Machine Learning- Exercise 3

SVM, AdaBoost

George Lydakis & Idil Esen Zulfikar

<lastname>@vision.rwth-aachen.de

RWTH Aachen University - Computer Vision Group

http://www.vision.rwth-aachen.de/

2022-12-08

Content

Content

1. Support Vector Machine

2. AdaBoost

Support Vector Machine - Recap

- The SVM tries to find a classifier which maximizes the margin between 2 classes.
- Up to now considered linear classifiers

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

  - Formulation as a complex optimization problem
  - Find the hyperplane satisfying

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \frac{1}{2}||\mathbf{W}||^2$$

  - under the constraints

$$t_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1, \forall n$$

  - based on training data points x_n and target value $t_n \in \{-1, 1\}$

Support Vector Machine - Recap

- Lagrangian primal form

$$\mathcal{L}_p = \frac{1}{2}||\mathbf{w}||^2 - \sum_{n=1}^{N} \alpha_n \{ t_n(\mathbf{w}^T \mathbf{x}_n + b) - 1 \}$$
$$= \frac{1}{2}||\mathbf{w}||^2 - \sum_{n=1}^{N} \alpha_n \{ t_n y(\mathbf{x}_n) - 1 \}$$

- The solution of $\mathcal{L}_p$ needs to fulfill the KKT conditions
  - Necessary and sufficient conditions

$$\begin{aligned} \alpha_n &\geq 0 & \lambda &\geq 0 \\ t_n y(\mathbf{x}_n) - 1 &\geq 0 & f(x) &\geq 0 \\ \alpha_n \{ t_n y(\mathbf{x}_n) - 1 \} &= 0 & \lambda f(x) &= 0 \end{aligned}$$

Support Vector Machine - Recap

- Solution for hyperplane
  - computed as linear combination of the training examples

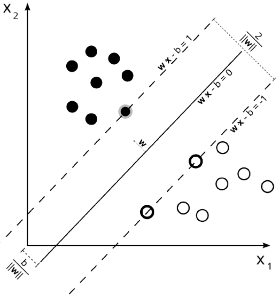
$$\mathbf{w} = \sum_{n-1}^{N} \alpha_n t_n \mathbf{x}_n$$

  - Suppose solution: $\alpha_n \neq 0$ only for some points, the support vectors
    - Only the SVs actually influence the decision Boundary!
  - Compute b by averaging over all support vectors:

$$b = \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} \alpha_m t_m \mathbf{x}_m^T \mathbf{x}_n \right)$$

Support Vector Machine - Recap

- The training points for which $\alpha_n \geq 0$ are called support vectors



- Graphical interpretation:
  - The support vectors are the points on the margin.
  - They define the margin and thus the hyperplane.
  - All other points can be discarded.

Support Vector Machine - Recap

- Primal form has time complexity of $\mathcal{O}(D^3)$ in D dimensions
- Dual form can be obtained by substituting weight vectors

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n t_n \mathbf{x}_n$$

$$\mathcal{L}_d(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m t_n t_m (\mathbf{x}_m^T \mathbf{x}_n)$$

- Under the conditions

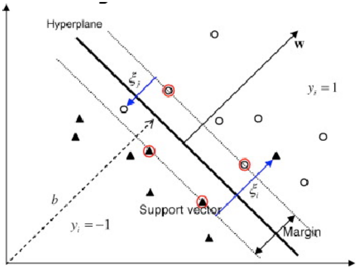
$$\alpha_n \geq 0 \quad \forall n$$

$$\sum_{n=1}^{N} \alpha_n t_n = 0$$

- time complexity between $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$

Support Vector Machine - Recap

- In case of overlapping class distribution, we allow some points to be miss-classified but with linear penalty that increases with the distance from the boundary.



Support Vector Machine - Recap

- Slack variables
  - $\xi \geq 0$ for each data point.
- Interpretation
  - $\xi_n = |t_n - y(\mathbf{x}_n)|$
  - $\xi_n = 0$ for points correctly classified or on the margin
  - $0 \leq \xi_n \leq 1$ points lie inside the margin on the correct side of the decision boundary
  - $y(x_n) = 0$ and $\xi = 1$ for the data point on the decision boundary
  - $\xi_n > 1$ for misclassified points
- The exact classification constraints are then replaced with

$$t_n y(\mathbf{x}) \geq 1 - \xi_n$$

Support Vector Machine - Recap

- Goal is to maximize margin while softly penalizing points that lie on the wrong side of the margin boundary.
- Therefore we minimize :

$$\mathcal{C} \sum_{n=1}^{N} \xi_n + \frac{1}{2} ||\mathbf{w}||^2$$

- Where the parameter $\mathcal{C}$ controls the trade-of between the slack variable penalty and the margin.
- In the limit of $\mathcal{C} \rightarrow \infty$ we will recover the earlier, fully linearly separable case (non overlapping class distribution).

