Experimental Data Analysis

in ©MATLAB

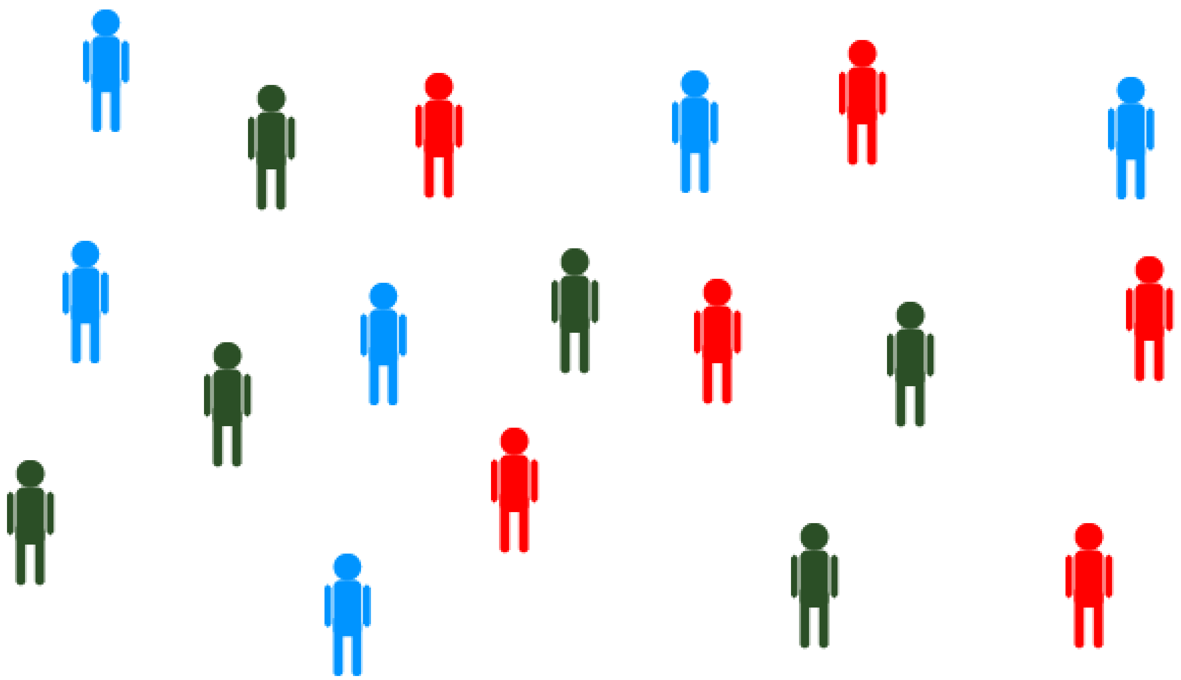
Lecture 10:

Machine learning (via kernel SVM), model validation, statistical models of the performance of binary classification test

Jan Rusz

Czech Technical University in Prague

Why machine learning?



To decide accurately such as possible and without influence of human factor!

Support vector machine: Optimization

Learning the SVM can be formulated as an optimization:

$$\max_{w} \frac{2}{\|w\|} \text{ subject to } w^T x + b \begin{cases} \geq 1 \; if \; y_i = +1 \\ \leq -1 \; if \; y_i = -1 \end{cases}$$

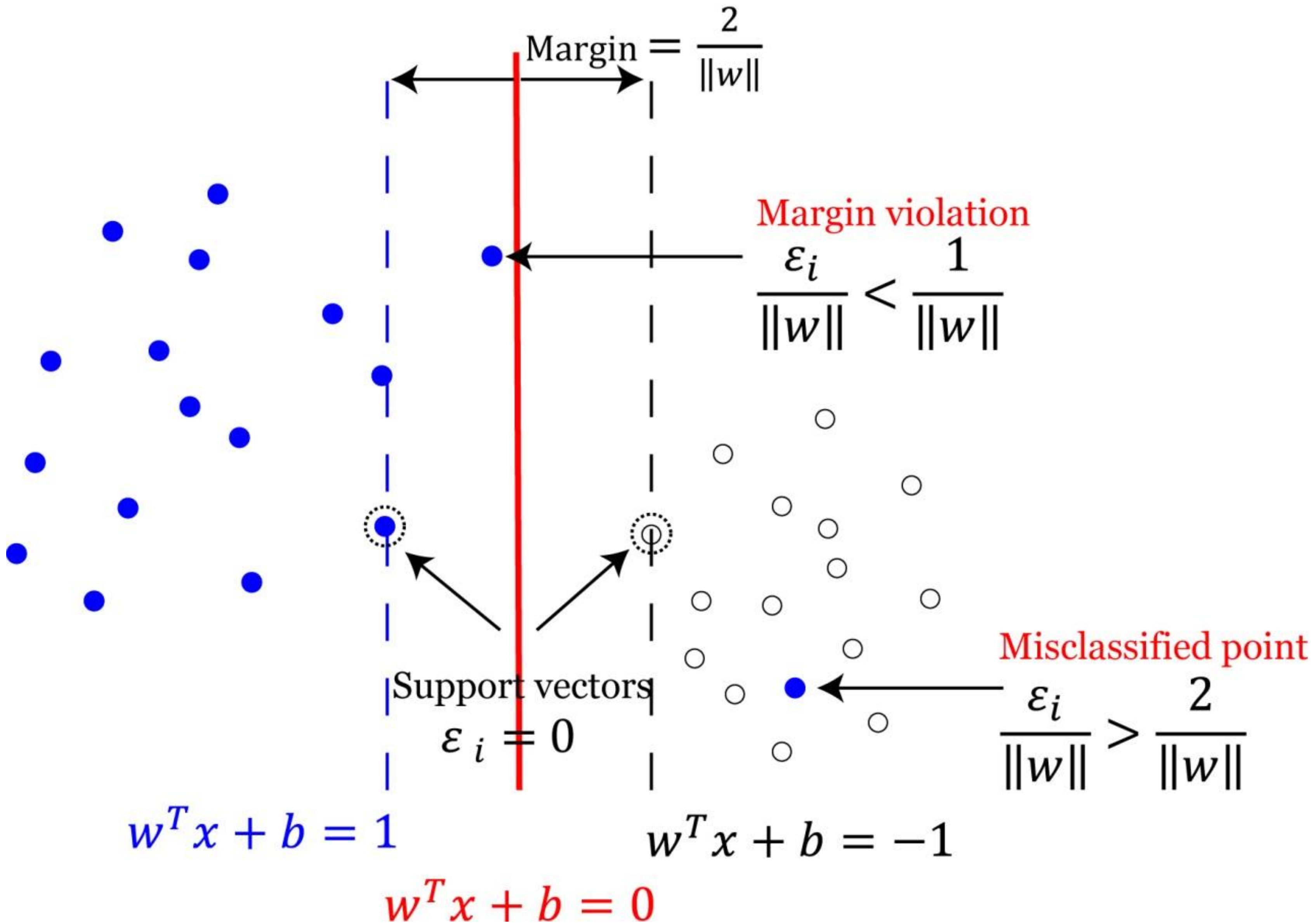
Or equivalently:

$$\min_{w} \|w\|^2 \text{ subject to } y_i(w^T x_i + b) \geq 1$$

Leading to quadratic optimization problem …

But what if data are not linearly separable??

SVM: Introducing "slack" variables $\varepsilon_i \geq 0$



Introducing regularization parameter C

The optimization problem becomes:

