Model Non-linearization, Overfitting & Regularization

DCS310

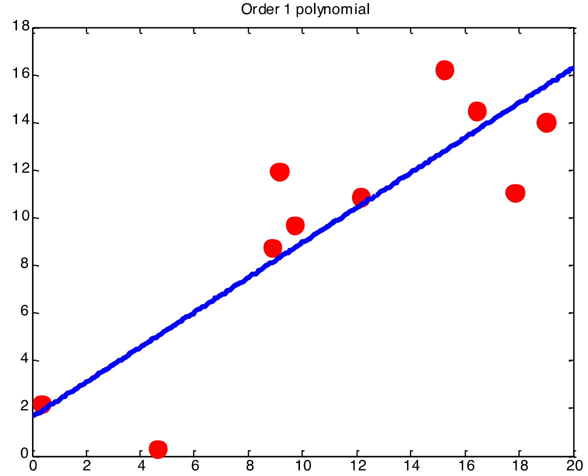
Sun Yat-sen University

Outline

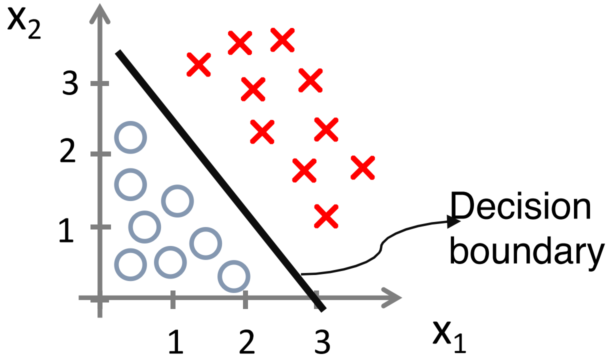
- Model Non-linearization
- Overfitting
- Model Selection
- Regularization

Introduction

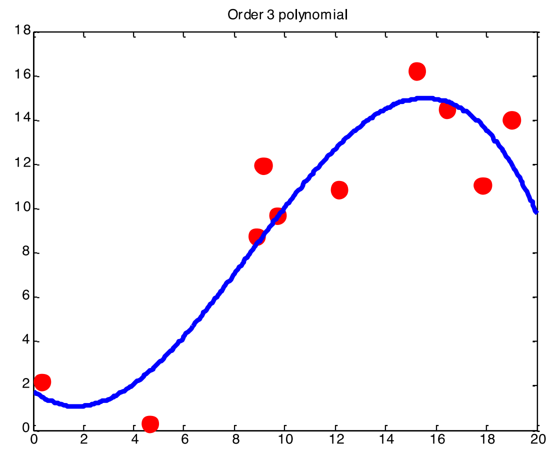
- Only linear relation between input $\boldsymbol{x}$ and output y can be modelled in linear regression



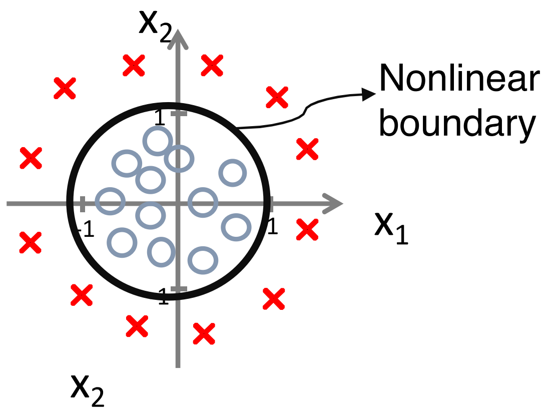
- For linear classifiers, the decision boundaries can only be linear



- For more complex applications, models should be able to handle
  - nonlinear input-output relation



  - nonlinear decision boundaries



How to obtain models with nonlinear representation ability ?

