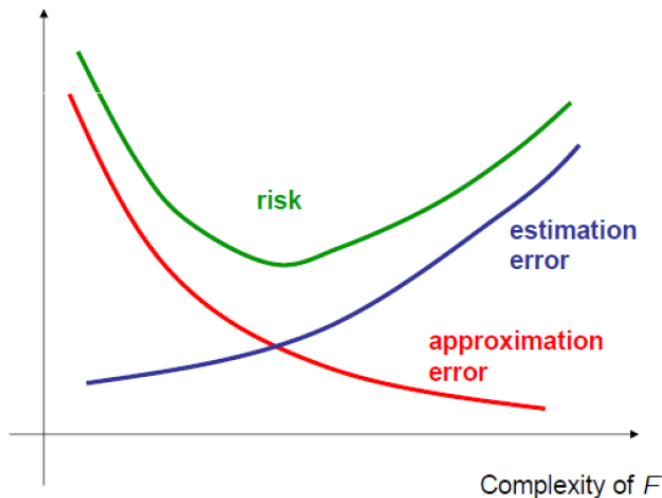
Due to randomness
of training data

Due to restriction
of model class



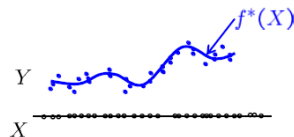
Behavior of True Risk

$$E[R(\hat{f}_n)] - R^* = \underbrace{\left(E[R(\hat{f}_n)] - \inf_{f \in \mathcal{F}} R(f)\right)}_{\text{estimation error}} + \underbrace{\left(\inf_{f \in \mathcal{F}} R(f) - R^*\right)}_{\text{approximation error}}$$



Behavior of True Risk

Regression: $Y = f^*(X) + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$



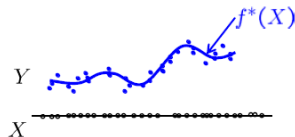
$$R^* = \mathbb{E}_{XY}[(f^*(X) - Y)^2] = \mathbb{E}[\epsilon^2] = \sigma^2$$

Notice: Optimal predictor does not have zero error

$$\begin{aligned}\mathbb{E}_D[R(\hat{f}_n)] &= \mathbb{E}_{X,Y,D}[(\hat{f}_n(X) - Y)^2] \\&= \mathbb{E}_{X,Y,D} [(\hat{f}_n(X) - \mathbb{E}_D[\hat{f}_n(X)] + \mathbb{E}_D[\hat{f}_n(X)] - Y)^2] \\&= \mathbb{E}_{X,Y,D} [(\hat{f}_n(X) - \mathbb{E}_D[\hat{f}_n(X)])^2 + (\mathbb{E}_D[\hat{f}_n(X)] - Y)^2 \\&\quad + 2(\hat{f}_n(X) - \mathbb{E}_D[\hat{f}_n(X)])(\mathbb{E}_D[\hat{f}_n(X)] - Y)] \\&= \mathbb{E}_{X,Y,D} [(\hat{f}_n(X) - \mathbb{E}_D[\hat{f}_n(X)])^2] + \mathbb{E}_{X,Y,D} [(\mathbb{E}_D[\hat{f}_n(X)] - Y)^2] \\&\quad + \underbrace{\mathbb{E}_{X,Y} [2(\mathbb{E}_D[\hat{f}_n(X)] - \mathbb{E}_D[\hat{f}_n(X)])(\mathbb{E}_D[\hat{f}_n(X)] - Y)]}_{0}\end{aligned}$$

Behavior of True Risk

Regression: $Y = f^*(X) + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$



$$R^* = \mathbb{E}_{XY}[(f^*(X) - Y)^2] = \mathbb{E}[\epsilon^2] = \sigma^2$$

Notice: Optimal predictor
does not have zero error

$$\begin{aligned} \mathbb{E}_D[R(\hat{f}_n)] &= \mathbb{E}_{X,Y,D}[(\hat{f}_n(X) - Y)^2] \\ &= \underbrace{\mathbb{E}_{X,Y,D}[(\hat{f}_n(X) - \mathbb{E}_D[\hat{f}_n(X)])^2]}_{\text{variance}} + \mathbb{E}_{X,Y,D}[(\mathbb{E}_D[\hat{f}_n(X)] - Y)^2] \end{aligned}$$

variance - how much does the predictor vary about its mean
for different training data points

Now, let's look at the second term:

$$\mathbb{E}_{X,Y,D}[(\mathbb{E}_D[\hat{f}_n(X)] - Y)^2] = \mathbb{E}_{X,Y}[(\mathbb{E}_D[\hat{f}_n(X)] - Y)^2]$$

Note: this term doesn't depend on D

Behavior of True Risk

$$\begin{aligned}\mathbb{E}_{X,Y} \left[(\mathbb{E}_D[\hat{f}_n(X)] - Y)^2 \right] &= \mathbb{E}_{X,Y} \left[(\mathbb{E}_D[\hat{f}_n(X)] - f^*(X) - \epsilon)^2 \right] \\&= \mathbb{E}_{X,Y} \left[(\mathbb{E}_D[\hat{f}_n(X)] - f^*(X))^2 + \epsilon^2 \right. \\&\quad \left. - 2\epsilon(\mathbb{E}_D[\hat{f}_n(X)] - f^*(X)) \right] \\&= \mathbb{E}_{X,Y} \left[(\mathbb{E}_D[\hat{f}_n(X)] - f^*(X))^2 \right] + \mathbb{E}_{X,Y} [\epsilon^2] \\&\quad - 2\mathbb{E}_{X,Y} [\epsilon(\mathbb{E}_D[\hat{f}_n(X)] - f^*(X))]\end{aligned}$$