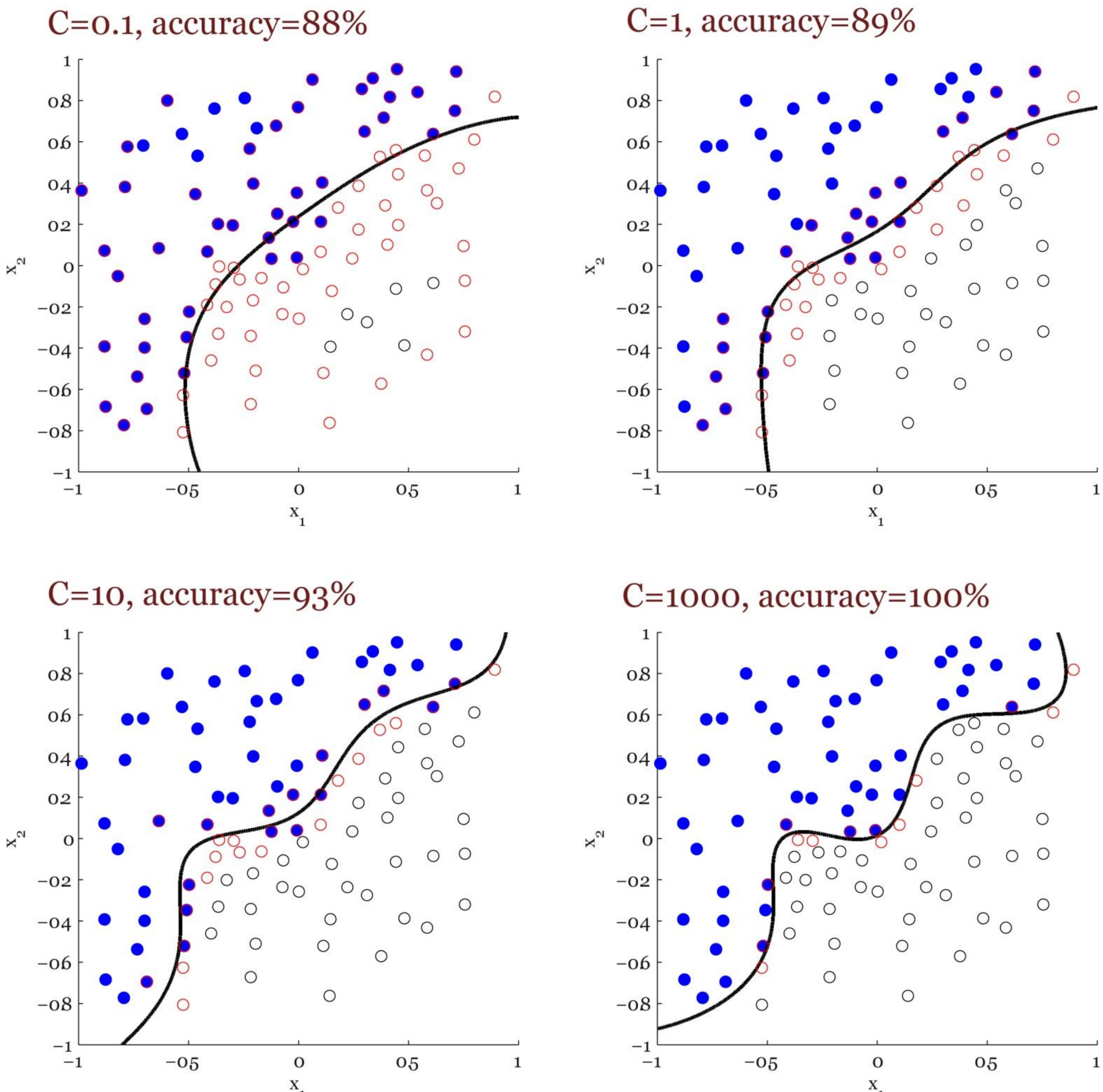
$$\min_{w,\varepsilon_i} \|w\|^2 + C \sum_{i=1}^{N} \varepsilon_i \text{ subject to } y_i(w^T x_i + b) \geq 1 - \varepsilon_i$$

Still leading to quadratic optimization problem…

C-parameter controls the smoothness of decision boundary:

- $C \to 0$ => large margin => smooth decision boundary
- $C \to \infty$ => narrow margin => convoluted decision boundary

Primal version of classifier $f(x) = w^T x + b$



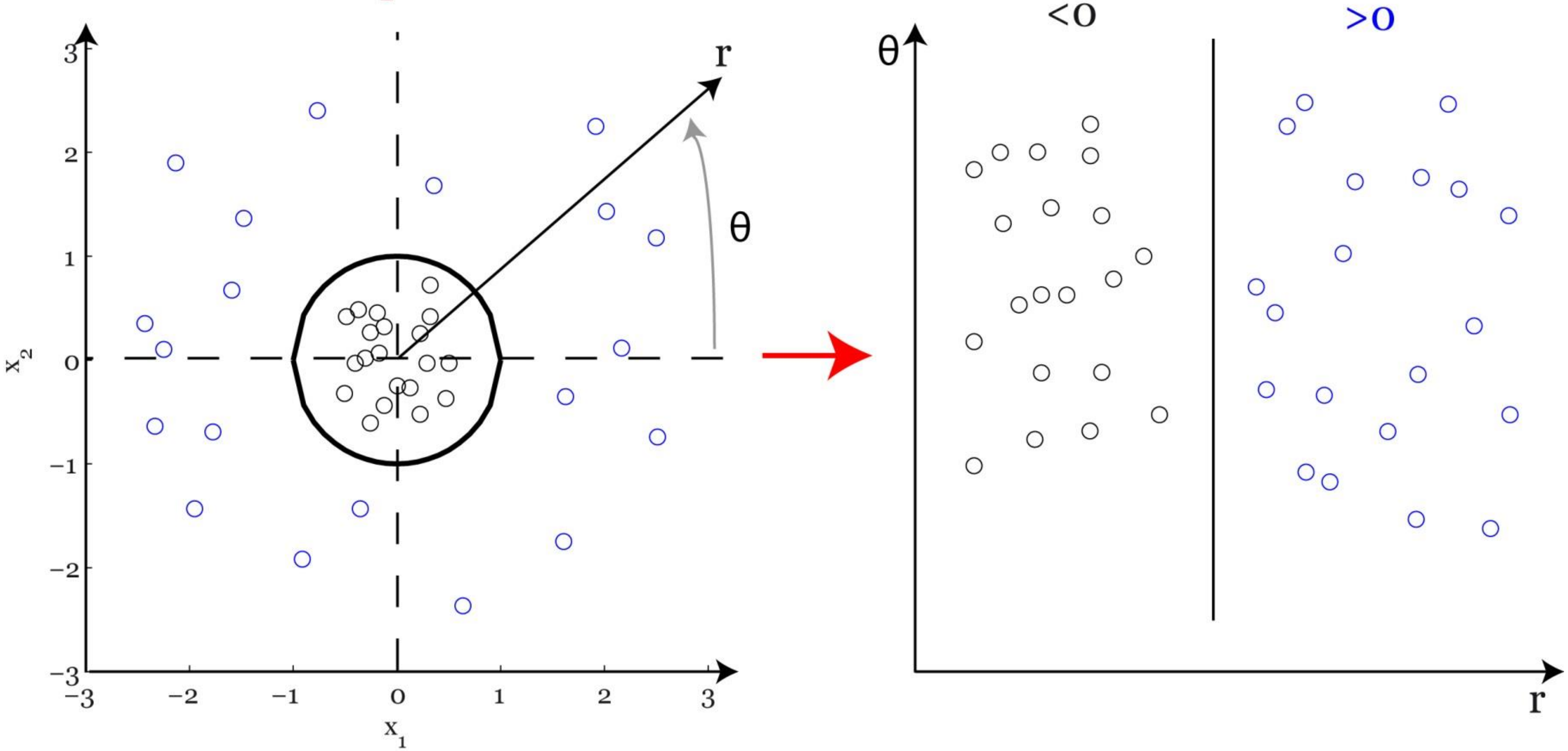
C=0.1, accuracy=88%
C=1, accuracy=89%
C=10, accuracy=93%
C=1000, accuracy=100%
x_1
x_2

Handling data that are not linearly separable?

We introduced slack variables.

But what if linear classifier is not appropriate??

