

Funkce fitcsvm (natrénování SVM klasifikátoru)

- syntax: SVMStruct = fitcsvm (data_training, group, 'parameter', 'value')

| Parametr | Možné hodnoty a komentář | Default |
|------------------|---|----------|
| 'BoxConstraint' | nastavení parametru C | 1 |
| 'KernelFunction' | 'linear' – lineární jádro 'gaussian' – Gaussovské jádro 'rbf' – radiální bázové jádro 'polynomial' – polynomiální jádro (s defaultem PolynomialOrder=3) @kfun – vlastní funkce | 'linear' |
| 'Solver' | 'ISDA' – nalezení hranice pomocí Iterative Single Data Algorithm 'L1QP' – nalezení hranice pomocí kvadratického programování 'SMO' – nalezení hranice pomocí sekvenční minimální optimalizace | 'SMO' |
| 'Standardize' | 'true' – standardizace dat (odečtením průměru a vydělením SD) 'false' – data bez žádné úpravy | 'false' |

- ukázka: SVMStruct = fitcsvm (data_training, group, 'BoxConstraint', 0.1)
- další parametry: 'CrossVal', 'Kfold', 'IterationLimit', 'Cost',...
- následná klasifikace: class = predict (SVMStruct,data_testing)
- pro data s velkým počtem proměnných použití funkce fitlinear místo fitcsvm

Funkce svmtrain (natrénování SVM klasifikátoru)

- syntax: SVMStruct = svmtrain (data_training, group, 'parameter', 'value')

| Parametr | Možné hodnoty a komentář | Default |
|-------------------|---|----------|
| 'boxconstraint' | nastavení parametru C | 1 |
| 'kernel_function' | 'linear' – lineární jádro 'quadratic' – kvadratické jádro 'polynomial' – polynomiální jádro (s defaultem polyorder=3) 'rbf' – radiální bázové jádro (s defaultem rbf_sigma=1) 'mlp' – vícevrstvé perceptronové jádro (s defaultem mlp_params=[1 -1]) <i>@kfun</i> – vlastní funkce | 'linear' |
| 'method' | 'QP' – nalezení hranice pomocí kvadratického programování 'SMO' – nalezení hranice pomocí sekvenční minimální optimalizace 'LS' – nalezení hranice pomocí metody nejmenších čtverců | 'SMO' |
| 'autoscale' | 'true' – centrování a standardizace dat 'false' – data bez žádné úpravy | 'true' |

- ukázka: SVMStruct = svmtrain (data_training, group, 'boxconstraint', 0.1)
- další parametry: 'kernelcachelimit', 'kktviolationlevel', 'tolkkt', 'showplot'
- následná klasifikace: class = svmclassify (SVMStruct,data_testing)

Funkce classify (natrénování i otestování Bayesova k.)

- syntax: `class = classify (data_testing, data_training, group, 'type', prior)`

| Parametr | Komentář | Default |
|----------|--|---------------------|
| 'type' | Píše se přímo název typu klasifikátoru. Možnosti jsou: 'linear' – používá váženou kovarianční matici (stejnou pro obě skupiny) 'diaglinear' – používá diagonální kovarianční matici (stejnou pro obě skupiny) – tzv. naivní Bayesův klasifikátor 'quadratic' – kovarianční matice počítána pro každou skupinu zvlášť 'diagquadratic' – diagonální kovarianční matice pro každou skupinu zvlášť 'mahalanobis' – výpočet Mahalanobisových vzdáleností, kovarianční matice pro každou skupinu zvlášť | 'linear' |
| prior | Vektor apriorních pravděpodobností pro třídy (pokud chceme zvolit jiné, než které se vypočítají z trénovacích dat). | výpočet z trén. dat |

- ukázka: `class = classify (data_testing, data_training, group, 'quadratic', [0.3 0.7])`
- poznámka: naivní Bayesův klasifikátor předpokládá, že proměnné jsou nekorelované

Funkce fitcknn (natrénování k-NN klasifikátoru)

- syntax: `model = fitcknn(data_training, group, 'parameter', 'value')`

| Parametr | Možné hodnoty a komentář | Default |
|----------------|--|-------------------|
| 'NumNeighbors' | počet nejbližších sousedů | 3 |
| 'Distance' | 'cityblock', 'chebychev', 'correlation', 'cosine', 'euclidean', 'hamming' (pro kategoriální data), 'jaccard', 'mahalanobis', 'minkowski', 'seuclidean', 'spearman', vlastní fce | 'euclidean' |
| 'NSMethod' | 'kdtree' – vytvoření kd-stromu pro rychlejší vyhledávání (možné jen pro: 'euclidean', 'cityblock', 'minkowski', 'chebychev') 'exhaustive' – vypočítá vzdálenosti všech subjektů v testovacích datech od všech subjektů v trénovacích datech | závislé na datech |
| 'BreakTies' | 'smallest' – při „ties“ subjekt zařazen do skupiny s menším identifikátorem skupiny 'nearest' – při „ties“ subjekt zařazen do skupiny s nejbližším sousedem 'random' – při „ties“ subjekt zařazen do jedné ze skupin náhodně | 'smallest' |
| 'IncludeTies' | 'true' – ke klasifikaci se použijí všichni sousedi se vzdáleností $\leq k$ -tá nejmenší vzdálenost 'false' – ke klasifikaci se použije právě k nejbližších sousedů | 'false' |

- ukázka: `model = fitcknn(train,group,'NumNeighbors',3,'NSMethod','exhaustive','Distance','euclidean')`
- další parametrů je mnoho (viz. <https://www.mathworks.com/help/stats/fitcknn.html>)
- následná klasifikace: `class = model.predict(data_testing)`

Funkce TreeBagger (natrénování klasifikačních lesů)

- syntax: `model = TreeBagger (NumTrees, data_training, group, 'parameter', 'value')`

| Parametr | Možné hodnoty a komentář | Default |
|-------------------------|---|------------------|
| 'Method' | 'classification' – vytvoření klasifikačních lesů 'regression' – vytvoření regresních lesů | 'classification' |
| 'SampleWithReplacement' | 'on' – výběr subjektů (či objektů) s opakováním 'off' – výběr subjektů (či objektů) bez opakování | 'on' |
| 'InBagFraction' | číslo od 0 do 1 – proporce subjektů, ze kterých se náhodně vybírá (pokud 'SampleWithReplacement'='off', je nutné zvolit číslo menší než 1) | 1 |
| 'NumPredictorsToSample' | číslo od 0 do p (kde p je počet proměnných) nebo 'all' (u regresních stromů je defaultem $p/3$) | \sqrt{p} |
| 'MinLeafSize' | minimální počet subjektů (či objektů) v listu (u regresních stromů je defaultem 5) | 1 |

- ukázka: `model = TreeBagger(100, data_training, group, 'NumPredictorsToSample', 'all')`
- další parametry: 'Cost', 'OOBPrediction',
- následná klasifikace: `class = predict (model, data_testing)`