Basic idea: non-linearizing the linear models with basis functions

$$[x] \longrightarrow [x, x^2, x^3]$$

Non-linearization via Basis Functions

- Transform the features by polynomial

$$[x] \longrightarrow [x, x^2, x^3]$$

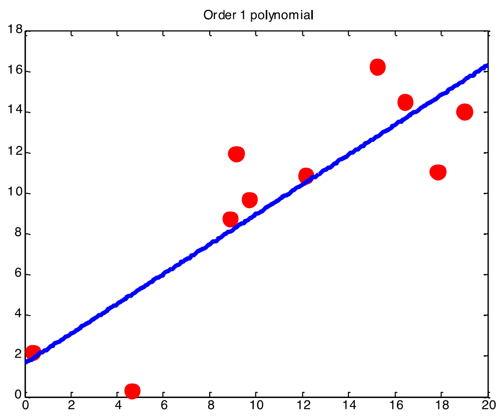
Single feature is expanded into 3 features

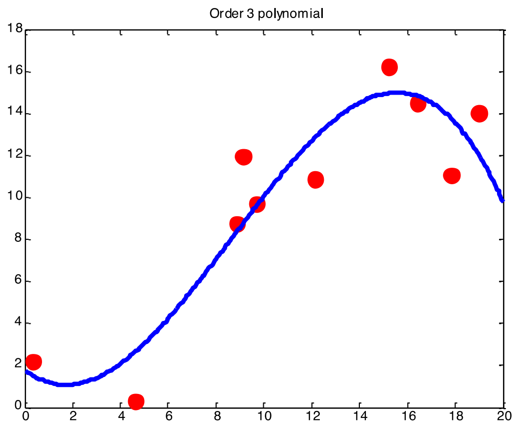
- Model with original feature

$$f(x) = w_0 + w_1 x$$
$$= [1, x]\boldsymbol{w}$$

- Model with expanded features

$$f(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$
$$= \boldsymbol{\phi}(x)\boldsymbol{w}$$





- Generally, the transformation could be expressed as

$$[x_1, x_2, \cdots, x_m] \in \mathbb{R}^m \longrightarrow [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \cdots, \phi_n(\boldsymbol{x})] \in \mathbb{R}^n \triangleq \boldsymbol{\phi}(\boldsymbol{x})$$

$\phi_k(\boldsymbol{x})$ could be any functions that produce useful features, *e.g.*,

$$\sqrt{x}, \quad \log x, \quad \frac{1}{x}, \quad x_1 + x_2, \quad x_1 - x_2, \quad x_1 x_2$$

- The non-linearized model now becomes

$$f(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{w}$$

which is called basis function model

The basis function model is nonlinear *w.r.t.* $\boldsymbol{x}$, but is *still linear w.r.t. the model parameters* $\boldsymbol{w}$

- With the nonlinearly transformed feature $\boldsymbol{\phi}(\boldsymbol{x})$, the optimal model parameters $\boldsymbol{w}^*$ for regression is obtained by optimizing the loss

$$L(\boldsymbol{w}) = \frac{1}{N} \|\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{w} - \boldsymbol{y}\|^2$$

where $\boldsymbol{\Phi}(\boldsymbol{X}) \triangleq \begin{bmatrix} \boldsymbol{\phi}\left(\boldsymbol{x}^{(1)}\right) \\ \vdots \\ \boldsymbol{\phi}\left(\boldsymbol{x}^{(N)}\right) \end{bmatrix}$

- With the notation $\boldsymbol{\Phi} = \boldsymbol{\Phi}(\boldsymbol{X})$, the optimal model parameters $\boldsymbol{w}^*$ is

$$\boldsymbol{w}^* = (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{y}$$

The same as linear regression *except that* $\boldsymbol{X}$ *is replaced by* $\boldsymbol{\Phi}$

- We can also employ the numerical methods, *e.g.* gradient descent, to obtain the optimal solution

- For the classification using the basis functions, the cross-entropy loss becomes

$$L(\boldsymbol{W}) = -\frac{1}{N}\sum_{\ell=1}^{N}\sum_{k=1}^{K} y_k^{(\ell)} \log softmax_k\left(\boldsymbol{\phi}\left(\boldsymbol{x}^{(\ell)}\right)\boldsymbol{W}\right)$$