Solution 1: use polar coordinates

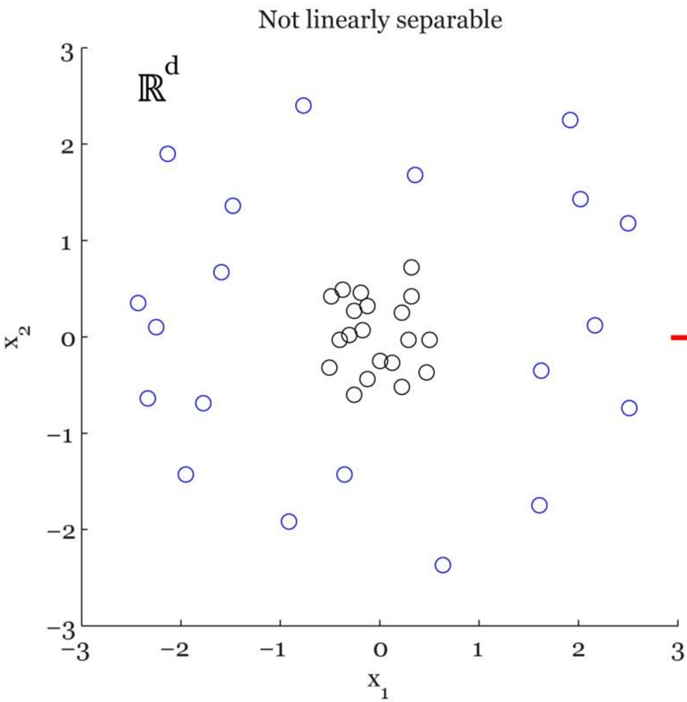


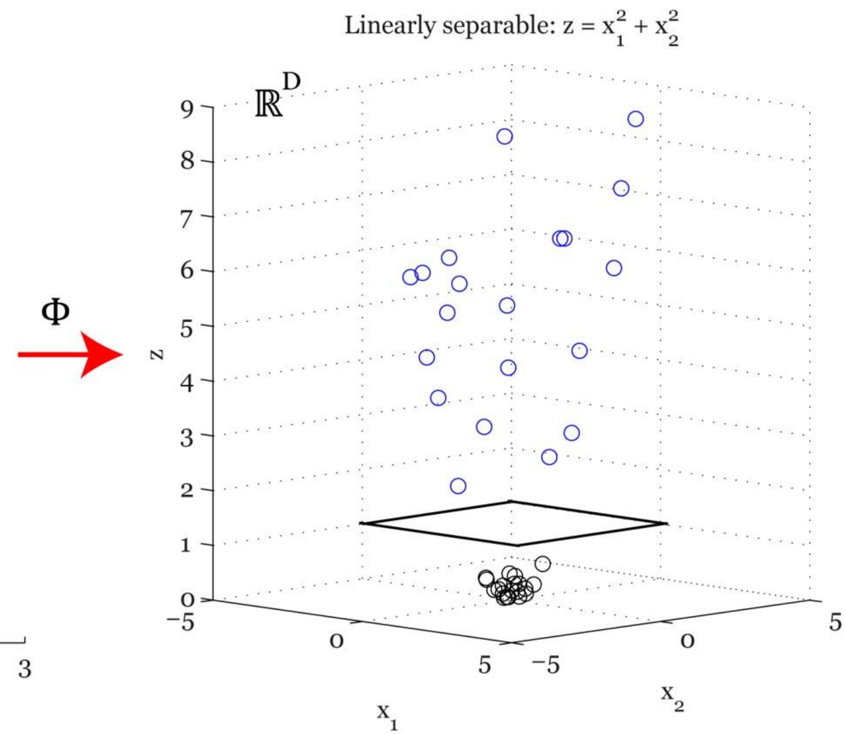
Data are linearly separable in polar coordinates

Acts non-linearly in original space

$$\Phi: \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r \\ \theta \end{pmatrix} \qquad \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

Solution 2: map data to higher dimension





Data are linearly separable in 3D space

Problem can still be solved as linear classifier

$$\Phi: \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix} \quad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$\Phi: x \rightarrow \Phi(x) \quad \mathbb{R}^d \rightarrow \mathbb{R}^D$$

$$f(x) = w^T \Phi(x) + b$$

feature map

Classifier is linear i w for $\mathbb{R}^D$

Kernel trick

$$f(x) = \sum_{i}^{N} \alpha_i \, y_i k(x_i, x) + b$$

Dual version of the classifier

$$f(x) = \sum_{i}^{N} \alpha_i \, y_i x_i^{\mathrm{T}} x + b$$

$$f(x) = \sum_{i}^{N} \alpha_i \, y_i \Phi(x_i)^{\mathrm{T}} \Phi(x) + b$$

α_i = weight (may be zero, $0 \leq \alpha_i \leq C$ for $\forall i$ and $\Sigma_i \, \alpha_i \, y_i = 0$)
x_i = support vector
N = size of training data

Kernel: $k(\mathrm{x_j}, \mathrm{xi}) = \Phi(x_j)\Phi(xi)$

MIT OpenCourseWare
https://www.youtube.com/watch?v=_PwhiWxHK8o

Kernel examples

Linear kernels $k(\mathrm{x},\mathrm{x}') = \mathrm{x}^{\mathrm{T}}\mathrm{x}'$

Polynomial kernels $k(\mathrm{x},\mathrm{x}') = (1 + \mathrm{x}^{\mathrm{T}}\mathrm{x}')^{d}$ for any $d > 0$

- contains all polynomial terms up to degree d

Gaussian kernels $k(\mathrm{x},\mathrm{x}') = \exp(-||\mathrm{x} - \mathrm{x}'||^{2}/\sigma^{2})^{d}$ for $\sigma > 0$

- infinite dimensional feature space

Radial Basis Function (RBF) SVM

$$f(x) = \sum_{i}^{N} \alpha_i \, y_i \exp\left(\frac{-\|x - x_i\|^2}{\sigma^2}\right) + b$$

Influenced by 2 parameters:

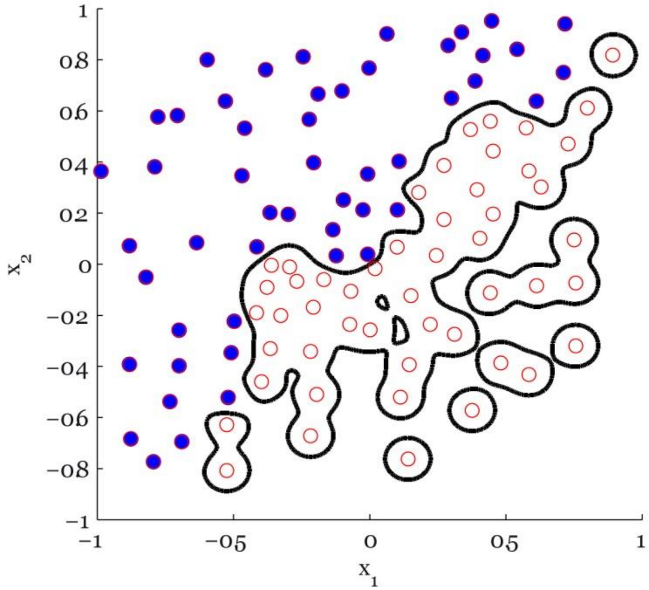
C-parameter controls the smoothness of decision boundary:

- $C \rightarrow 0$ => large margin => smooth decision boundary
- $C \rightarrow \infty$ => narrow margin => convoluted decision boundary

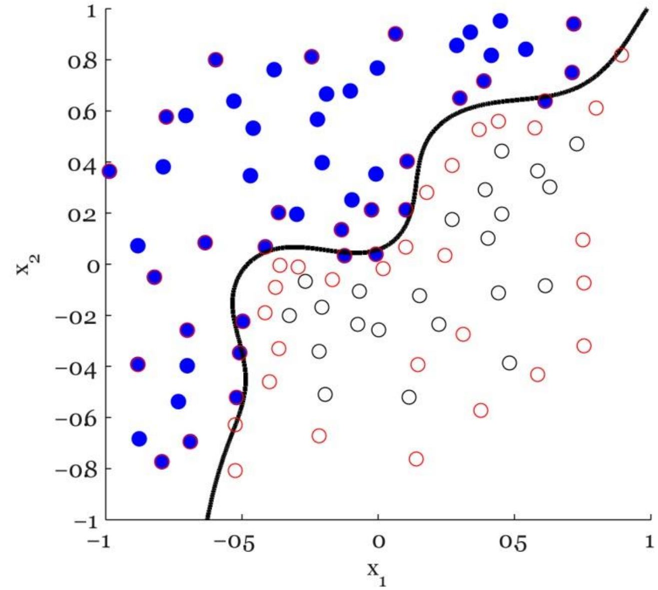
σ-parameter represents inverse of the radius of influence of samples selected by the model as support vectors:

- $\sigma \rightarrow 0$ => decision boundary tends to be too flexible => hazard of overfitting
- $\sigma \rightarrow \infty$ => decision boundary tends to be constrained and cannot capture the complexity or shape of the data => it is influenced by entire training set and behave similarly to linear model => tends to make wrong classification while predicting but avoid the hazard of overfitting

