CLASSIFICATION II: SVM

Machine Learning for Autonomous Robots

Manuel Meder
Robotics Group, University of Bremen

November 29, 2022 – Bremen, Deutschland

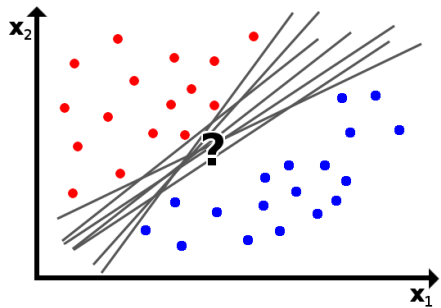
University of Bremen

German Research Center for Artificial Intelligence

Regression

Problem

How would and should we separate these data points for classification?



- We want to minimize possible future misclassification due to e.g. sensor noise

Idea

Separate two classes with largest distance (margin) between them.

- How do we get the distance of a point to the plane?

Hyperplane definition:

$$w^T x + b = 0 \text{ for all } x \text{ on the hyperplane}$$

Let x_n be the data point closest to this hyperplane, this means:

$$|w^T x_n + b| \geq 0$$

Scale invariance

The vector w describes the norm which is perpendicular to the plane.

$\Rightarrow$ This means by scaling w and b by the same factor λ describes still the same hyperplane.

Definition

To facilitate the maths later on we define that the distance to the closest data point shall be 1.

$$|w^T x_n + b| = 1$$

Calculate distance

Let x be a point on the hyperplane. The distance vector is then:

$$(x_n - x)$$

To get the distance of the hyperplane to x_n we project this distance vector onto the normalized vector of w.

$$\hat{w} = \frac{w}{||w||}$$

We can do projection by forming the inner product. Hence we get the distance d

$$d = < \hat{w}, (x_n - x) > = < \frac{w}{||w||}, (x_n - x) > = \frac{1}{||w||}(w^T x_n - w^T x)$$

Calculate distance

We introduce artificially the offset b again.

$$d = \frac{1}{||w||}(w^T x_n + b - w^T x - b)$$

- We notice that we have the formula for a point on the hyperplane which collapses to zero by definition.
- We also have the formula for the closest point to the hyperplane which is by definition 1.

$$d = \frac{1}{||w||}(\underbrace{w^T x_n + b}_{1} - \underbrace{(w^T x + b)}_{0}) = \frac{1}{||w||}$$

Class labeling

Once the hyperplane is defined, we can use a simple decision function for class labelling:

Decision function

$$d(\mathbf{x}) = \text{sign}(w^T\mathbf{x} + b)$$

Implication of defining the margin

With the margin being defined by the nearest points, this has some implications on other data points:

Implications

$$\forall(\mathbf{x}, y_1)\ \mathbf{w} \cdot \mathbf{x} + b \geq +1$$

$$\forall(\mathbf{x}, y_2)\ \mathbf{w} \cdot \mathbf{x} + b \leq -1$$

"Combined" representation

$$y_i(\mathbf{w} \cdot \mathbf{x} + b) \geq 1\ \forall i$$

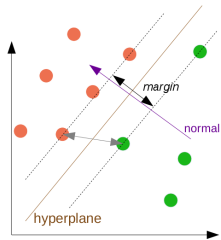
The margin

So, using the nearest point to the margin, define the size of the margin m as:

Margin

$$|m| = 2 * d = \frac{2}{||\mathbf{w}||}$$

Those not interacting with the margin are called inner points and do not contribute to the hyperplane.



Maximizing the margin

Goal

(For better generalization) find a parameter set of $\mathbf{w}, b$ for the decision hyperplane that maximizes the margin $\frac{2}{||w||}$.

Equivalent optimization problem

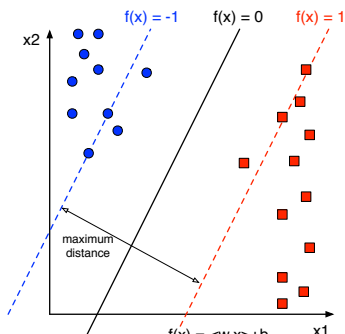
Minimize (the norm) $||\mathbf{w}||$ subject to $y_i(\mathbf{w} \cdot \mathbf{x} + b) \geq 1$

Note: For reasons of computability $||\mathbf{w}||$ is replaced by $J(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2$. This can be solved using *quadratic programming* and does not affect the optimization problem.