The optimal $\boldsymbol{W}^*$ can only be obtained by numerical methods

- Denoting $\boldsymbol{\phi}\left(\boldsymbol{x}^{(\ell)}\right)$ as $\boldsymbol{\phi}^{(\ell)}$, the gradient can be derived equal to

$$\frac{\partial L(\boldsymbol{W})}{\partial \boldsymbol{w}_j} = \frac{1}{N}\sum_{\ell=1}^{N}\left(softmax_j\left(\boldsymbol{\phi}^{(\ell)}\boldsymbol{W}\right) - y_j^{(\ell)}\right)\boldsymbol{\phi}^{(\ell)T}$$

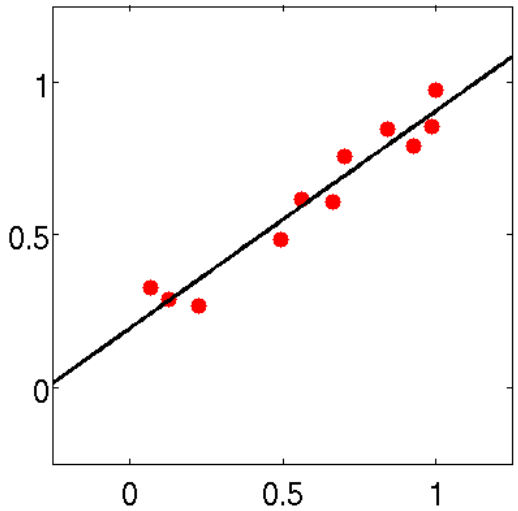
The same as multi-class logistic regression *except that $\boldsymbol{x}^{(\ell)}$ is replaced by $\boldsymbol{\phi}^{(\ell)}$*

Outline

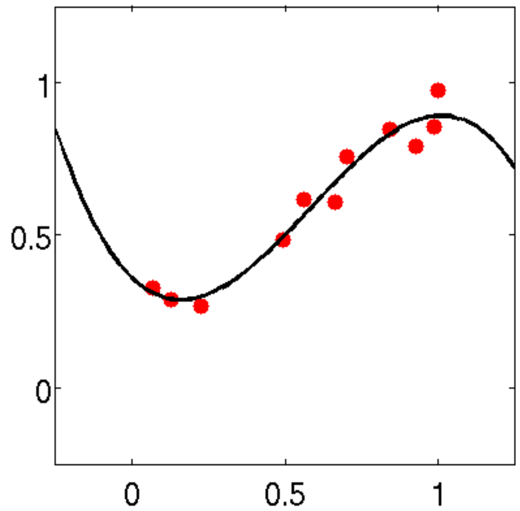
- Model Non-linearization
- Overfitting
- Model Selection
- Regularization

Overfitting

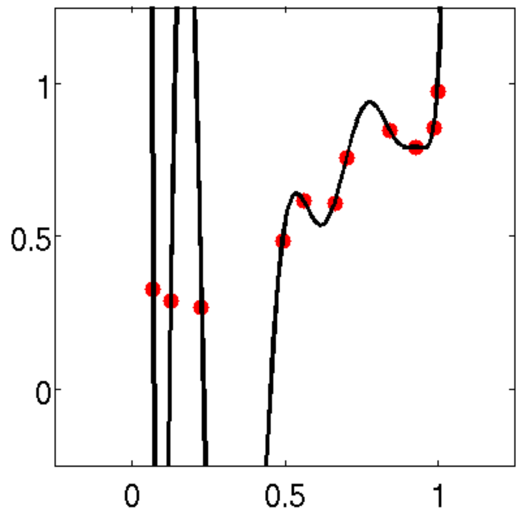
- Higher-dimensional features $\boldsymbol{\phi}(\boldsymbol{x})$ leads to better fitness on the *training data*