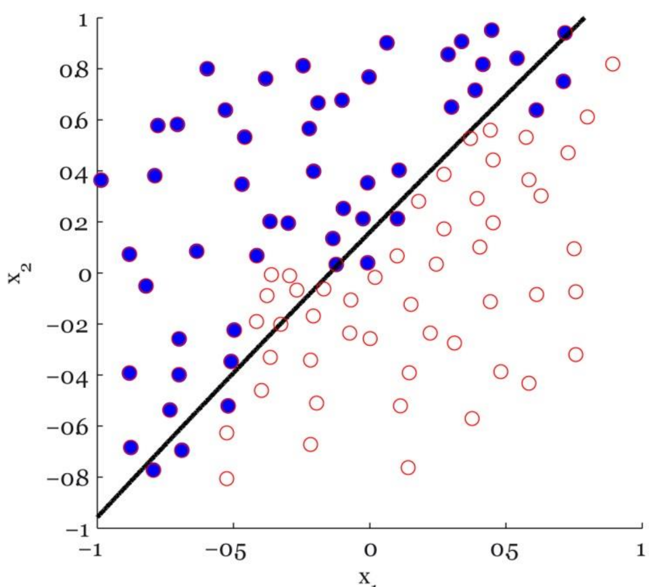
C=0.1, σ=0.1, accuracy=100%



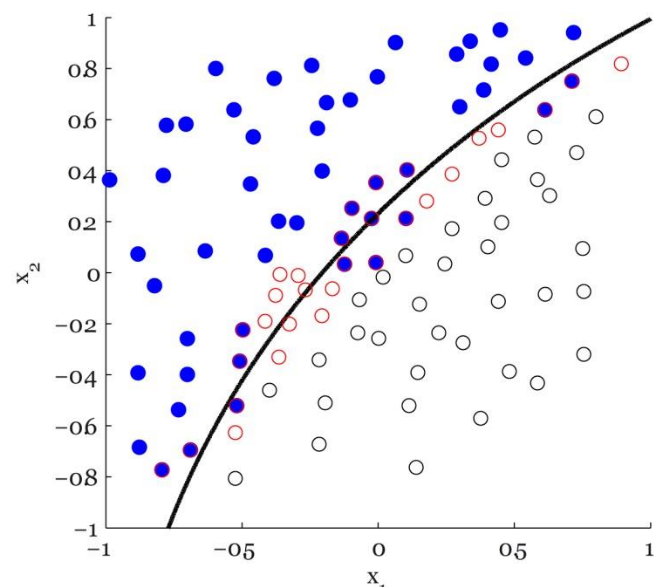
C=1, σ=0.5, accuracy=95%



C=1, σ=10, accuracy=88%

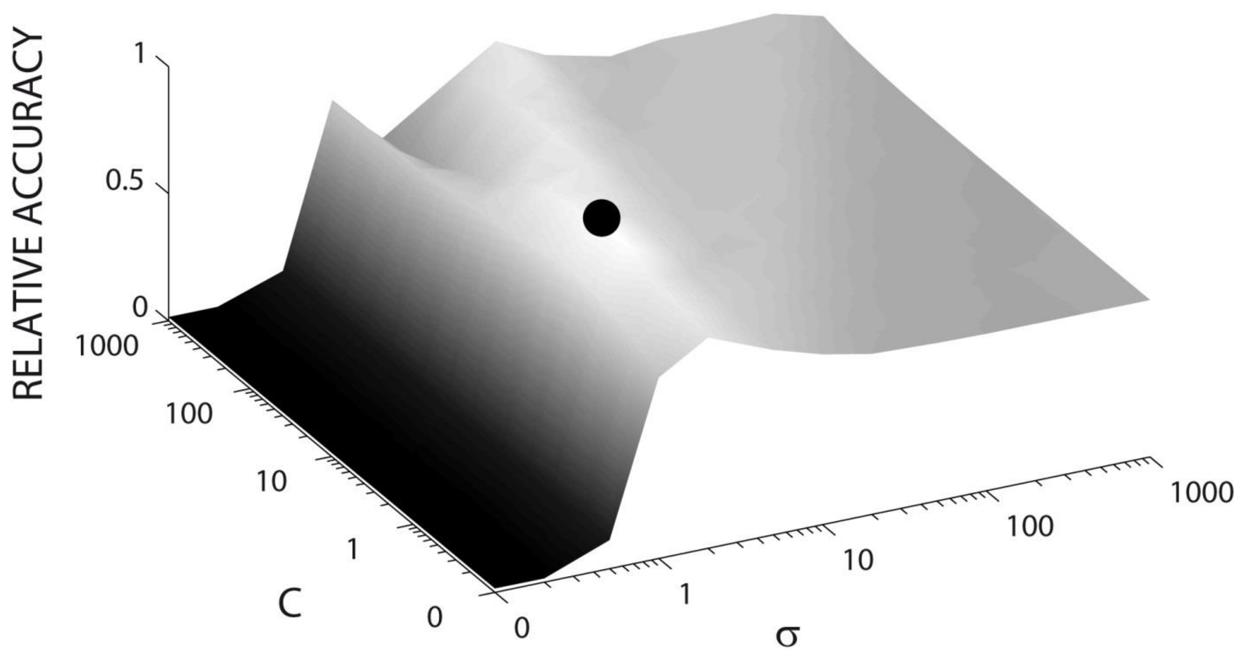


C=1000, σ=10, accuracy=89%



How to select optimal C and σ parameters?

Grid search: determination of the optimal parameter C and σ over the defined sets, for example C = $[2^{-15}, 2^{-12}, \ldots, 2^{15}]$ and σ = $[2^{-15}, 2^{-12}, \ldots, 2^{3}]$



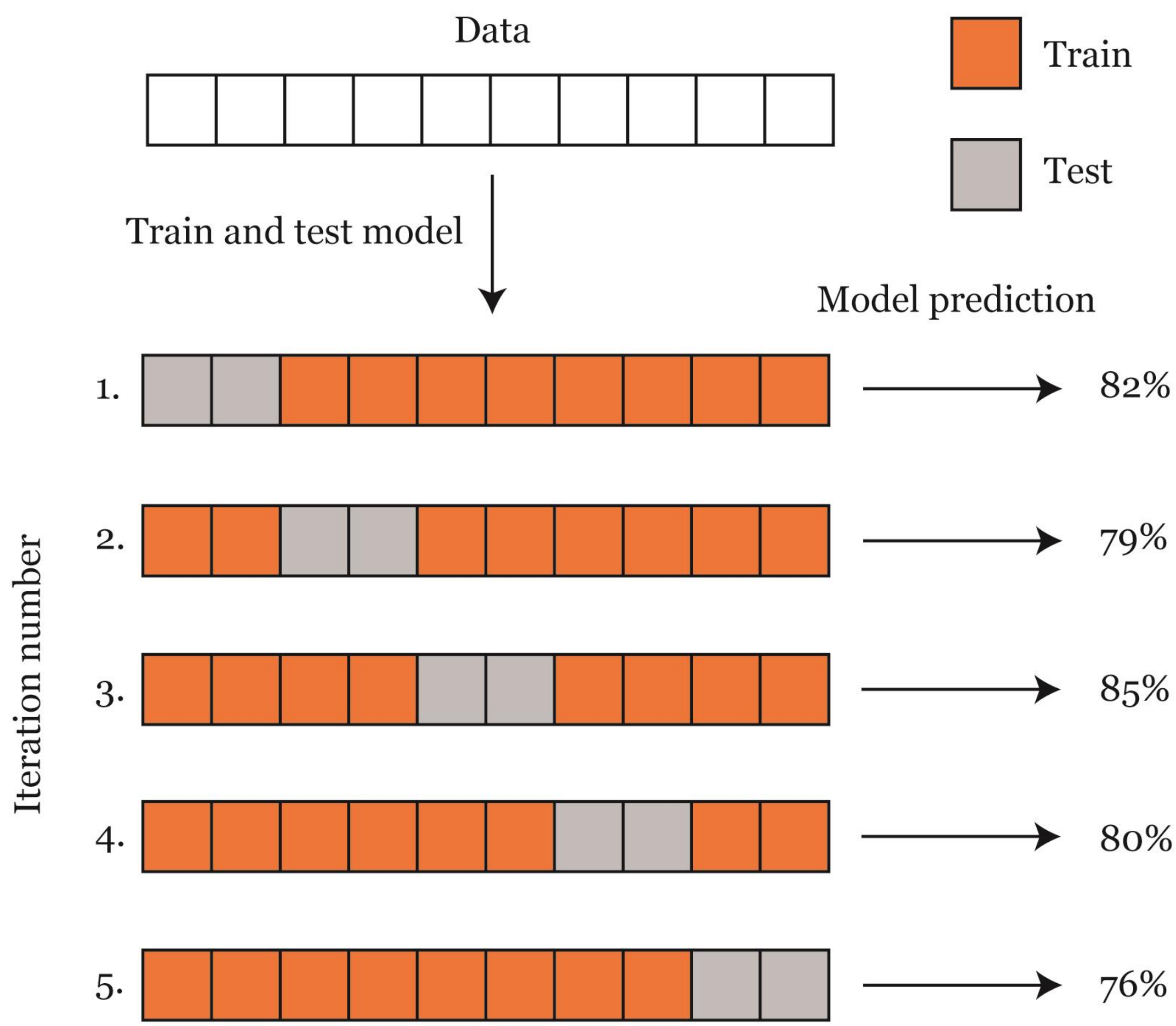
How to select optimal measures?

