Regularization in Machine Learning

Simone Filice
filice.simone@gmail.com

University of Roma Tor Vergata

Motivations

- Is there a direct way to prevent overfitting?
- Among a set of hyphotesis that correctly fit training data, which one should be used?

Overview

- Overfitting
- Regularization
- A concrete example: SVM

Overview

- **Overfitting**
- Regularization
- A concrete example: SVM

Overfitting

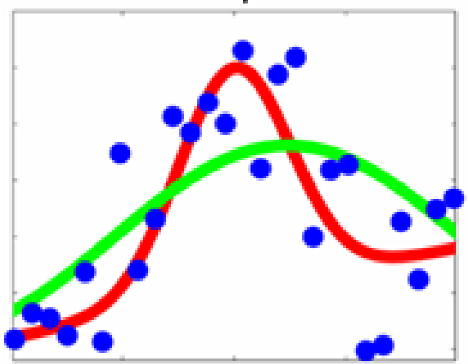
- *Overfitting*: the model perfectly fits training data but is too complex and does not generalize well on new data

- Reason: Usually a ML algorihm must solve an ill-posed problem. The hypothesis space is very complex (the learning algorithm has a <u>large number of parameters</u>) and the <u>available training data</u> does not sufficiently constraint the learning process in searching the optimal hypothesis, then multiple solutions can exist

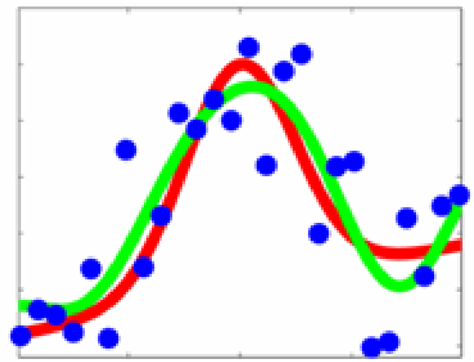
Overfitting in Regression Tasks

BIAS PROBLEM:

Learned function
with too simple model

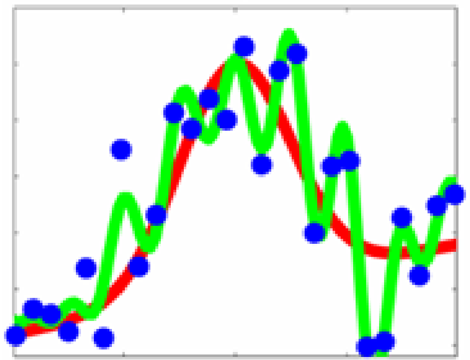


Learned function
with appropriate model

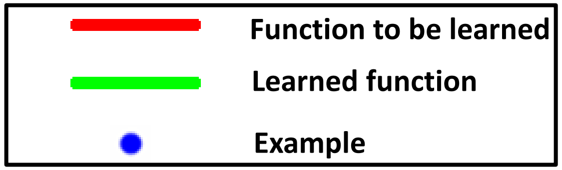


VARIANCE PROBLEM:

Learned function
with too complex model

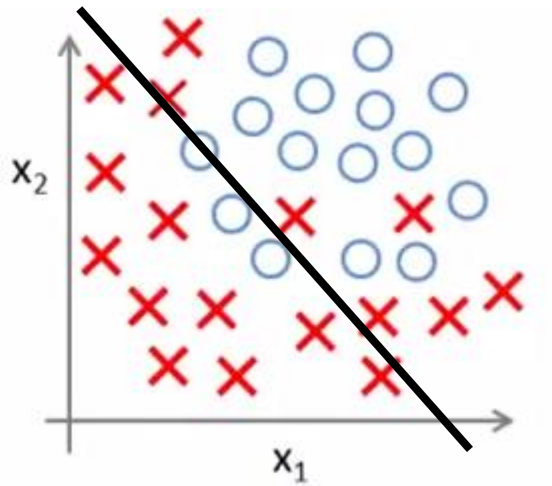


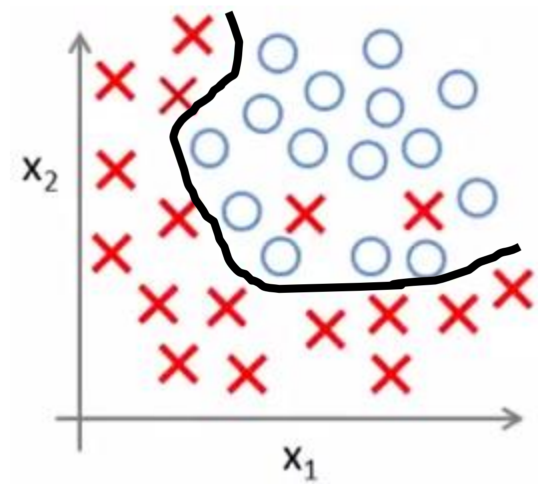
Overfitting!!!



