Machine learning:
SVM, ANN, ensembles, active learning, practical issues

Agenda

- SVM
- Neural networks
- Ensemble methods
- Active learning
- Practical issues

Logistic Regression

- Probabilistic linear classifier
- Logistic (sigmoid) function $f(x)=1/(1+e^{-x})$
  - Where $x = w_0 + \sum_i w_i x_i$
  - $f(x) = P(C=1|X)$

- $w_0 + \sum_i w_i x_i = 0$ defines a hyperplane where $P(C=1|X) = 0.5$ and $P(C=0|X) = 0.5$

  and $w_0 + \sum_i w_i x_i$ is proportional to the distance from the hyperplane
- Learning
  - no closed form solution - optimization, e.g., with gradient descent
  - definition of a cost function (several options); $-y \log(y') - (1-y) \log(1-y')$; y in $\{0,1\}$
  - updating of weights (according to optimization results); $w_j = w_j - \alpha \sum_i (y'_i - y_i)x_{ij}$ for all instances, multiple times

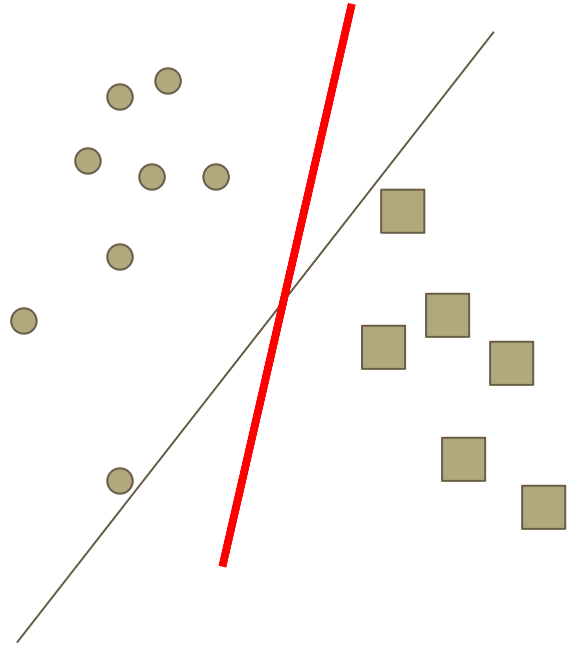
SVM

- Linear binary classifier (not probabilistic)
- Extension of linear classifiers to model non-linear decision boundaries
  - Transformation of the feature space using synthetic features of higher order

    $y' = w_0 + w_1x_1 + w_2x_2 \qquad + w_3x_1^2 + w_4x_2^2 + w_5x_1x_2$

- But this brings problems
  - Computational complexity (a lot more parameters to learn, transformation operations)
  - Overfitting
- SVM algorithm deals with these (max. margin & SV, kernel trick & SV)

SVM - max. margin



- Model (linear, *hyperplane*) for separation of data by using the *maximal margin* principle (MAX: robustness, SV: stability)

$$\overline{W} \cdot \overline{X} + b = 0$$
$$\overline{W} \cdot \overline{X_i} + b \geq 0 \quad \forall i : y_i = +1$$
$$\overline{W} \cdot \overline{X_i} + b \leq 0 \quad \forall i : y_i = -1$$

- Learning: maximal margin (optimal hyperplane) optimization problem
- Soft margin to allow misclassifications
    - Distance on the wrong side: ξ_i
    - Parameter *C* (misclassification cost) - set with experimentation!
    - Penalty: $C \cdot \xi^r_i$

SVM - kernel trick

- Use of higher dimensions for linearly non-separable data
  - https://www.youtube.com/watch?v=3liCbRZPrZA
- Learning (optimization) involves dot products in the term to maximize:

$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \overline{X_i} \cdot \overline{X_j}$$

Dot product of training data points is needed, (not feature values)

~similarity

classification too:

We can avoid representing W

$$F(\overline{Z}) = \text{sign}\{\overline{W} \cdot \overline{Z} + b\} = \text{sign}\{(\sum_{i=1}^{n} \lambda_i y_i \overline{X_i} \cdot \overline{Z}) + b\}$$

SVM - kernel trick, here it is

- We do not need the feature values, just dot products
- Transformation to another (higher dimensional) feature space would mean:

  $\Phi(x_i) \cdot \Phi(x_i)$

  calculation of transformations, then the lengthy dot products...

- Instead, we can use a function such that: $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_i)$
  - And $K(x_i, x_j)$ is in original space!
    - EXAMPLE !
  - We can only calculate kernels (polynomial, Gaussian RBF, ...)
  - Simetric, positive semi-definite; similarity ; even for strings, graphs
  - The mapping Φ can now be only implicitly used