1-order



3-order

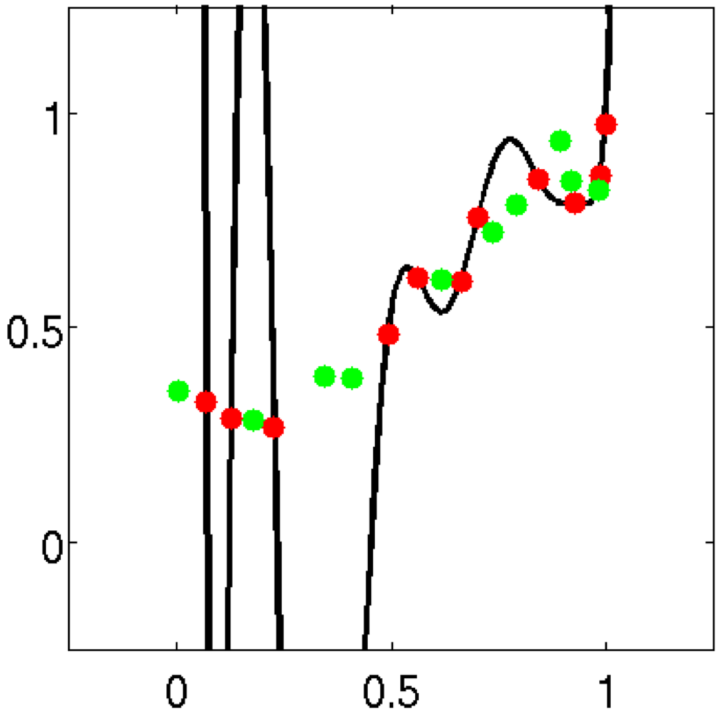


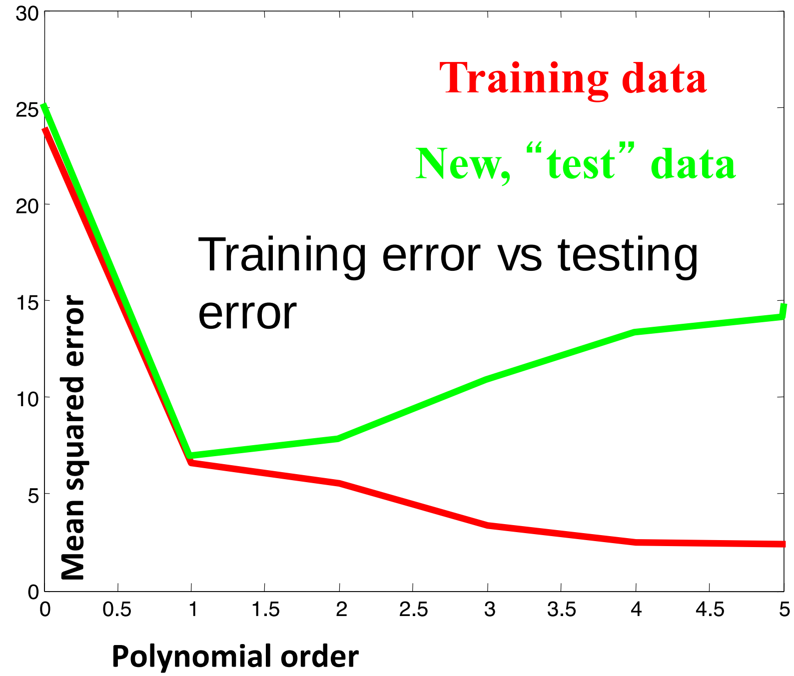
5-order

Which model is better??