SVM

Support Vector Machine - Softmargin

The world is not perfect and data definitely is not. Therefore a hard decision boundary might not always be beneficial.



Support Vector Machine - Optimization problem

We also want to allow violations of the maximum margin, but we will penalize them.

$$\begin{aligned} \min_{w,b} \quad & \tfrac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i & & \\ \text{s.t.} \quad & |(\langle w, x_i\rangle + b)| & \geq 1-\xi_i \quad & \forall 1 \leq i \leq n \\ & \xi_i & \geq 0 \quad & \forall 1 \leq i \leq n \end{aligned}$$

As a constraint the *absolute value* is not so nice but we can easily solve this by multiplying with the labels.

$$\begin{aligned} \min_{w,b} \quad & |-x|\tfrac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i & & \\ \text{s.t.} \quad & y_i(\langle w, x_i\rangle + b) & \geq 1-\xi_i \quad & \forall 1 \leq i \leq n \\ & \xi_i & \geq 0 \quad & \forall 1 \leq i \leq n \end{aligned}$$

Lagrangian of the optimization problem

In order to include the contraints into the optimization problem we have to reformulate them in such a way that they are greater or equals to zero. **Constraints**:

$$\begin{aligned} y_i(\langle w, x_i \rangle + b) - 1 + \xi_i &\geq 0 \quad \forall 1 \leq i \leq n \\ \xi_i &\geq 0 \quad \forall 1 \leq i \leq n \end{aligned}$$

If we have constraints in such a form according to Karush-Kuhn-Tucker (KKT) we can subtract those terms from the optimization problem by multiplying them with Lagrangian multipliers.

Lagrangian of the optimization problem

Hence we get the following optimization problem:

$$\min_{w,b,\xi} \mathcal{L}(w, b, \xi, \alpha, \beta) = \min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i$$
$$- \sum_{i=1}^{n} \alpha_i (y_i (w^T x_i + b) - 1 + \xi_i) - \sum_{i=1}^{n} \beta_i \xi_i$$

and maximize with respect to α, β.

To solve this and get rid of w, ξ we calculate the partial derivatives of the Lagrangian.

SVM: Formal definition

We get the following derivatives:

$$\nabla_w \mathcal{L} = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0$$

$$\Rightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \qquad \forall i$$

These equations we can put back into the original Lagrangian.

SVM: Formal definition

We get the following optimization problem:

$$\min_{w,b,\xi} \mathcal{L}(w, b, \xi, \alpha, \beta) = \min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i$$
$$- \sum_{i=1}^{n} \alpha_i (y_i (w^T x_i + b) - 1 + \xi_i) - \sum_{i=1}^{n} \beta_i \xi_i$$

Since $C - \alpha_i - \beta_i = 0$ those terms collapse.

$$\min_{w,b,\xi} \mathcal{L}(w, b, \xi, \alpha, \beta) = \min_{w,b,\xi} \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i (y_i (w^T x_i + b) - 1)$$
$$= \min_{w,b,\xi} \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i y_i (w^T x_i + b) + \sum_{i=1}^{n} \alpha_i$$

SVM: Formal definition

From the derivation we have found:

$$-\sum_{i=1}^{n} \alpha_i y_i = 0$$

Plugging this into the optimization problem we get:

$$\min_{w,b,\xi} \mathcal{L}(w, b, \xi, \alpha, \beta) = \min_{w,b,\xi} \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i y_i (w^T x_i + b) + \sum_{i=1}^{n} \alpha_i$$

$$= \min_{w,b,\xi} \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i y_i (w^T x_i) + \sum_{i=1}^{n} \alpha_i$$

SVM: Formal definition

From the derivation we also have found:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

Plugging this into the optimization problem we get:

$$\min_{w,b,\xi} \mathcal{L}(w, b, \xi, \alpha, \beta) = \min_{w,b,\xi} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i x_j - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i x_j + \sum_{i=1}^{n} \alpha_i$$