- Exhaustive search for all possible measure combinations
- Least absolute shrinkage and selection operator (LASSO)
- Minimum redundancy maximum relevance (mRMR)
- Local learning-based feature selection (LLBFS)
- Margin maximization using the k-Nearest-Neighbor (RELIEF)

Cross-validation

- method of estimating expected prediction error
- helps selecting the best fit model
- helps ensuring model is not over fit
- approach:
    - leave some data out
    - fit model
    - evaluate model on left-out data

K-fold cross-validation

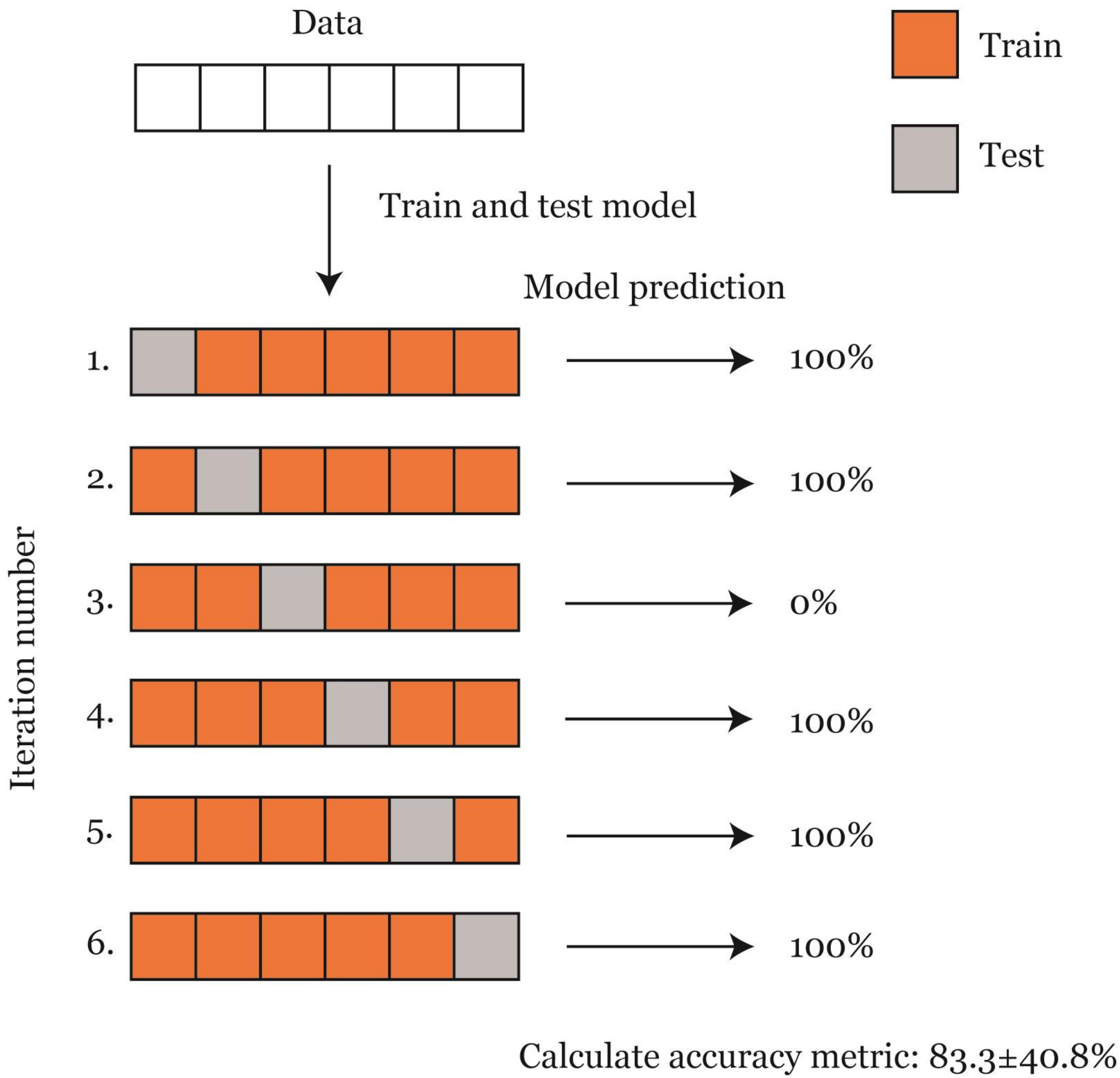


Calculate accuracy metric: 80.4±3.4%

K-fold cross-validation

- k equal sized subsamples
- validation process is repeated k times (folds)
- k results from the folds can be averaged to produce a single estimation
- advantage is that all observations are used for both training and validation, each observation is used exactly once
- 5-10 fold cross-validation is typically used