- From the viewpoint of fitting the training data, of course, the higher the model order is, the better the fitting looks

- But high-order models may *perform poor on the testing data*





The ability that a model can perform well on unseen data is called the *generalization ability of the model*

Model Complexity

- Each model corresponds to a degree of complexity

- But it is difficult to give an exact expression to describe the model complexity

- In general, the model complexity depends on the number of parameters, the more parameters, the more complex the model is

- To have the model perform well, we should *balance between the model complexity and its representational ability*

Outline

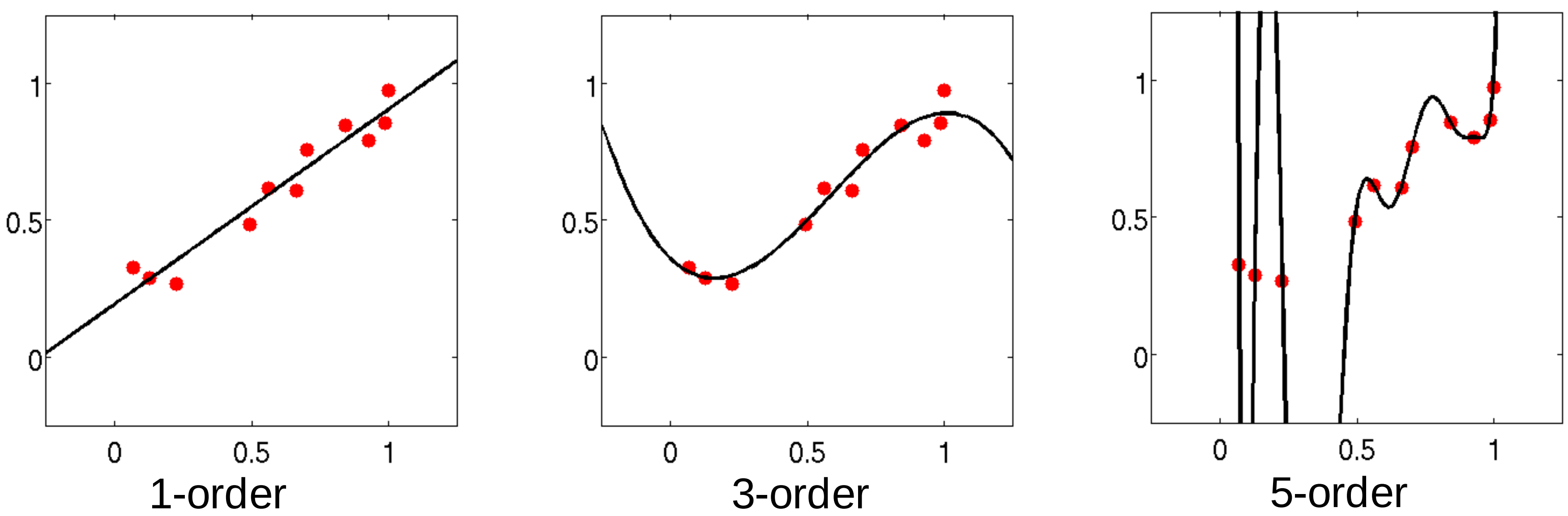
- Model Non-linearization
- Overfitting
- Model Selection
- Regularization

Model Selection

- Model selection: Given a set of models $\{\mathcal{M}_1, \cdots, \mathcal{M}_m\}$, choose the one that can *perform best on the* ***unseen testing data***

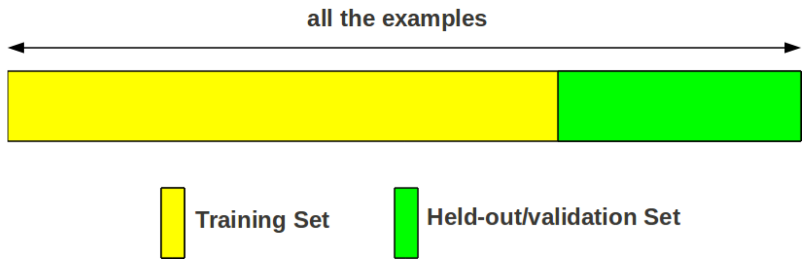
Model candidates could be of the same type, or different types

Cannot select the model based their performance on training data



Validation Set

- Set aside a portion (20% ~ 30%) of training data as the validation set, and use the remaining as the training data