0 since noise is independent
and zero mean

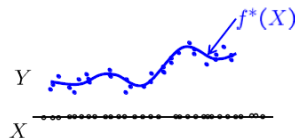
$$= \underbrace{\mathbb{E}_{X,Y} \left[(\mathbb{E}_D[\hat{f}_n(X)] - f^*(X))^2 \right]}_{\text{bias}^2} + \underbrace{\mathbb{E}_{X,Y} [\epsilon^2]}_{\text{noise variance}}$$

bias² - how much does the
predictor on average differ from the
optimal predictor

noise variance

Behavior of True Risk

Regression: $Y = f^*(X) + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$



$$R^* = \mathbb{E}_{XY}[(f^*(X) - Y)^2] = \mathbb{E}[\epsilon^2] = \sigma^2$$

Notice: Optimal predictor does not have zero error

$$\mathbb{E}_D[R(\hat{f}_n)] = \mathbb{E}_{X,Y,D}[(\hat{f}_n(X) - Y)^2]$$

$$= \underbrace{\mathbb{E}[(\hat{f}_n(X) - \mathbb{E}[\hat{f}_n(X)])^2]}_{\text{variance}} + \underbrace{\mathbb{E}[(\mathbb{E}[\hat{f}_n(X)] - f^*(X))^2]}_{\text{bias}^2} + \underbrace{\sigma^2}_{\text{Noise var}}$$

variance

bias²

Noise var

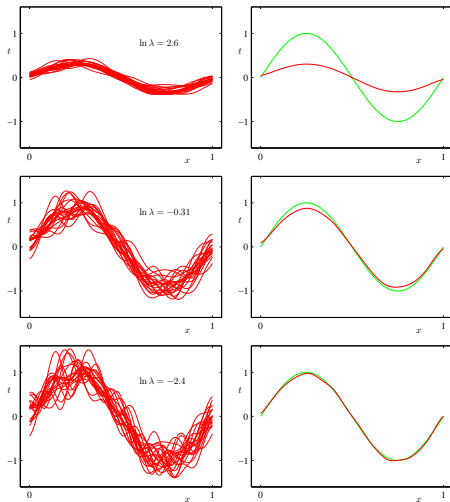
$$\text{Excess Risk} = \mathbb{E}_D[R(\hat{f}_n)] - R^* = \text{variance} + \text{bias}^2$$

Random component \equiv est err

\equiv approx err

Bias-variance in Regression

1. The bias term measures how far the mean prediction is from the true value.
2. The variance term measures how far each prediction is from the mean prediction.



Bias-variance Decomposition

1. Given an estimator $\hat{\theta}$ of a parameter θ underlying the distribution $P(X|\theta)$, the mean-squared error of $\hat{\theta}$ is defined as $MSE(\hat{\theta}) = E_{\theta}(\hat{\theta} - \theta)^2$
2. A very general way to understand any machine learning model is to decompose its error into a bias and a variance term.
3. Let us define the bias of an estimator as $bias(\hat{\theta}) = E_{\theta}(\hat{\theta}) - \theta$
4. An unbiased estimator has zero bias, of course, but often it turns out that the estimator with the lowest MSE may be biased!
5. The variance of an estimator is defined as $Var(\hat{\theta}) = E_{\theta}(\hat{\theta} - E_{\theta}(\hat{\theta}))^2$

Bias-variance Decomposition

1. Bias-Variance Theorem: The mean-squared error of any estimator $\hat{\theta}$ can be decomposed into its (squared) bias and variance components.

$$\begin{aligned}MSE(\hat{\theta}) &= \\&= E_{\theta}(\hat{\theta} - \theta)^2 \\&= E_{\theta}[\hat{\theta} - E_{\theta}(\hat{\theta}) + E_{\theta}(\hat{\theta}) - \theta]^2 \\&= E_{\theta} \left[(\hat{\theta} - E_{\theta}(\hat{\theta}))^2 + (E_{\theta}(\hat{\theta}) - \theta)^2 - 2(\hat{\theta} - E_{\theta}(\hat{\theta}))(E_{\theta}(\hat{\theta}) - \theta) \right] \\&= E_{\theta}(\hat{\theta} - E_{\theta}(\hat{\theta}))^2 + E_{\theta}(E_{\theta}(\hat{\theta}) - \theta)^2 - 2E_{\theta}(\hat{\theta} - E_{\theta}(\hat{\theta}))(E_{\theta}(\hat{\theta}) - \theta) \\&= E_{\theta}(\hat{\theta} - E_{\theta}(\hat{\theta}))^2 + E_{\theta}(E_{\theta}(\hat{\theta}) - \theta)^2 \\&= E_{\theta}(\hat{\theta} - E_{\theta}(\hat{\theta}))^2 + (E_{\theta}(\hat{\theta}) - \theta)^2 \\&= Var(\hat{\theta}) + Bias^2(\hat{\theta})\end{aligned}$$

Bias-variance Decomposition

Bias-Variance Theorem.

$y(x)$: true value to be predicted

$E(y|x)$

$f(x)$: estimate of $y(x)$

$$\begin{aligned} E\{f(x) - y(x)\}^2 &= E\{f(x) - E(y|x) + E(y|x) - y(x)\}^2 \\ &= E\{f(x) - E(y|x)\}^2 + E\{E(y|x) - y(x)\}^2 - 2E\{(f(x) - E(y|x))(E(y|x) - y(x))\} \\ &= E\{f(x) - E(y|x)\}^2 + E\{E(y|x) - y(x)\}^2 \\ &= E\{f(x) - E(y|x)\}^2 + \text{noise} \end{aligned}$$

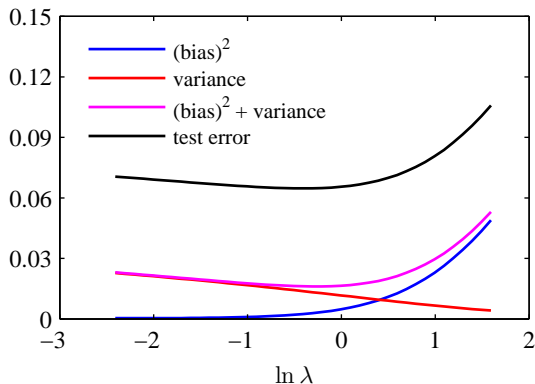
$$\begin{aligned} &E\{f(x, D) - E(y|x)\}^2 \\ &= E\{f(x, D) - E\{f(x, D)\} + E\{f(x, D)\} - E(y|x)\}^2 \\ &= E\{f(x, D) - E\{f(x, D)\}\}^2 + E\{E\{f(x, D)\} - E(y|x)\}^2 \\ &\quad - 2E\{(f(x, D) - E\{f(x, D)\})(E\{f(x, D)\} - E(y|x))\} \\ &= E\{f(x, D) - E\{f(x, D)\}\}^2 + E\{E\{f(x, D)\} - E(y|x)\}^2 \\ &= \text{Var}(f(x, D)) + \text{Bias}^2(f(x, D)) \end{aligned}$$

Expected loss = (bias)² + variance + noise

Bias-variance Tradeoff

Example

1. The bias term measures how far the mean prediction is from the true value.
2. The variance term measures how far each prediction is from the mean prediction.