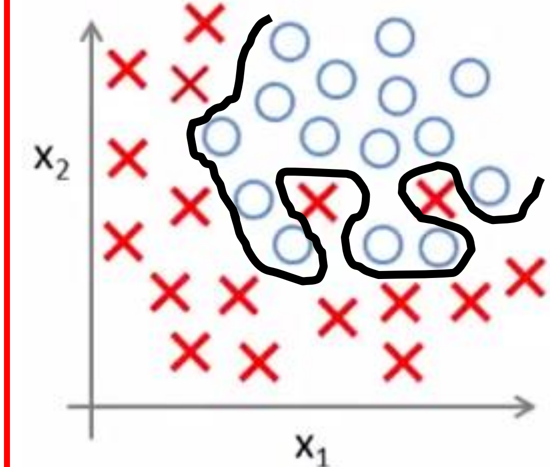
Overfitting in Classification Tasks

BIAS PROBLEM:





VARIANCE PROBLEM:



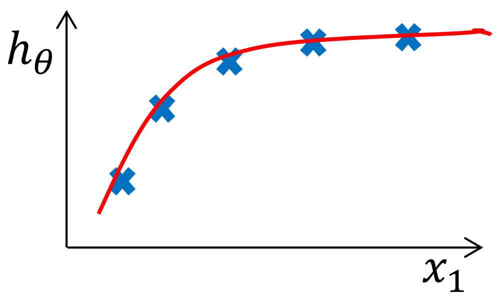
Overfitting!!!

Overview

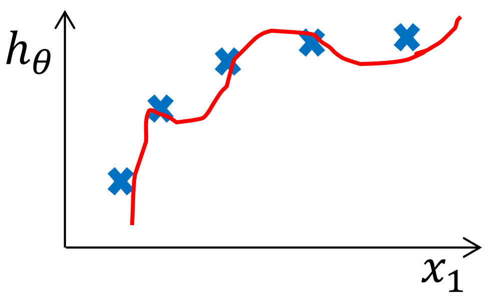
- Overfitting
- **Regularization**
- A concrete example: SVM

Regularization: an Intuition

- *Occam's Razor*: among the set of hypothesis that correctly explain a phenomenon, the simplest one should be chosen



$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 {x_1}^2$$



$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \ \ldots + \theta_{10} {x_1}^{10}$$

- Regression Example:
  - we want to estimate a 2nd order polynomial using a higher order polynomial
  - A lot of high order polynomials can fit the training data and among them there is the 2nd order correct one
  - Penalizing $\theta_3 \ldots \theta_{10}$ the learning algorithm will keep those parameters small, and the correct hypothesis will be selected

Regularization

- In general a learning algorithm has a lot of parameters and we do not know a priori which ones should be penalized

- For instance, in a standard NLP task each word in the vocabulary is a different feature, then, adopting a linear model, the number of parameters will be the vocabulary size

- A regularizer operating over all the parameters θ_i must be added to the learning optimization function

Regularizer

- A regularizer is a function that penalizes large θ_i in order to keep the solution as simple as possible
- A regularizer helps the learning algorithm to choose more generic solutions, preventing overfitting
- Norms are the most common regularizers:
- 1-norm: $\|\theta\|_1 = \sum_i |\theta_i|$ — Favourite sparser solutions
- 2-norm: $\|\theta\|_2 = \sqrt{\sum_i {\theta_i}^2}$
- P-norm: $\|\theta\|_p = (\sum_i |\theta_i|^p)^{\frac{1}{p}}$ — Discourage large absolute values

Regularization in Optimization Problems

- A lot of Machine Learning Algorithms solve an optimization problem of the following form:

$$\theta^* = arg\min_{\theta \in \Theta}(\lambda \cdot L(\theta, D) + R(\theta))$$

Where:

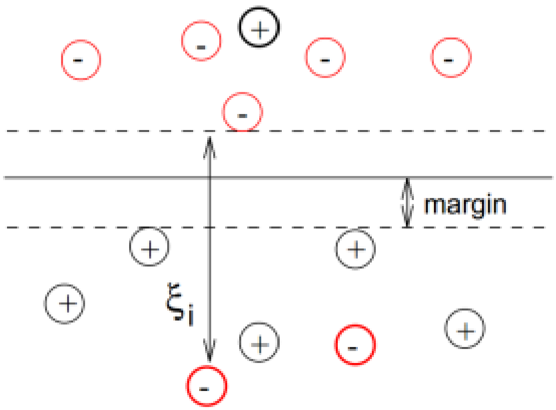
- D is the available training dataset
- θ^* is the hypothesis that optimize the cost function
- Θ is the hypothesis space
- L is a the Empirical Risk (evaluates the training error)
- R is the regularizer
- λ is a coefficient that weights the contribution of L and R in the cost function

Overview

- Overfitting
- Regularization
- **A concrete example: SVM**

Support Vector Machines

$$\theta^* = \underset{\theta\in\Theta}{argmin}\left(R(\theta) + \lambda\cdot L(\theta, D)\right)$$