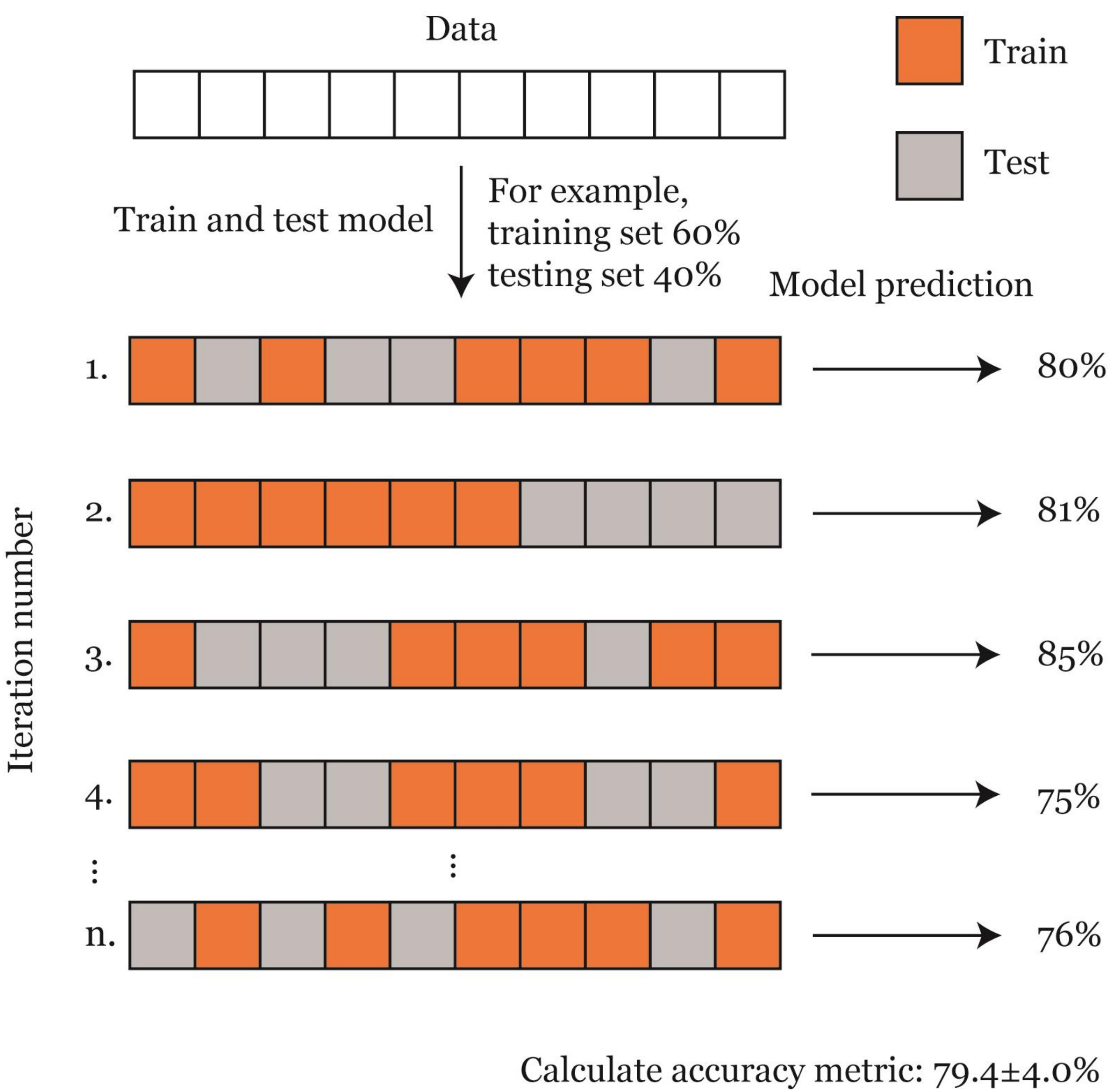
Leave-one-out cross-validation



Leave-one-out (LOO) cross-validation

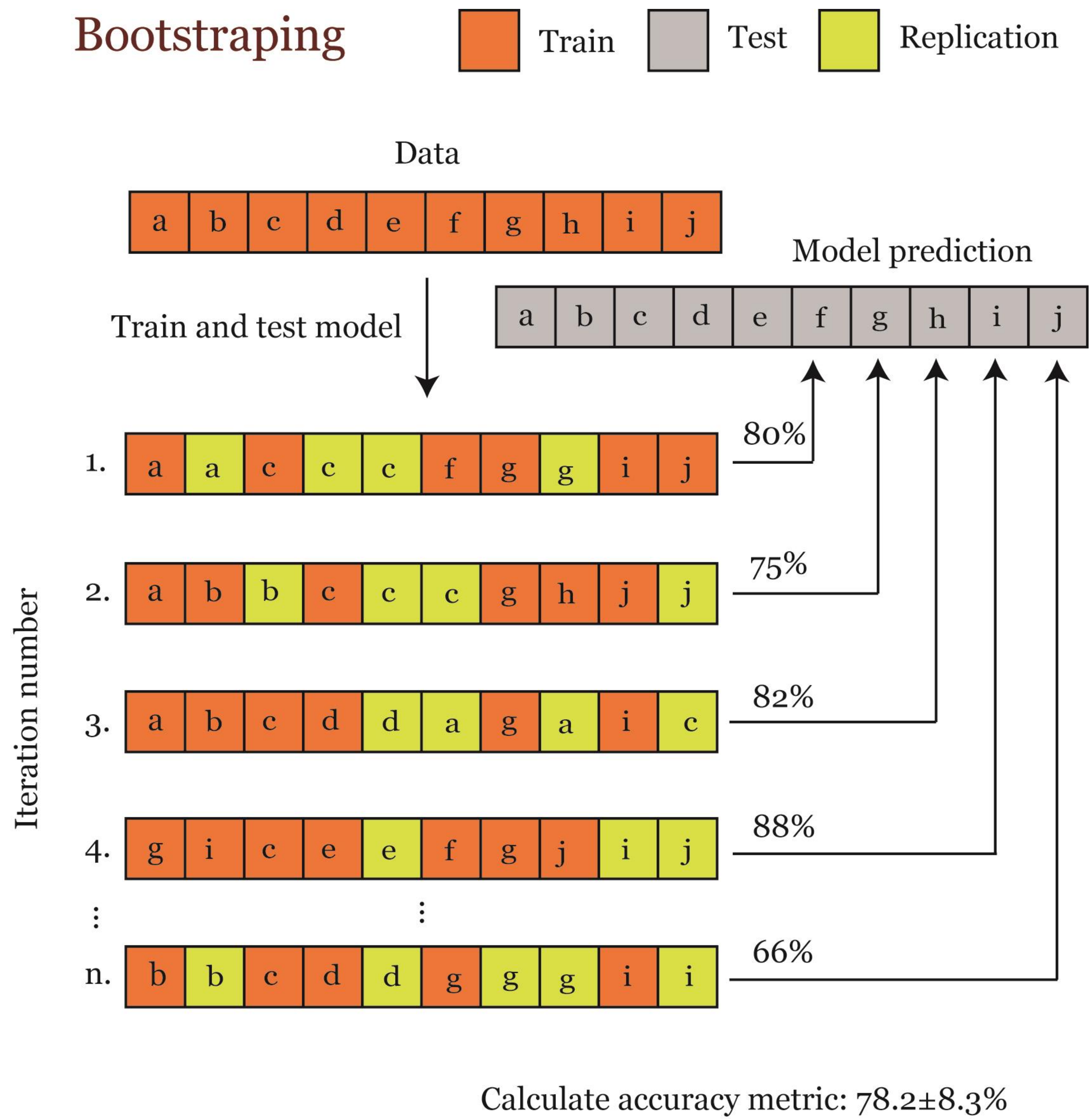
- special case of k-fold cross-validation where k=n
- accurate and typically used for small sample size data
- disadvantage is high standard deviation
- high computation time

Monte Carlo cross-validation



Monte Carlo (repeated random sub-sampling) cross-validation

- randomly splits dataset into training and testing set
- advantage over k-fold cross-validation is that proportion of training/testing split is not dependent on number of folds
- disadvantage is that some observations may never be selected in the testing/validation subsample
- accuracy is dependent on number of performed iterations
- may be computationally demanding
- typically more than 10 iterations are recommended


Bootstraping
Train
Test
Replication
Data
a b c d e f g h i j
Model prediction
a b c d e f g h i j
Train and test model
1. a a c c c f g g i j
80%
2. a b b c c c g h j j
75%
3. a b c d d a g a i c
82%
4. g i c e e f g j i j
88%
n. b b c d d g g g i i
66%
Iteration number
Calculate accuracy metric: 78.2±8.3%

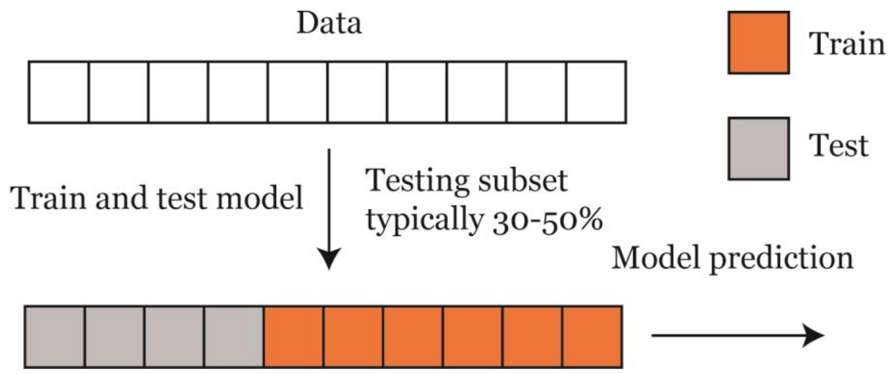
Bootstrapping

- typically 50 replicates

Bootstrapping vs. cross-validation

- bootstrapping gives idea how stable is the model
- cross-validation gives clue how much one can expect that data generalize to new data sets
- bootstrapping = measure of reliability/stability
- cross-validation = measure of validity