Examples of Model Spaces

Model Spaces with increasing complexity:

- ▶ Nearest-Neighbor classifiers with varying neighborhood sizes $k = 1, 2, 3, \dots$
Small neighborhood \Rightarrow Higher complexity
- ▶ Decision Trees with depth k or with k leaves
Higher depth/ More # leaves \Rightarrow Higher complexity
- ▶ Regression with polynomials of order $k = 0, 1, 2, \dots$
Higher degree \Rightarrow Higher complexity
- ▶ Kernel Regression with bandwidth h
Small bandwidth \Rightarrow Higher complexity

How can we select the right complexity model ?

Model selection and assessment

Data-rich situation

1. selection: estimating the performance of different models in order to choose the (approximate) best one
2. assessment: having chosen a final model, estimating its prediction error (generalization error) on new data
3. Data-rich situation:
 - ▶ randomly divide the dataset into three parts: a training set, validation set and a test set.
 - ▶ the training set is used to fit the models; the validation set is used to estimate prediction error for model selection; the test set is used for assessment of the generalization error of the final chosen model.
 - ▶ training error: the average loss over the training sample
$$\overline{err} = \frac{1}{N} \sum_{n=1}^N L(y_i, f(x_i))$$
 - ▶ test error or generalization error: the expected prediction error over an independent test sample $Err = E[L(Y, f(X))]$
 - ▶ typically the training error \overline{err} is less than the true error Err because the same data is being used to fit the method and assess its error.

Model selection and assessment

Data-rich situation

Hold-out method

We would like to pick the model that has smallest generalization error.

Can judge generalization error by using an independent sample of data.

Hold - out procedure:

n data points available $D \equiv \{X_i, Y_i\}_{i=1}^n$

1) Split into two sets: Training dataset Validation dataset **NOT test Data !!**
 $D_T = \{X_i, Y_i\}_{i=1}^m$ $D_V = \{X_i, Y_i\}_{i=m+1}^n$

2) Use D_T for training a predictor from each model class:

$$\hat{f}_\lambda = \arg \min_{f \in \mathcal{F}_\lambda} \hat{R}_T(f)$$

└─ Evaluated on training dataset D_T


Model selection and assessment

Data-rich situation

Hold-out method

3) Use D_V to select the model class which has smallest empirical error on D_V

$$\hat{\lambda} = \arg \min_{\lambda \in \Lambda} \hat{R}_V(\hat{f}_\lambda)$$

 Evaluated on validation dataset D_V

4) Hold-out predictor

$$\hat{f} = \hat{f}_{\hat{\lambda}}$$

Intuition: Small error on one set of data will not imply small error on a randomly sub-sampled second set of data

Ensures method is “stable”

Model selection and assessment

Data-rich situation

Hold-out method

Drawbacks:

- May not have enough data to afford setting one subset aside for getting a sense of generalization abilities
- Validation error may be misleading (bad estimate of generalization error) if we get an “unfortunate” split

Limitations of hold-out can be overcome by a family of random sub-sampling methods at the expense of more computation.

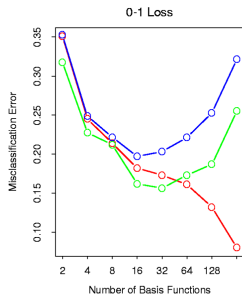
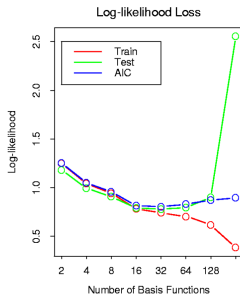
Estimates of Prediction

In-Sample Estimates

An obvious way to estimate the prediction error is to estimate the optimism of the training error and then add it to the training error

$$\hat{err}: \hat{Err} = \overline{err} + optimism$$

- ▶ Akaike information criterion: $AIC = \overline{err} + 2\frac{d}{N}\hat{\sigma}_\epsilon^2$
 - ▶ N: number of samples
 - ▶ $\hat{\sigma}_\epsilon^2$: estimate of the noise variance, obtained from the mean-squared error of a low-bias model.
 - ▶ d: effective number of parameters. The concept of number of parameters can be generalized to models with regularization.



Estimates of Prediction

In-Sample Estimates

- ▶ Bayesian information criterion:

$$BIC = \frac{N}{\hat{\sigma}_\epsilon^2} [\overline{err} + (\log N) \frac{d}{N} \hat{\sigma}_\epsilon^2]$$

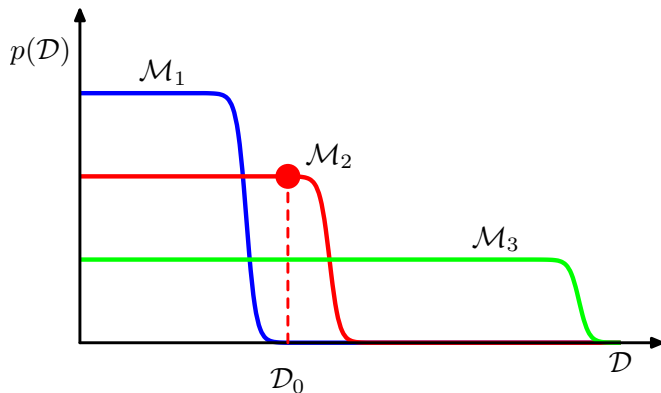
- ▶ assuming $N > e^2 \approx 7.4$, BIC tends to penalize complex models more heavily than AIC, giving preference to simpler models in selection.
- ▶ like AIC, BIC is applicable in settings where the fitting is carried out by maximization of a log-likelihood
- ▶ arises in a Bayesian approach to model selection, under some approximating conditions
- ▶ $\frac{Pr(\mathcal{M}_m|D)}{Pr(\mathcal{M}_\ell|D)} = \frac{Pr(\mathcal{M}_m)}{Pr(\mathcal{M}_\ell)} \cdot \frac{Pr(D|\mathcal{M}_m)}{Pr(D|\mathcal{M}_\ell)}$
- ▶ assuming some simplifications to $Pr(D|\mathcal{M}_m)$, one obtain the generic form of BIC: $BIC = -2loglik + (logN).d$