Support Vector Machine - Recap

- The corresponding Lagrangian multiplier can be given:

$$\mathcal{L}(\mathbf{w}, b, \alpha, \xi) = \frac{1}{2}||\mathbf{w}||^2 + \mathcal{C}\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\alpha_n\left\{t_n y(\mathbf{x}_n) - 1 + \xi_n\right\} - \sum_{n=1}^{N}\mu_n\xi_n$$

- Where α_n and μ_n are Lagrangian multipliers
- The corresponding set of KKT conditions are given by:

$$\begin{aligned} \alpha_n &\geq 0 & \mu_n &\geq 0 \\ t_n y(\mathbf{x}_n) - 1 + \xi_n &\geq 0 & \xi_n &\geq 0 \\ \alpha_n\left\{t_n y(\mathbf{x}_n) - 1 + \xi_n\right\} &= 0 & \mu_n\xi_n &= 0 \end{aligned}$$

- We can optimize $\mathbf{w}, b, \{\xi_n \mid n \in \{1, \ldots, N\}\}$ as follows:

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{n=1}^{N}\alpha_n t_n x_n$$

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad 0 = \sum_{n=1}^{N}\alpha_n t_n$$

$$\frac{\partial L}{\partial \xi_n} = 0 \quad \Rightarrow \quad a_n = C - \mu_n$$

Support Vector Machine - Recap

- New SVM Dual: Maximize

$$\mathcal{L}_d(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m t_n t_m (\mathbf{x}_m^T \mathbf{x}_n)$$

- Under conditions

$$0 \leq \alpha_n \leq \mathcal{C}$$

$$\sum_{n=1}^{N} \alpha_n t_n = 0$$

- This is a quadratic programming problem

Support Vector Machine - Recap

- Using one of the KKT condition and result from partial derivation of Lagrange function we can derive condition for data points to be either support vector or slacks as below,
- We know that

$$\frac{\partial L}{\partial \xi_n} = 0 \qquad \Rightarrow \qquad \alpha_n = C - \mu_n$$

and

$$\mu_n \geq 0$$
$$\xi_n \geq 0$$
$$\mu_n \xi_n = 0$$

- For support vectors and slacks $0 \leq \alpha_n \leq \mathcal{C}$
- Moreover for slacks $\xi_n \geq 0$ which implies $\mu_n = 0$
- From the partial derivation shown above $\rightarrow \alpha_n = \mathcal{C}$

Content

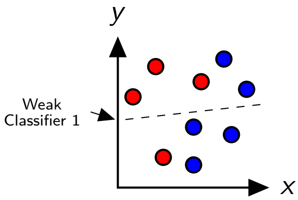
1. Support Vector Machine
2. AdaBoost

Adaboost[Freund & Schapire, 1996]-Recap

- Main idea
  - Instead of resampling, reweight misclassified training examples.
    - Increase the chance of being selected in a sampled training set.
    - Or increase the misclassification cost when training on the full set.
- Components
  - $c_k(\mathbf{x})$: "weak" or base classifier
    - Condition: $<50\%$ training error over any distribution
  - $C(\mathbf{x})$:"strong or final classifier
- Adaboost:
  - Construct a strong classifier as a thresholded linear combination of the weighted classifiers:

$$C(\mathbf{x}) = \operatorname{sign}\left(\sum_{k=1}^{K} \alpha_k c_k(\mathbf{x})\right)$$

Adaboost - Recap

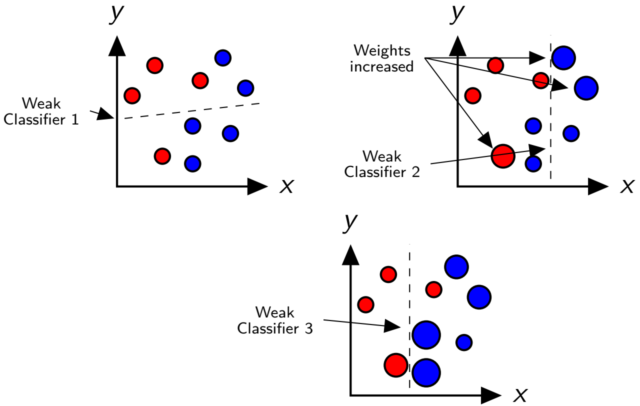


Consider a 2D feature space with positive and negative examples.

Each weak classifier splits the training examples with at least 50% accuracy.

Examples misclassified by a previous weak learner are given more emphasis at future rounds.

Adaboost - Recap



- Final classifier is combination of the weak classifiers.

Adaboost - Algorithm

- Initialization: Set $w_n^{(1)} = \frac{1}{N}$ for $n = 1, \ldots, N$.
- For $k = 1, \ldots, k$ iterations
  - Train a new weak classifier $c_k(\mathbf{X})$ using current weights $\mathbf{W}^{(k)}$ by minimizing the weighted error function
  - estimate the weighted error of this classifier on $\mathbf{X}$:

$$\epsilon_k = \frac{\sum_{n=1}^{N} w_n^{(k)} I(c_k(\mathbf{X}) \neq y_n)}{\sum_{n=1}^{N} w_n^{(k)}}$$

  - Calculate a weighting coefficient for $c_k(\mathbf{X})$:

$$\alpha_k = \ln\left\{\frac{1-\epsilon_k}{\epsilon_k}\right\}$$

  - Update the weighting coefficients:

$$w_n^{(k+1)} = w_n^{(k)} \exp\{\alpha_k I(c_k(\mathbf{X}) \neq y_n)\}$$