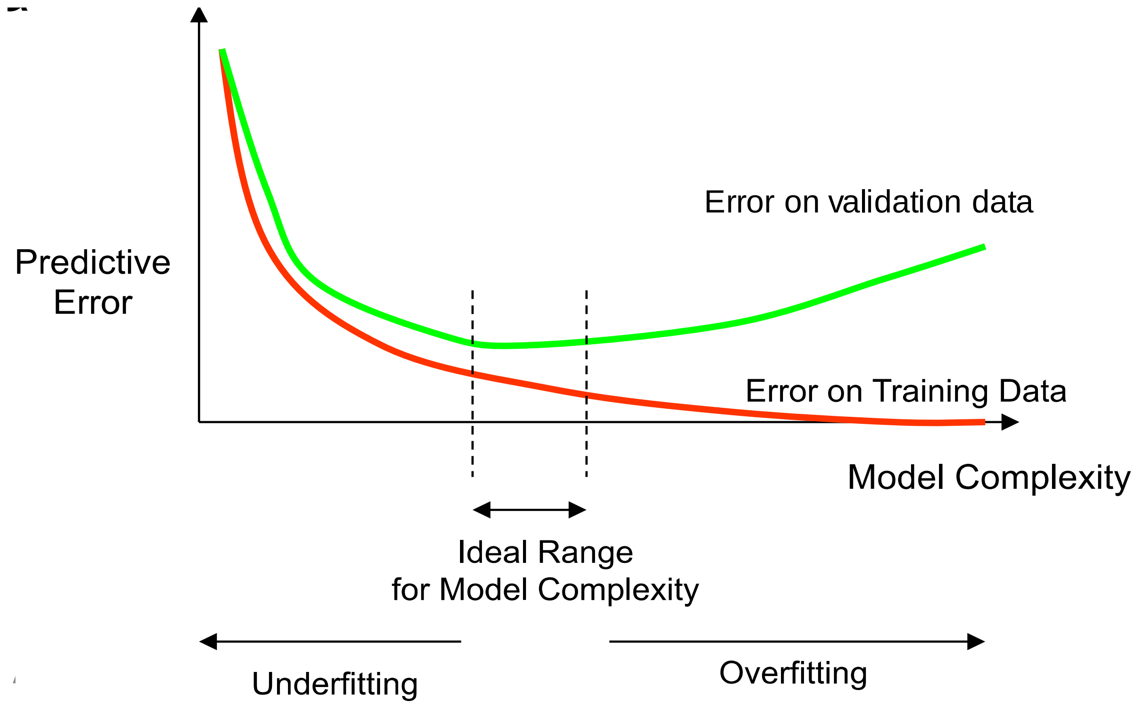
Both the training and validation set *cannot include testing examples*

The validation set cannot be too small. Why??

- Train the model on the training set, while evaluating the model on the held-out validation set

- Choose the model with the best performance on the validation set

- The prediction error on the training and validation datasets



- If the validation error decreases as the model complexity grows, it suggests the model is *under-fitting*

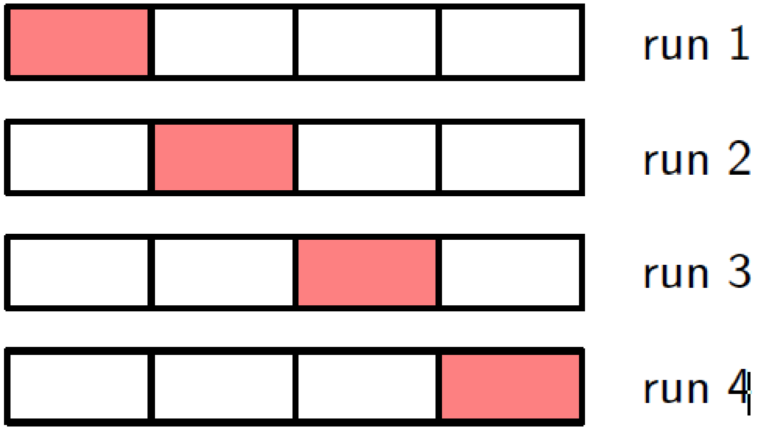
- Otherwise, it implies the model is *overfitting*