Estimates of Prediction

In-Sample Estimates

Bayesian Model Comparison

- ▶ $p(\mathcal{M}_i|D) \propto p(\mathcal{M}_i)p(D|\mathcal{M}_i)$
- ▶ $p(D|\mathcal{M}_i) = \int p(D|w, \mathcal{M}_i)p(w|\mathcal{M}_i)dw$



Estimates of Prediction

In-Sample Estimates

Bayesian Model Comparison

- ▶ the Bayesian framework avoids the problem of over-fitting and allows models to be compared on the basis of the training data alone.
- ▶ a Bayesian approach, like any approach to pattern recognition, needs to make assumptions about the form of the model, and if these are invalid then the results can be misleading.
- ▶ In particular, the model evidence can be sensitive to many aspects of the prior, such as the behaviour in the tails.
- ▶ In a practical application, therefore, it will be wise to keep aside an independent test set of data on which to evaluate the overall performance of the final system.

Estimates of Prediction

In-Sample Estimates

Minimum description length

- ▶ the minimum description length approach gives a selection criterion formally identical to the BIC but motivated from a optimal coding viewpoint
- ▶ assume we have a model with parameters θ and data $Z = (X, y)$ consisting of both the inputs and outputs. Let the probability of the outputs under the model be $Pr(y|\theta, X)$, assume the receiver knows all the inputs and we wish to transmit the outputs. Then the message length required to transmit the outputs is $\text{length} = -\log Pr(y|\theta, X) - \log Pr(\theta)$
- ▶ the preceding view of MLD for model selection says that we should choose the model with highest posterior probability. However, many bayesians would instead do inference by sampling from the posterior probability.

Estimates of Prediction

Vapnik-Chernovenkis Dimension

- ▶ a difficulty in using estimates of in-sample error is the need to specify the number of parameters (or the complexity) used in the fit.
- ▶ The effective number of parameters is not fully general
- ▶ the Vapnik-Chernovenkis (VC) theory provides such a general measure of complexity.

Vapnik-Chervonenkis Dimension

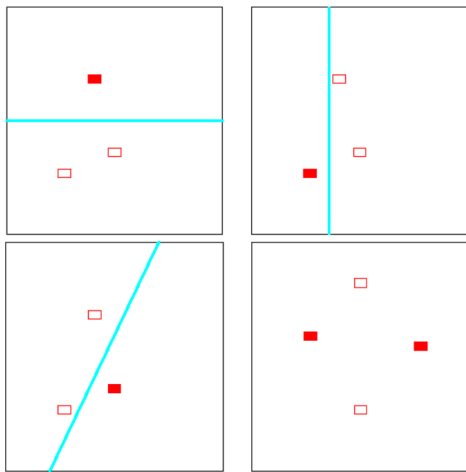
Shattering

- ▶ **Idea:** measure complexity by the effective number of distinct functions that can be defined.
- ▶ A hypothesis class is more complex if it can represent lots of possible classification decisions.
- ▶ **Def:** A set of points is **shattered** by \mathcal{H} (the hypothesis space) if for all possible binary labelings of the points, there exists $h \in \mathcal{H}$ that can represent this decision.
- ▶ We will measure complexity by the number of points that can be shattered by \mathcal{H} .

Vapnik-Chervonenkis Dimension

Shattering

Def: A set of points is **shattered** by \mathcal{H} (the hypothesis space) if for all possible binary labelings of the points, there exists $h \in \mathcal{H}$ that can represent this decision.



Vapnik-Chernovenkis Dimension

The Game

VC dimension is a shattering game between us and an adversary.

To show that \mathcal{H} has VC dimension $\geq d$:

- ▶ I choose d points positioned however I want
- ▶ Adversary chooses a labeling of these d points
- ▶ I choose $h \in \mathcal{H}$ that labels the points properly

The VC dimension of \mathcal{H} is the maximum d I can choose so that I can always succeed in the game.

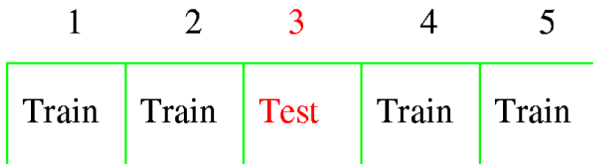
- ▶ a linear indicator function in p dimensions has VC equal to $(p+1)$, which is also the number of free parameters.
- ▶ the VC dimension can be extended to real-valued functions
- ▶ VC dimension is a measure of complexity of infinite hypothesis classes

Estimates of Prediction

Extra-Sample Estimates

Cross validation

- ▶ ideally, if we had enough data, we would set aside a validation set and use it to assess the performance of our prediction model.
- ▶ K-fold cross-validation uses part of the available data to fit the model and a different part to test it.
- ▶ we split the data into K roughly equal-size parts. For $k = 1, 2, \dots, K$, we fit the model to the data with the k th part removed and calculate the prediction error of the fitted model when predicting for the k th part of the data



Estimates of Prediction

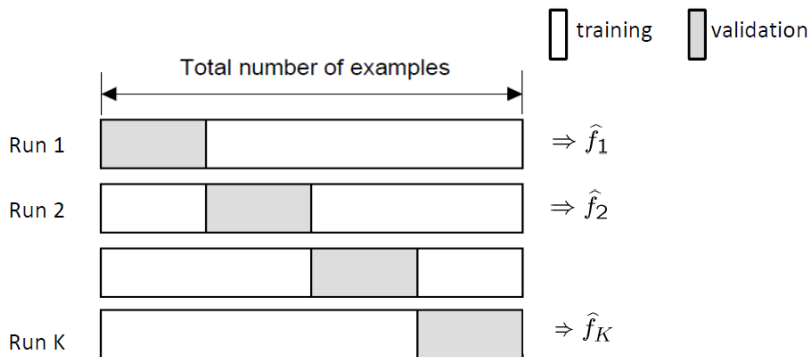
Extra-Sample Estimates

K-fold cross-validation

Create K-fold partition of the dataset.

Form K hold-out predictors, each time using one partition as validation and rest K-1 as training datasets.

Final predictor is average/majority vote over the K hold-out estimates.