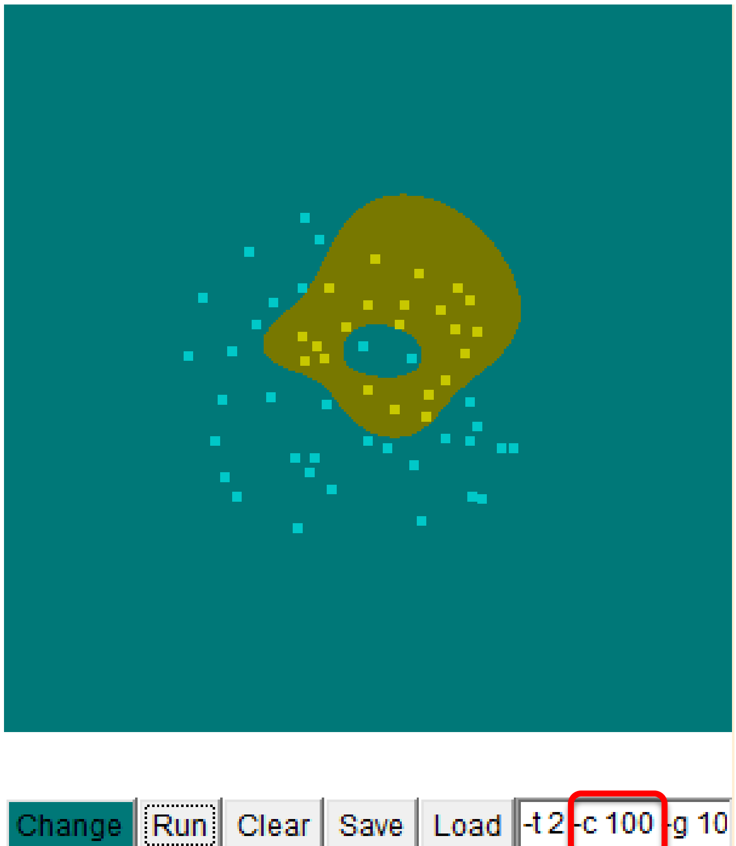
$$w^*, b^* = \underset{w,b}{argmin}\left(\frac{1}{2}\|w\|_2 + C\cdot\sum_{i=1}^{m} \xi_i\right)$$

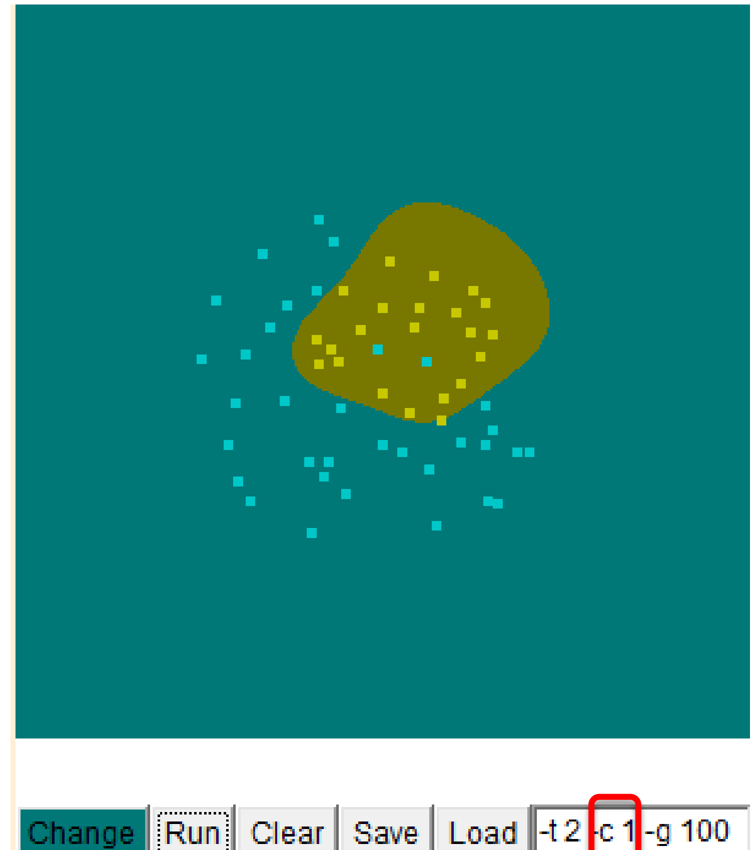
such that for all $1\le i \le m$: $y_i(w\cdot x_i + b) \ge 1 - \xi_i\,,\quad \xi_i \ge 0$

$$=$$

$$\sum_{i=1}^{m} \underbrace{l(w,b,x_i,y_i)}_{\text{Hinge loss}} = \max(0, 1 - y_i(w\cdot x_i + b))$$

Regolarization Effect on SVMs





http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Summary

- *Generalization* capability can be achieved via *Simplicity*

- **Regularization** keeps the model simple, and helps in preventing *overfitting*

- A lot of ML algorithms include a *regularizer* in their formulation