Which collapses to:

$$\min_{w,b,\xi} \mathcal{L}(w, b, \xi, \alpha, \beta) = \min_{w,b,\xi} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i x_j$$

Primal and dual optimization problem

Primal optimization problem (Lagrange multipliers α, β)

$$\mathcal{L}(w, b, \alpha, \xi, \beta) = \frac{1}{2}||w||^2 + C\sum_i \xi_i - \sum_i \alpha_i \left[y_i(w^T x_i + b) - 1 + \xi_i\right] - \sum_i \beta_i \xi_i$$

Known dual representation of optimization problem

$$\mathcal{L}'(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j < x_i, x_j >$$

subject to (box constraints):

$$0 \leq \alpha_i \leq C \tag{1}$$

$$\sum_i \alpha_i y_i = 0 \tag{2}$$

Support Vectors (SV)

All the α_i's are chosen to satisfy the following condition:

$$\alpha_i \left[y_i(w^T x_i + b) - 1 + \xi_i \right] = 0$$

So either α_i or the term in the bracket is zero.

- is α_i zero then x_i is an inner point
- is α_i non-zero then $y_i(w^T x_i + b) - 1 + \xi_i$ holds and x_i is a support vector
- there are two kinds of support vectors (SV):
  - margin SV: is located on the maximal margin lines ($\xi = 0$)
  - non-margin SV: violates the maximum margin ($\xi > 0$)

The resulting model

From the derivation of the optimization problem we get the following term:

$$w = \sum_i \alpha_i y_i x_i$$

Using this we can plug in a support vector in order to get the offset b, as all support vectors have to satisfy:

$$y_i\big(\langle w, x_i\rangle + b\big) = 1 - \xi_i$$

Non linear classification

In order to solve the dual form of the optimization problem we needed to have the inner product of the data points.

So far, we have used SVMs with a linear kernel for that, i.e.

$$k(\mathbf{x}, \mathbf{x}') = < \mathbf{x}, \mathbf{x}' >$$

To allow non-linear classification the *kernel trick* can be applied, and this linear kernel function can be replaced by a non-linear one.

Kernel Trick

Motivation

Higher feature space dimensionality simplifies the search of a linear solution (in that higher dimensional space) for our classification problem. But the data transformation into that space comes with a certain cost.

We had to . . . :

- choose a suitable value of our hyper-parameter for ϕ
- transfer the data using ϕ
- calculate the inner product in the higher feature space
- calculate the inverse of the resulting matrix

Drawbacks:

- Choice of ϕ limits the function class used for classification to the chosen function
- Finding a suitable hyper-parameter we hade to use CV (costly)
- Tranformation of data costs calculation power

Note

To calculate the solution for our weights (model) we only needed to calculate the inner product of transformed data points.

Idea

If we somehow could calculate the inner product of our transformed data and get the solution without actually transforming the data to that higher dimension, we could benefit of the higher dimension while not paying for the costly transformation.

Kernel-Trick

There exist functions which calculate a value that is the inner product of the same data in higher dimensional space directly without transforming the data.