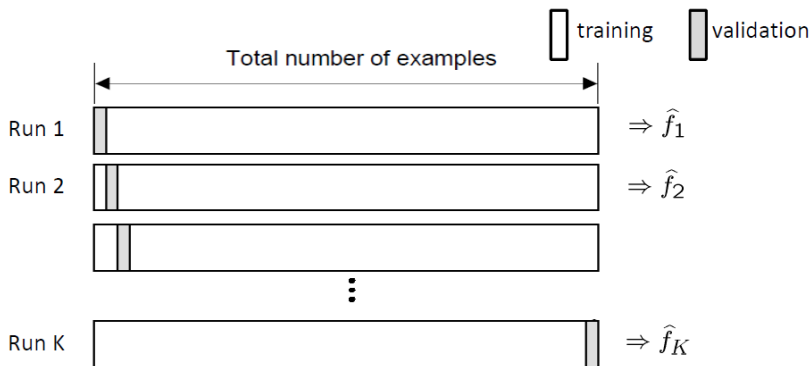
Estimates of Prediction

Extra-Sample Estimates

Leave-one-out (LOO) cross-validation

Special case of K-fold with $K=n$ partitions

Equivalently, train on $n-1$ samples and validate on only one sample per run for n runs



Estimates of Prediction

Extra-Sample Estimates

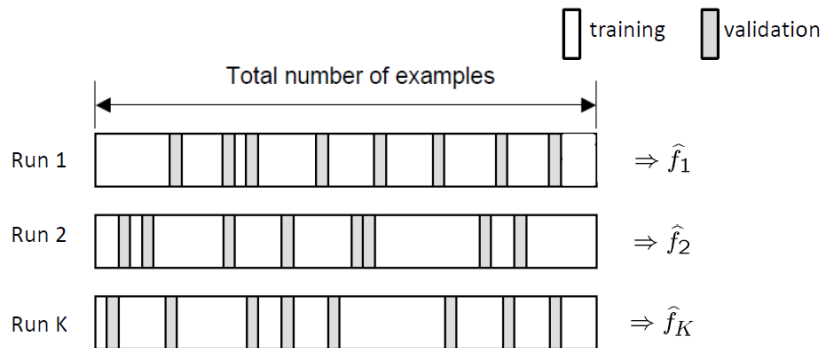
Cross-validation: Random subsampling

Randomly subsample a fixed fraction αn ($0 < \alpha < 1$) of the dataset for validation.

Form hold-out predictor with remaining data as training data.

Repeat K times

Final predictor is average/majority vote over the K hold-out estimates.



Estimates of Prediction

Extra-Sample Estimates

Estimating generalization error

$$\text{Generalization error } \mathbb{E}_D[R(\hat{f}_n)]$$

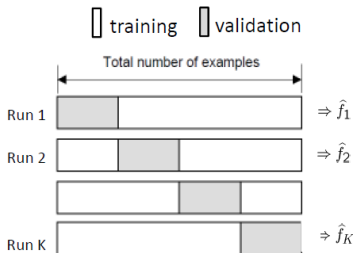
Hold-out \equiv 1-fold: Error estimate = $\hat{R}_V(\hat{f}_T)$

K-fold/LOO/random sub-sampling: Error estimate = $\frac{1}{K} \sum_{k=1}^K \hat{R}_{V_k}(\hat{f}_{T_k})$

We want to estimate the error of a predictor based on n data points.

If K is large (close to n), bias of error estimate is small since each training set has close to n data points.

However, variance of error estimate is high since each validation set has fewer data points and \hat{R}_{V_k} might deviate a lot from the mean.

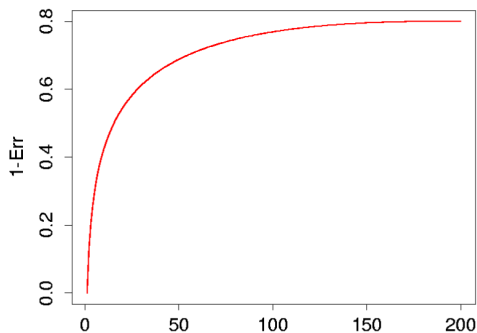


Estimates of Prediction

Extra-Sample Estimates

Cross validation - What value should we choose for K ?

- ▶ with $k=N$, CV is approximately unbiased for the true prediction error, but can have high variance because the N “training sets” are so similar to one another. The computational burden is also considerable.
- ▶ on the other hand, with $K=5$ CV has low variance but the bias could be a problem.



Estimates of Prediction

Extra-Sample Estimates

Practical Issues in Cross-validation

How to decide the values for K and α ?

- Large K
 - + The bias of the error estimate will be small
 - The variance of the error estimate will be large
 - The computational time will be very large as well (many experiments)
- Small K
 - + The # experiments and, therefore, computation time are reduced
 - + The variance of the error estimate will be small
 - The bias of the error estimate will be large

In practice, the choice of the number of folds depends on the size of the dataset:

For large datasets, even 3-Fold Cross Validation will be quite accurate
For very sparse datasets, we may have to use leave-one-out in order to train on as many examples as possible

- A common choice is $K=10$ and $\alpha = 0.1$

Evaluation Measures

- Are we doing well? Is system A better than B?
- A measure of how (in)accurate a system is on a task
 - in many cases Error (Accuracy / PC) is not the best measure
 - using the appropriate measure will help select best algorithm
- Classification
 - how often we classify something right / wrong
- Regression
 - how close are we to what we're trying to predict
- Unsupervised
 - how well do we describe our data
 - in general – really hard