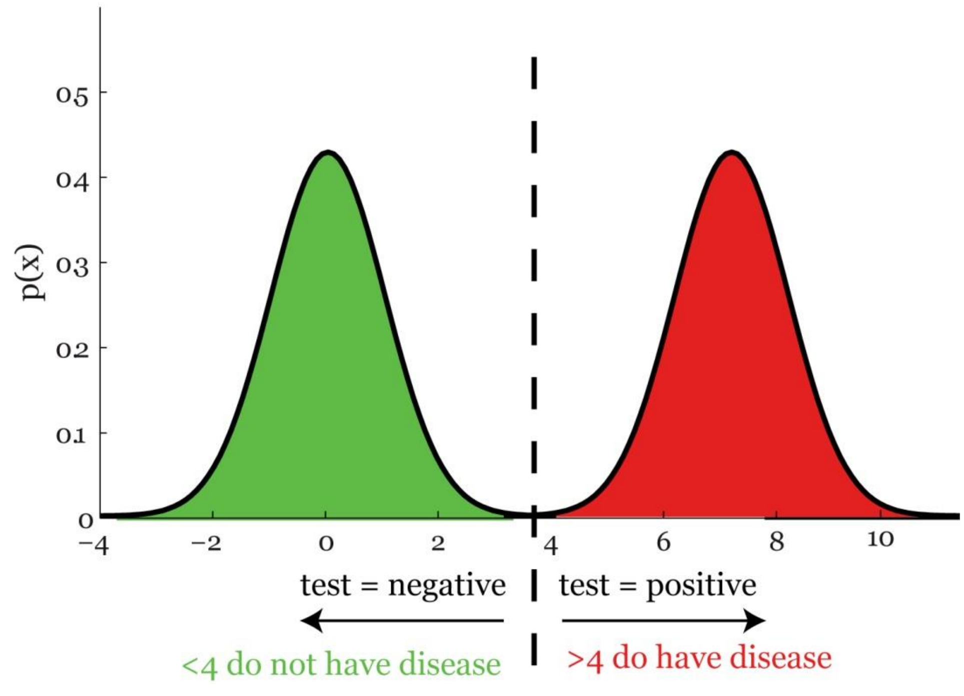
Holdout method



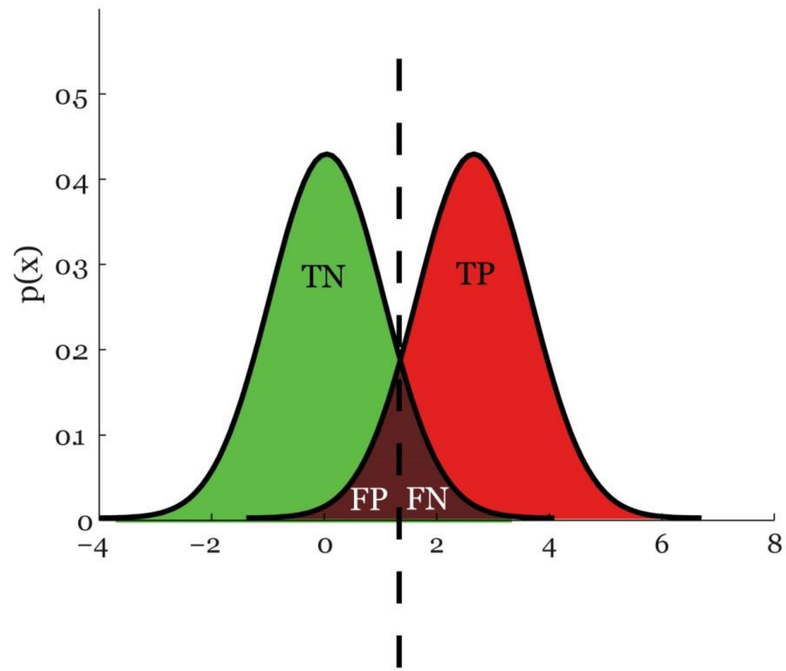
- involves single run
- “the simplest kind of cross-validation”
- testing data are never used for training

- important especially in medical research, validation on independent subset
- typically, part of data are used for model fit using cross-validation and model is then validated using hold-out method

Tradeoff between sensitivity and specificity

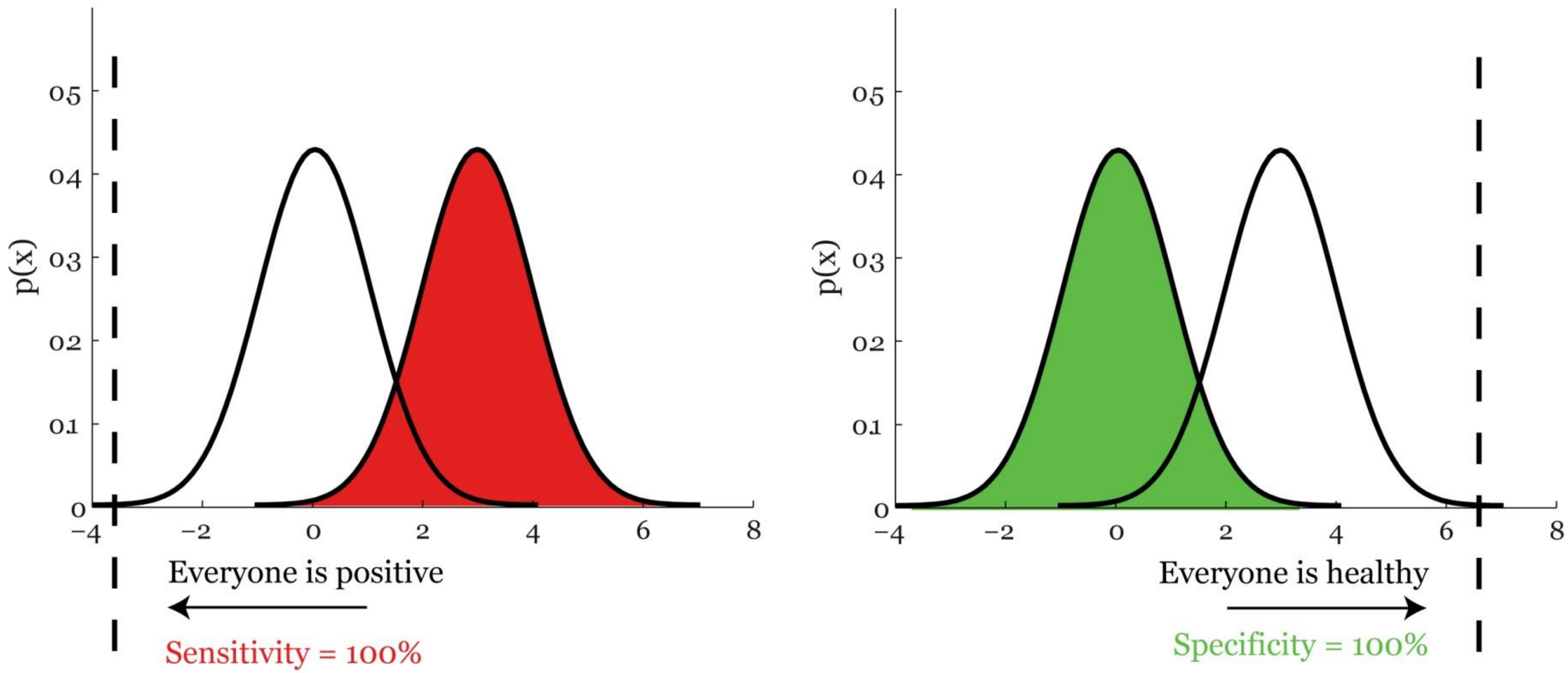


Tradeoff between sensitivity and specificity

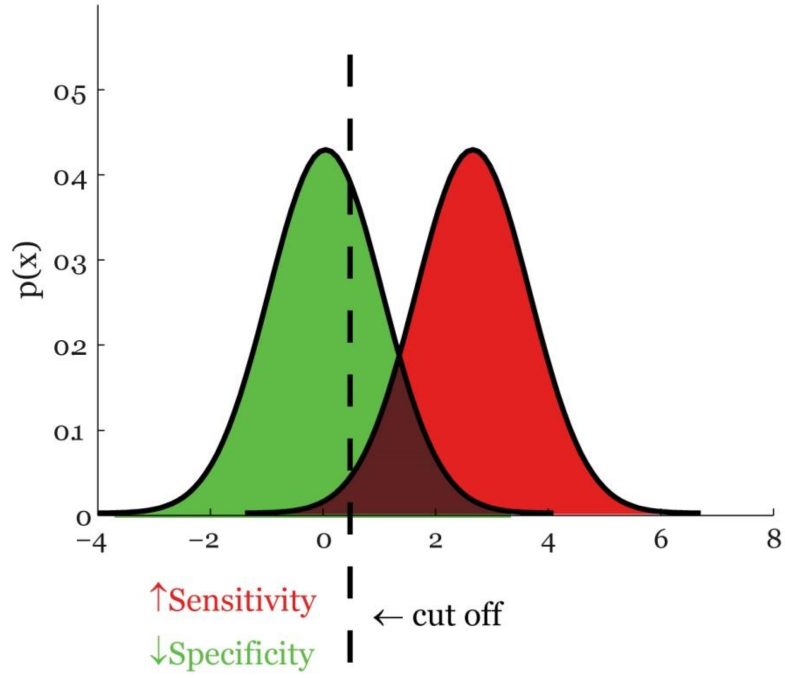


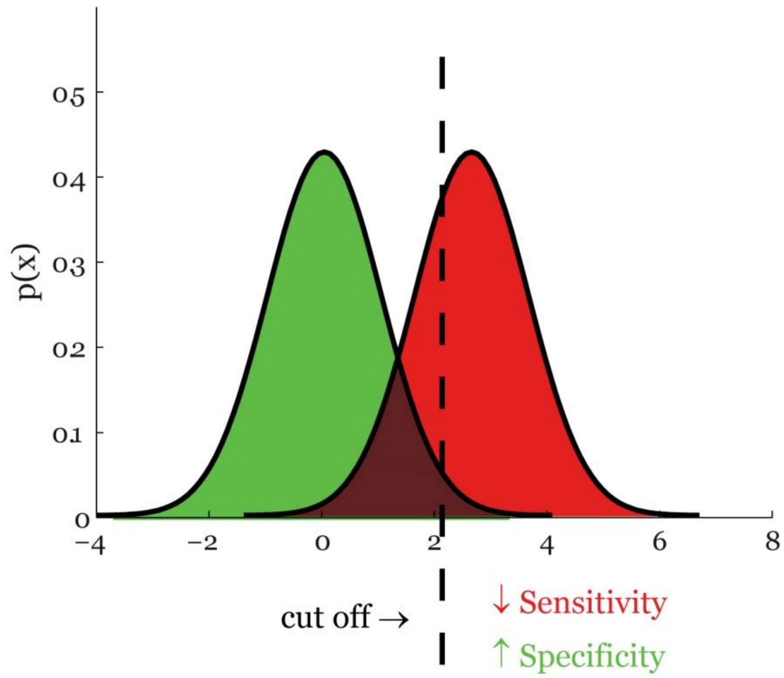
Predicted condition	True condition: Negative	True condition: Positive
Positive	False positive Type I error	True positive
Negative	True negative	False negative Type II error