Cross-Validation

- Issue with the ordinary validation method

  The training data is often scarce. If a large portion is set aside for validation, no sufficient training data can be used

- A compromise solution: K-fold cross-validation
  - Partition the whole training dataset into K subsets equally
  - Train on the $(K-1)$ subsets, evaluate on the remaining subset
  - Repeat the above step for K times, each using a different subset for validation



Information Criteria

- Akaike Information Criterion (AIC)

$$AIC = 2M - 2\log(\mathcal{L})$$

  - *M is the number of parameters*
  - *$\mathcal{L}$ is the likelihood*

- Bayesian Information Criterion (BIC)

$$BIC = M\log N - 2\log(\mathcal{L})$$

  - *N is the number of training data examples*

These criteria *can only be used in the probabilistic models due to the requirement of log-likelihood* $\log(\mathcal{L})$

Outline

- Model Non-linearization
- Overfitting
- Model Selection
- Regularization

- Imposing some prior preferences on the parameters, in addition to fitting the training data, *e.g.*,

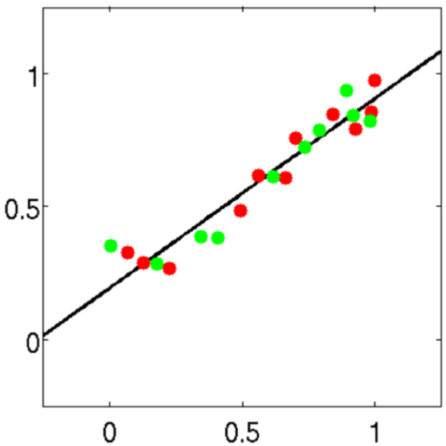
$$\tilde{L}(\boldsymbol{w}) = L(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_2^2$$

  - $L(\boldsymbol{w})$ is the original regression or classification loss
  - $\|\boldsymbol{w}\|_2 = \left(\sum_{k=1}^{K} w_k^2\right)^{\frac{1}{2}}$ is the L_2 norm
  - λ is the hyper-parameter used to control the influence of $\|\boldsymbol{w}\|^2$

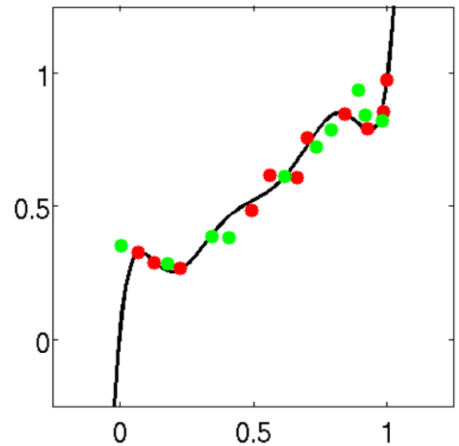
$\boxed{L_2 \text{ regularization}}$

- The properties of L_2 regularization
  - Prone to shrink the model parameters towards zero
  - The larger the λ is, the preference to small values of $\boldsymbol{w}$ is more strong

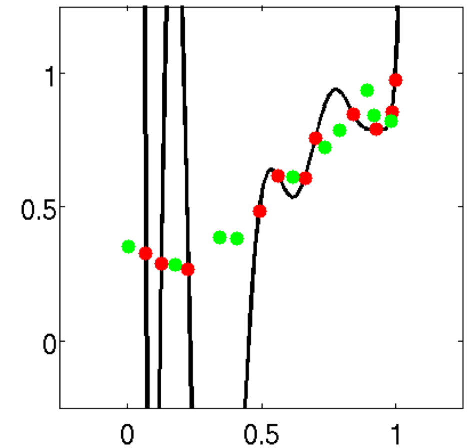
- Visualization of the impacts of regularization
  - No regularization