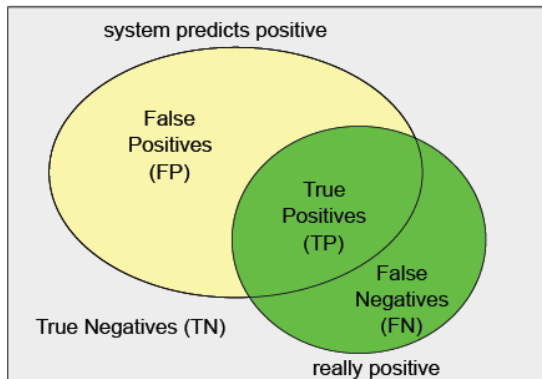
Classification measures: basics

Really positive?	Predict positive?	
	Yes	No
Yes	TP	FN
No	FP	TN

Confusion matrix for
two-class classification

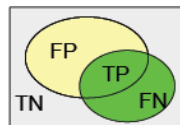
Want: large diagonal,
small FP, FN

all testing instances



Classification Error

- Classification error = $\frac{\text{errors}}{\text{total}} = \frac{FP + FN}{TP + TN + FP + FN}$
- Accuracy = $(1 - \text{error}) = \frac{\text{correct}}{\text{total}} = \frac{TP + TN}{TP + TN + FP + FN}$
- Basic measure of “goodness” of a classifier
- Problem: cannot handle unbalanced classes
 - ex1: predict whether an earthquake is about to happen
 - happen very rarely, very good accuracy if always predict “No”
 - solution: make FNs much more “costly” than FPs
 - ex2: web search: decide if a webpage is relevant to user
 - 99.9999% of pages not relevant to any query → retrieve nothing
 - solution: use measures that don’t involve TN (recall / precision)



Misses and False Alarms

- False Alarm rate = False Positive rate = $FP / (FP+TN)$
 - % of negatives we misclassified as positive
- Miss rate = False Negative rate = $FN / (TP+FN)$
 - % of positives we misclassified as negative
- Recall = True Positive rate = $TP / (TP+FN)$
 - % of positives we classified correctly (1 – Miss rate)
- Precision = $TP / (TP + FP)$
 - % positive out of what we predicted was positive
- Meaningless to report just one of these
 - trivial to get 100% recall or 0% false alarm
 - typical: recall/precision or Miss / FA rate or TP/FP rate

Evaluation (recap)

- Predicting class C (e.g. spam)

		Predicted C?	
Really C?		Yes	No
	Yes	TP	FN
	No	FP	TN

- classifier can make two types of mistakes:

- FP: false positives – non-spam emails mistakenly classified as spam
- FN: false negatives – spam emails mistakenly classified as non-spam
- TP/TN: true positives/negatives – correctly classified spam/non-spam

- common error/accuracy measures:

- Classification Error: $\frac{\text{errors}}{\text{total}} = \frac{FP + FN}{TP + TN + FP + FN}$

- Accuracy = 1-Error: $\frac{\text{correct}}{\text{total}} = \frac{TP + TN}{TP + TN + FP + FN}$

} meaningless
if classes
imbalanced

- False Alarm = False Positive rate = $FP / (FP + TN)$

- Miss = False Negative rate = $FN / (TP + FN)$

- Recall = True Positive rate = $TP / (TP + FN)$

- Precision = $TP / (TP + FP)$

} always report
in pairs, e.g.:
Miss / FA or
Recall / Prec.

Thresholds in Classification

Two systems have the following performance:

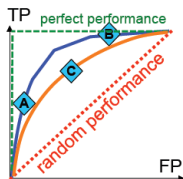
- A: True Positive = 50%, False Positive = 20%
- B: True Positive = 100%, False Positive = 60%

Which is better? (assume no-apriori utility)

- very misleading question
- A and B could be the same exact system
 - operating at different thresholds

ROC curves

- Many algorithms compute “confidence” $f(x)$
 - threshold to get decision: spam if $f(x) > t$, non-spam if $f(x) \leq t$
 - Naïve Bayes: $P(\text{spam}|x) > 0.5$, Linear/Logistic/SVM: $w^T x > 0$, Decision Tree: $p_L/p_R > 1$
 - threshold t determines error rates
 - False Positive rate = $P(f(x) > t | \text{ham})$, True Positive rate = $P(f(x) > t | \text{spam})$
- Receiver Operating Characteristic (ROC):
 - plot TPR vs. FPR as t varies from $-\infty$ to ∞

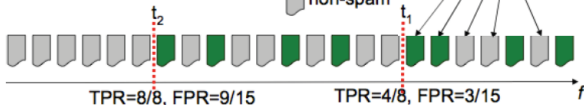


shows performance of system across all possible thresholds

AUC = area under ROC curve
popular alternative to Accuracy

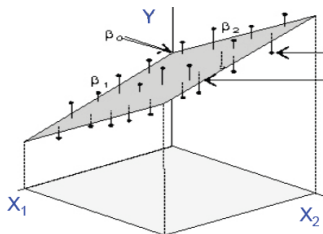
spam
non-spam

Really spam?	Predicted spam?	
	Yes	No
Yes	TP	FN
No	FP	TN



Evaluating regression

- Classification:
 - count how often we are wrong
- Regression:
 - predict numbers y_i from inputs x_i
 - always wrong, but by how much?
 - distance between predicted & true values
 - (root) mean squared error:
 - popular, well-understood, nicely differentiable
 - sensitive to single large errors (outliers)
 - mean absolute error:
 - less sensitive to outliers
 - correlation coefficient
 - insensitive to mean & scale



$$\sqrt{\frac{1}{n} \sum_{i=1}^n \underbrace{(f(x_i))}_{\text{predicted}} - \underbrace{(y_i)}_{\text{true}})^2}$$

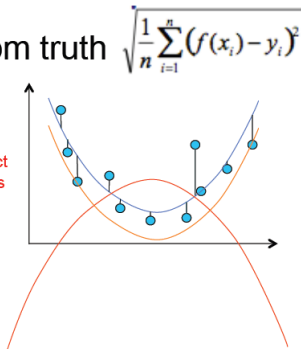
testing set

$$\frac{n \sum_{i=1}^n (f(x_i) - \mu_f)(y_i - \mu_y)}{\sqrt{\sum_{i=1}^n (f(x_i) - \mu_f)^2 \cdot \sum_{i=1}^n (y_i - \mu_y)^2}}$$

Mean Squared Error

- Average (squared) deviation from truth $\sqrt{\frac{1}{n} \sum_{i=1}^n (f(x_i) - y_i)^2}$
- Very sensitive to outliers
 - 99 exact, 1 off by \$10
 - all 100 wrong by \$1

} same MSE \Rightarrow large effect on models
- Sensitive to mean / scale
 - $\mu_y = 1/n \sum_i y_i$... good baseline
- Relative squared error (Weka)



$$\sqrt{\frac{\sum_{i=1}^n (f(x_i) - y_i)^2}{n}} \quad \left. \vphantom{\frac{\sum_{i=1}^n (f(x_i) - y_i)^2}{n}} \right\} \text{MSE of predictor}$$
$$\sqrt{\frac{\sum_{i=1}^n (\mu_y - y_i)^2}{n}} \quad \left. \vphantom{\frac{\sum_{i=1}^n (\mu_y - y_i)^2}{n}} \right\} \text{MSE when using the mean as a predictor}$$