Tradeoff between sensitivity and specificity



Tradeoff between sensitivity and specificity





Confusion matrix

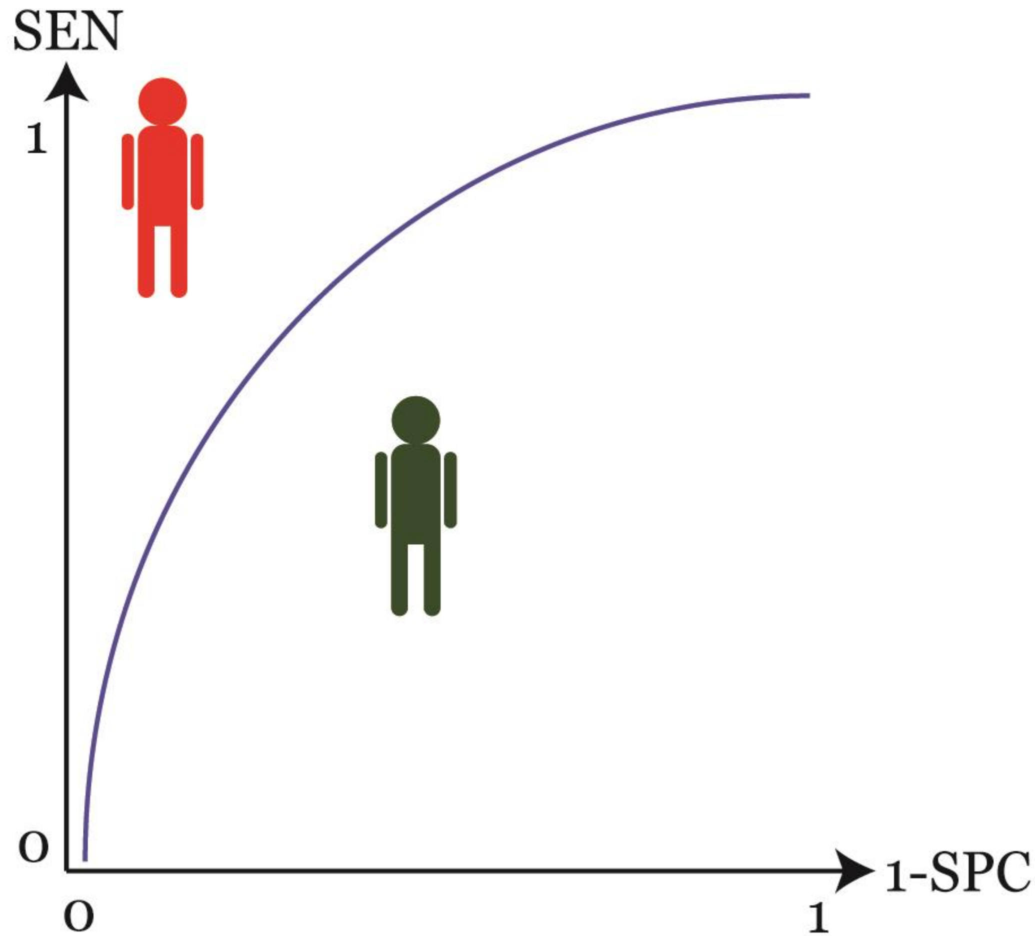
	True condition		
Predicted condition	Negative	Positive	
Positive	False positive Type I error	True positive	Positive predictive value $PPV = \frac{TP}{TP + FP}$
Negative	True negative	False negative Type II error	Negative predictive value $NPV = \frac{TN}{FN + TN}$
	Specificity $SPC = \frac{TN}{TN + FP}$	Sensitivity $SEN = \frac{TP}{TP + FN}$	**Accuracy** $ACC = \frac{TP + TN}{TP + FP + FN + TN}$

F-score

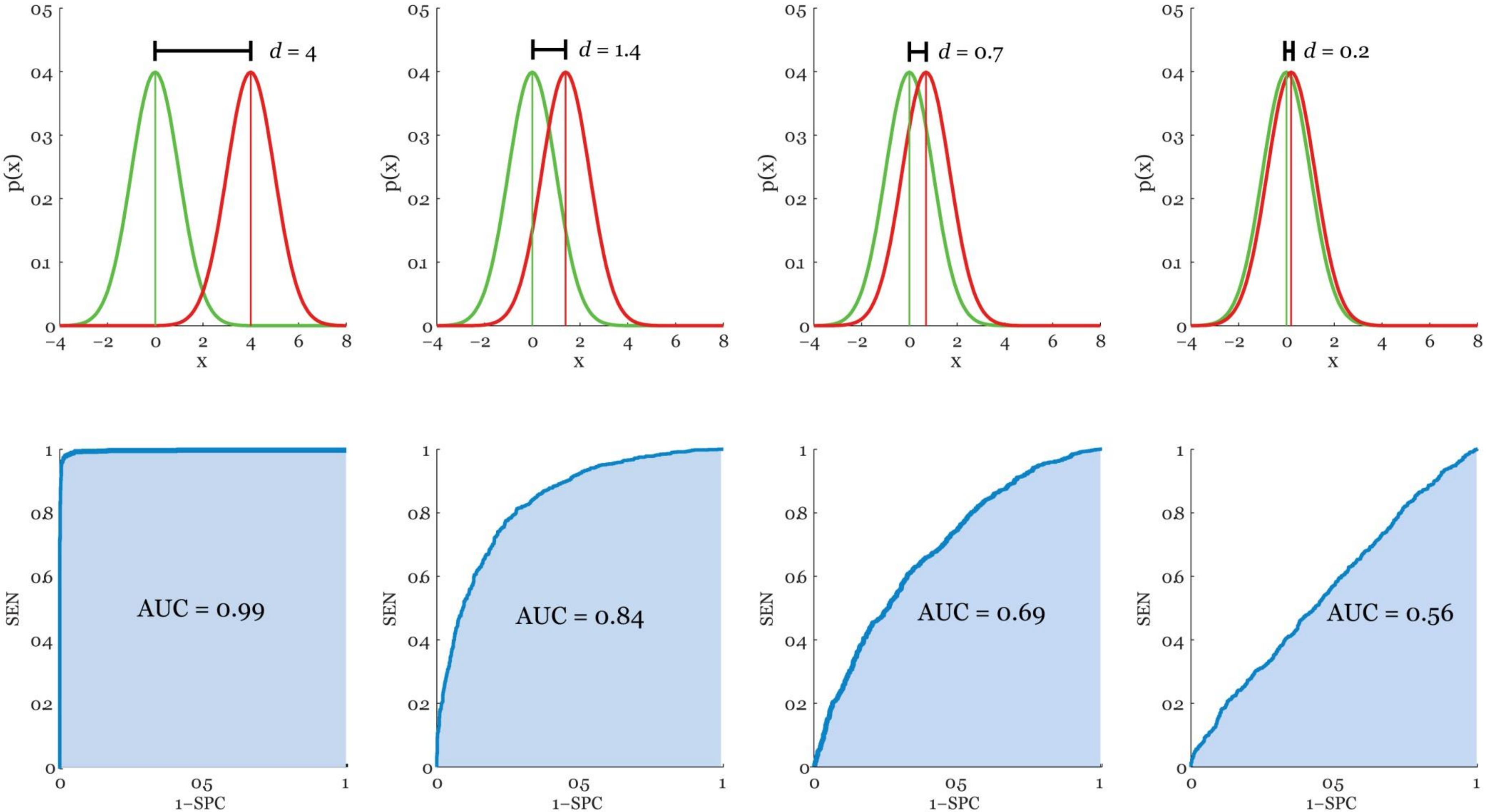
$$precision = \frac{TP}{TP + FP} \quad recall = \frac{TP}{TP + FN} \quad F = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

F-score is independent of the number of TN, which is generally unknown

Receiver operator characteristic (ROC) curve



Area under curve (AUC)



Matlab example 1

AUC	Quality of test
0.9–1.0	Excellent
0.8–0.9	Good
0.7–0.8	Fair
0.6–0.7	Poor
0.5–0.6	Fail