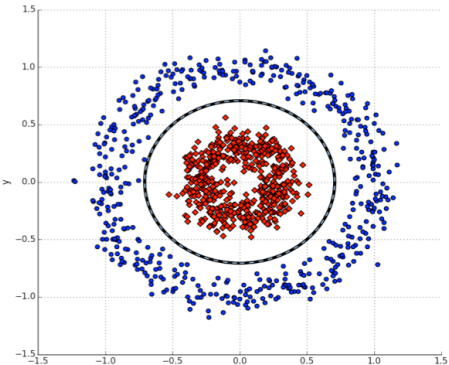
Examples

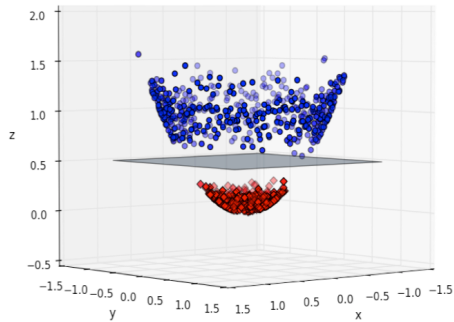
- $K(x, x') = x^T x' + c$
- $K(x, x') = (\alpha x^T x' + c)^d$ (polynomial kernel)
- $K(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$ (Gaussian- RBF kernel)
  $K(x, x') = \exp\left(-\gamma\|x - x'\|^2\right)$ with $\gamma = \frac{1}{2\sigma^2}$
- $K(x, x') = \exp\left(-\frac{\|x-x'\|}{\sigma}\right)$ (Laplacian kernel)
- Fisher, ANOVA, Sigmoid, Power, Wave, Log, Spline, String - kernel
- your own kernel

Kernel properties

Mercer's theorem

$K(x, x') =< \phi(x), \phi(x') >$ if and only if K (gram matrix) is symmetric and positive semidefinite.





http://beta.cambridgespark.com/courses/jpm/figures/mod5_kernel_trick.png

Classification for multiple classes

SVM do binary classification only. Various approaches have been developed to classify a number of $k > 2$ classes with SVM. Some are based on:

- a heuristic, that uses k-SVMs in a one-class-versus-the-rest approach
  suffers from an asymmetry of the training data: $(k-1)/k$ vs. $1/k$
- using a single objective function
  train all k-SVM simultaneously, resulting is slow training due to the higher complexity of the optimization problem
- one-versus-one approach: train $k(k-1)/2$ SVMs and use the class that 'wins' most often
  can be facilitated using directed acyclic graphs to reduce the number of classifiers to test on

Evaluation

Pro:

- 'easy' basic concept
- good generalization
- decision surface, i.e. hypothesis has explicit dependence on data
- optimization of a convex function, i.e. no dealing with local minima
- few parameters (with linear kernel)
- confidence measures can be included
- relatively fast in delivering the classifier and the classification
- general application as Kernel machine
- can handle high dimensional data

Evaluation

Con:

- only directly applicable for binary classification
- does provide a decision, but not a likelihood
- sensitivity to asymmetric class distributions
- numerous variants of SVMs using different kernels
- parameters not trivial to optimize
- usually not online feasable - there exist online variants

Online Learning

Online Learning - Solving the SVM Dual

Dual representation of the SVM optimization problem

$$\min \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

subject to:

$$0 \leq \alpha_i \leq C$$

$$\sum_i \alpha_i y_i = 0$$

Where is the problem?

$$sum_i \alpha_i y_i = 0 \tag{3}$$

The Trick: Offset Tweaking

Before:

$$\begin{aligned} \min_{w,b,\xi} \quad & \tfrac{1}{2}||w||_2^2 + C\sum \xi_i & & \\ \text{s.t.} \quad & y_i(\langle w, x_i\rangle + b) & \geq 1-\xi_i & \quad \forall 1 \leq i \leq n \\ & \xi_i & \geq 0 & \quad \forall 1 \leq i \leq n. \end{aligned}$$

After:

$$\begin{aligned} \min_{w,b,\xi} \quad & \tfrac{1}{2}||(w,b)||_2^2 + C\sum \xi_i & & \\ \text{s.t.} \quad & y_i\langle (w,b),(x_i,1)\rangle & \geq 1-\xi_i & \quad \forall 1 \leq i \leq n \\ & \xi_i & \geq 0 & \quad \forall 1 \leq i \leq n. \end{aligned}$$

The Benefit

$$
\begin{aligned}
\min_{w,b,\xi} \quad & \tfrac{1}{2}||(w,b)||_2^2 + C\sum \xi_i & & \\
\text{s.t.} \quad & y_i\langle (w,b),(x_i,1)\rangle & \geq 1-\xi_i & \quad \forall 1 \leq i \leq n \\
& \xi_i & \geq 0 & \quad \forall 1 \leq i \leq n.
\end{aligned}
$$

Dual of the modified SVM optimization problem

$$\min \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

subject to: $0 \leq \alpha_i \leq C$

Solution algorithm

Dual of the modified SVM optimization problem

$$\max \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j k(x_i, x_j) - \sum_i \alpha_i$$

subject to: $0 \leq \alpha_i \leq C$

Direction Search:

1. start with $\alpha = 0$
2. iterate over all indices
   $\rightarrow$ for each index i determine optimal α_i and update it
3. To many iterations? $\rightarrow$ STOP
4. To small change in the loop? $\rightarrow$ STOP
5. GOTO 2

Passive-Aggressive Algorithm

INPUT: aggressive parameter $C > 0$
INITIALIZE: $w_1 = (0, \ldots, 0)$
For $t = 1, 2, \ldots$

- receive instance: $x_t \in \mathbb{R}^m$
- predict: $\hat{y}_t = \text{sign} \langle w_t, x_t \rangle$
- receive correct label: $y_t \in \{-1, +1\}$
- suffer loss: $l = \max\{0, 1 - y_t \langle w_t, x_t \rangle\}$
- update:
  1. set: determine α_t
  2. update: $w_{t+1} = w_t + \alpha_t y_t x_t$

FINISHED

$$\alpha_t = \frac{l_t}{||x_t||^2} \qquad \text{(PA)}$$

$$\alpha_t = \min\{C, \frac{l_t}{||x_t||^2}\} \qquad \text{(PA-I)}$$

$$\alpha_t = \frac{l_t}{||x_t||^2 + \frac{1}{2C}} \qquad \text{(PA-II)}$$

different losses:
hard margin (PA), **hinge loss** (PA-I, online SVM), squared hinge loss (PA-II)

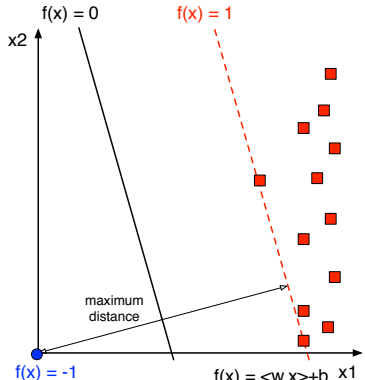
Outlier Detection

Why unary classification?

- data description
- novelty detection
- outlier detection
- one-vs-rest classifier in multi-class scenario

How?

Origin Separation



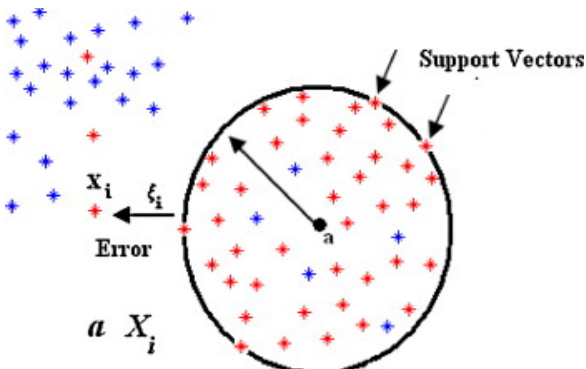
One-Class Support Vector Machine (simplified)

$$\begin{aligned} \min_{w,\xi} \quad & \tfrac{1}{2}||w||_2^2 + C\sum \xi_i \\ \text{s.t.} \quad & \langle w, x_i \rangle \geq 2 - \xi_i \\ \text{and} \quad & \xi_i \geq 0 \ \forall i \ . \end{aligned}$$

Publication:

Support Vector method for novelty detection by Schölkopf et al. (2000)

Support Vector Data Description



$$\begin{aligned} \min_{R,a,\xi} \quad & R^2 + C \sum \xi_i \\ \text{s.t.} \quad & ||a - x_i||_2^2 \leq R^2 + \xi_i \\ \text{and} \quad & \xi_i \geq 0 \ \forall i \end{aligned}$$

Publication:
Support Vector Data Description by David MJ Tax, Robert PW Duin (2004)

Conclusion

Wrap-up

Classifiers:

1. nearest neighbor
2. naive Bayes
3. logistic regression
4. decision trees
5. support vector machines
6. online passive aggressive algorithms
7. one-class SVM (zero separation)
8. support vector data description

Thank You!
Please feel free to ask questions in the forums.