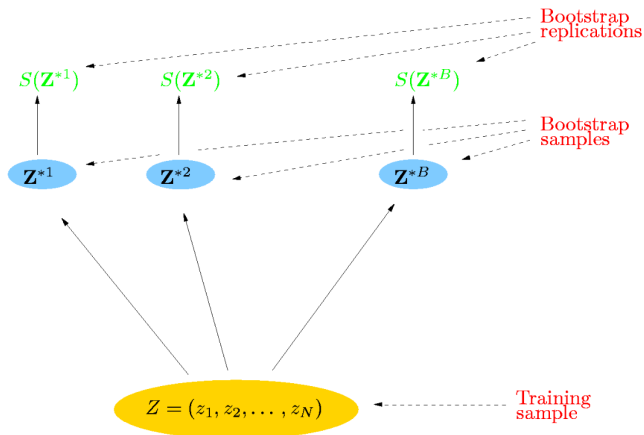
Model Assessment

Bootstrap methods

- ▶ Bootstrap is a general tool to assessing statistical accuracy, falling within a broader class of resampling methods.
- ▶ Bootstrapping is the practice of estimating properties of an estimator (such as its variance) by measuring those properties when sampling from an approximating distribution. One standard choice for an approximating distribution is the empirical distribution of the observed data. In the case where a set of observations can be assumed to be from an independent and identically distributed population, this can be implemented by constructing a number of resamples of the observed dataset (and of equal size to the observed dataset), each of which is obtained by random sampling with replacement from the original dataset.

Model Assessment

Bootstrap methods



Model Assessment

Bootstrap methods

- ▶ $S(Z)$ is any quantity computed from the data Z , for example, the prediction at some point.

- ▶ the estimate of the variance of $S(Z)$ is

$$\widehat{Var}[S(Z)] = \frac{1}{B-1} \sum_{b=1}^B (S(Z^{*b}) - \hat{S}^*)^2$$

- ▶ how to estimate the prediction error?

- ▶ $\widehat{Err} = \frac{1}{B} \frac{1}{N} \sum_{b=1}^B \sum_{n=1}^N L(y_n, f^{*b}(x_n))$ is optimistic because bootstrap datasets are acting as the training samples, while the original training set is acting as the test sample, and these two have observations in common.

- ▶ leave one out bootstrap estimate of the prediction error:

$$\widehat{Err}^{(1)} = \frac{1}{N} \sum_{n=1}^N \frac{1}{|C^{-i}|} \sum_{b \in C^{-i}} L(y_n, f^{*b}(x_n)), \text{ where } C^{-i} \text{ is the set of indices of the bootstrap samples that do not contain observation } i$$

- ▶ $\widehat{Err}^{(.632)} = .368 \overline{err} + .632 \widehat{Err}^{(1)}$
($Pr(\text{observation } i \in \text{bootstrap sample } b) = 1 - (1 - 1/N)^N \approx 1 - e^{-1} = 0.632$)

Ensemble Methods

Methods for Constructing Ensembles

- ▶ Subsampling the training data
 - ▶ Bagging
 - ▶ Boosting
- ▶ Manipulating the Input Features
- ▶ Manipulating the Output Targets
- ▶ Injecting Randomness

Model Averaging

Bagging (bootstrap aggregating)

Using bootstrap to improve the estimate or prediction itself.

Bagging is a meta-algorithm to improve machine learning of classification and regression models in terms of stability and classification accuracy. It also reduces variance and helps to avoid overfitting. Bagging is a special case of the model averaging approach.

For each bootstrap sample Z^{*b} we fit our model, giving prediction $f^{*b}(x)$. The bagging estimate is defined by

$$f_{bag}(x) = \frac{1}{B} \sum_{b=1}^B f^{*b}(x)$$

For classification, consider $f^{*b}(x)$ as an indicator vector with a single one and $K-1$ zeros. Then the bagged estimate is a K -vector (p_1, p_2, \dots, p_K) with p_k equal to the proportion of models predicting class k at x . Our predicted class is the one with more votes from the models.

Model Averaging

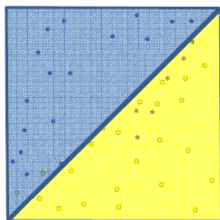
Bagging (bootstrap aggregating)

- ▶ under the squared-error loss, bagging (averaging) helps reducing variance and leaves bias unchanged.
- ▶ the same does not hold for classification under the 0-1 loss, because of the nonadditivity of bias and variance. Bagging a good classifier can make it better, but bagging a bad classifier can make it worse.
- ▶ when we bag a model, any simple structure in the model is lost.
A bagged tree is no longer a tree. For interpretation of the model, this can be a drawback.
- ▶ more stable models are not much affected by bagging.
- ▶ unstable models most helped by bagging are unstable because of the emphasis on interpretability, and this is lost in the bagging process.
- ▶ **Bumping**: choose the model that produces the smallest prediction error, averaged over the training set.

Boosting methods

Fighting the bias-variance tradeoff

- **Simple (a.k.a. weak) learners** e.g., naïve Bayes, logistic regression, decision stumps (or shallow decision trees)



Are good 😊 - Low variance, don't usually overfit

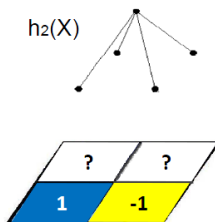
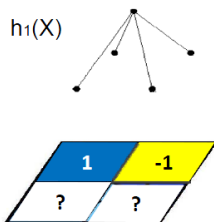
Are bad 😞 - High bias, can't solve hard learning problems

- **Can we make weak learners always good???**
 - **No!!!**
 - But often yes...**

Boosting methods

Voting (Ensemble Methods)

- Instead of learning a single (weak) classifier, learn **many weak classifiers** that are **good at different parts of the input space**
- **Output class:** (Weighted) vote of each classifier
 - Classifiers that are most “sure” will vote with more conviction
 - Classifiers will be most “sure” about a particular part of the space
 - On average, do better than single classifier!



$$H: X \rightarrow Y (-1, 1)$$

$$H(X) = h_1(X) + h_2(X)$$

$$H(X) = \text{sign}\left(\sum_t \alpha_t h_t(X)\right)$$

weights

Boosting methods

- ▶ procedure to combine the outputs of many 'weak' classifiers to produce a powerful 'committee'
- ▶ sequentially apply the weak classification algorithm to repeatedly modified versions of the data, thereby producing a sequence of weak classifiers $G_m(x)$, $m = 1, 2, \dots, M$.
- ▶ combine the predictions from all of them through a weighted majority vote $G(x) = \text{sgn} \left(\sum_{m=1}^M \alpha_m G_m(x) \right)$

Boosting methods

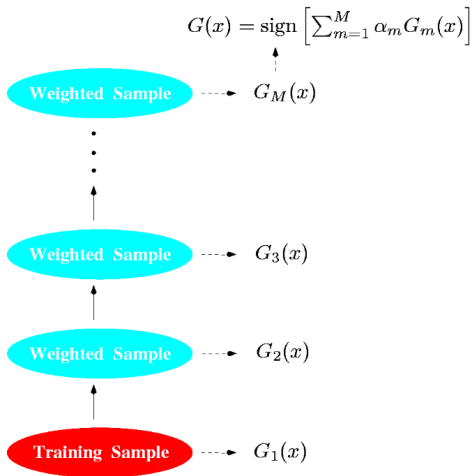
AdaBoost

- **Idea:** given a weak learner, run it multiple times on (reweighted) training data, then let learned classifiers vote
- On each iteration t :
 - weight each training example by how incorrectly it was classified
 - Learn a weak hypothesis – h_t
 - A strength for this hypothesis – α_t
- Final classifier:
$$H(X) = \text{sign}(\sum \alpha_t h_t(X))$$
- **Practically useful**
- **Theoretically interesting**

Boosting methods

AdaBoost

The data modifications at each boosting step consist of applying weights w_1, w_2, \dots, w_N to each of the training observations



Boosting methods

AdaBoost-Learning from weighted data

- **Consider a weighted dataset**
 - $D(i)$ – weight of i th training example (\mathbf{x}^i, y^i)
 - Interpretations:
 - i th training example counts as $D(i)$ examples
 - If I were to “resample” data, I would get more samples of “heavier” data points
- **Now, in all calculations, whenever used, i th training example counts as $D(i)$ “examples”**
 - e.g., in MLE redefine $Count(Y=y)$ to be weighted count

Unweighted data

$$Count(Y=y) = \sum_{i=1}^m \mathbf{1}(Y^i=y)$$

Weights $D(i)$

$$Count(Y=y) = \sum_{i=1}^m D(i) \mathbf{1}(Y^i=y)$$

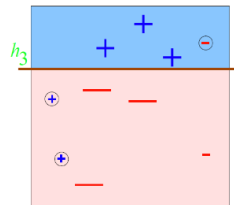
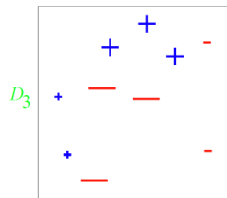
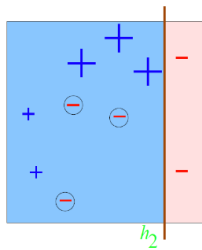
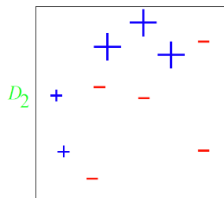
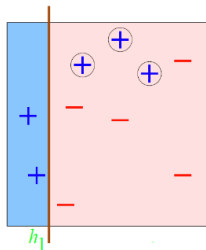
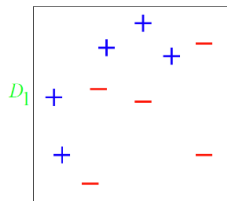
Boosting methods

AdaBoost

1. initialize the observation weights $w_i = 1/N, i = 1, \dots, N$
 2. for $m=1$ to M (the number of boosting steps)
 - 2.1 fit a classifier $G_m(x)$ to the training data using the weights w_i
 - 2.2 compute $err_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i}$
 - 2.3 compute $\alpha_m = \log((1 - err_m)/err_m)$
 - 2.4 update $w_i \leftarrow w_i \cdot \exp(\alpha_m I(y_i \neq G_m(x_i))), i = 1, \dots, N$
 3. output $G(x) = \text{sign}[\sum_{i=1}^M \alpha_m G_m(x)]$
- Adaboost is a forward stage additive modeling: sequentially adds new basis functions to the expansion without adjusting the parameters and coefficients of those that have already been added.

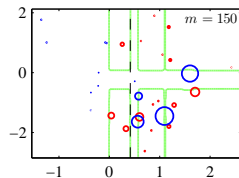
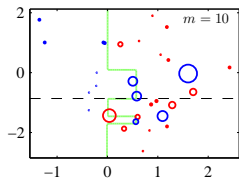
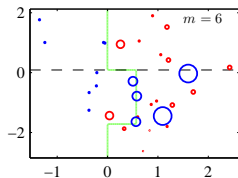
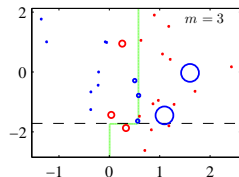
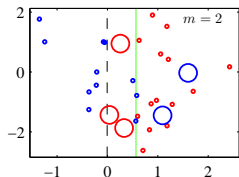
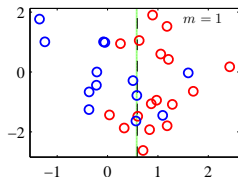
Boosting methods

AdaBoost



Boosting methods

AdaBoost



References



Aarti Singh

Model Selection

<http://www.cs.cmu.edu/~epxing/Class/10701/Lecture/lecture8.pdf>



Aarti Singh

Boosting

<http://www.cs.cmu.edu/~epxing/Class/10701/Lecture/lecture19.pdf>



Wikipedia

Wikipedia,

http://en.wikipedia.org/wiki/Main_Page



Tom Mitchell

Machine Learning,
Book.



Bishop

Pattern recognition and machine learning,
Book.



Trevor Hastie and Robert Tibshirani and Jerome Friedman

The elements of statistical learning,
Springer.