1-order

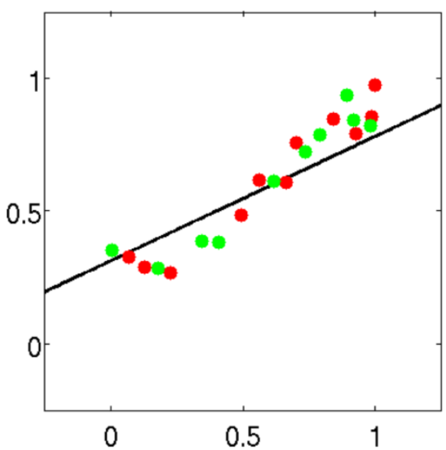


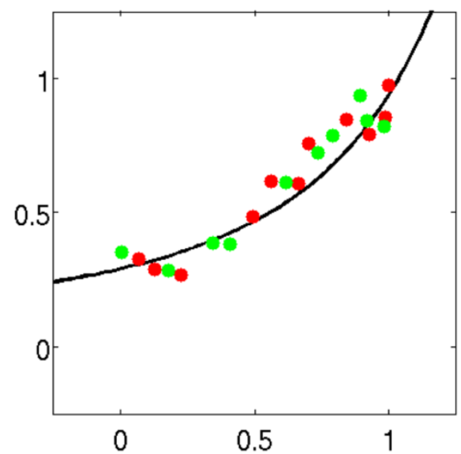
3-order

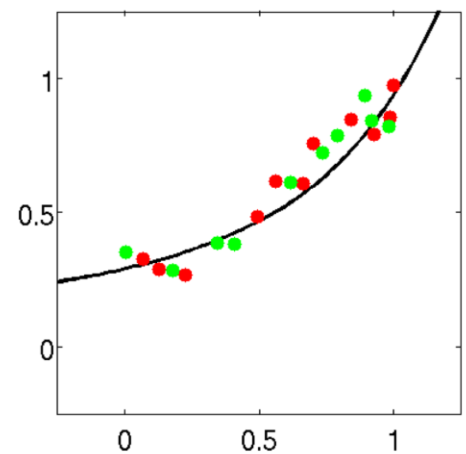


5-order

  - L_2 regularization with $\lambda = 1$





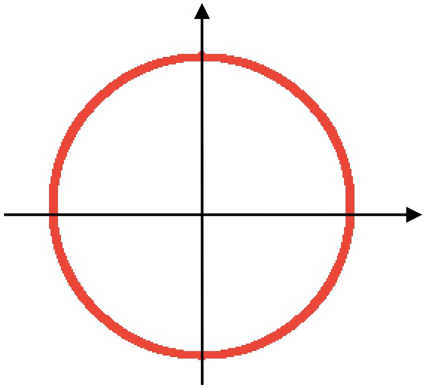


- Another commonly used regularization is L_1 regularization

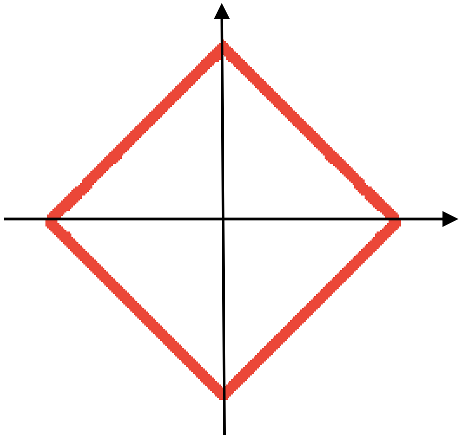
$$\tilde{L}(\boldsymbol{w}) = L(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_1$$

where $\|\boldsymbol{w}\|_1 \triangleq \sum_{k=1}^{K} |w_k|$ is the L_1 norm

- The contour line of L_2 and L_1 norm



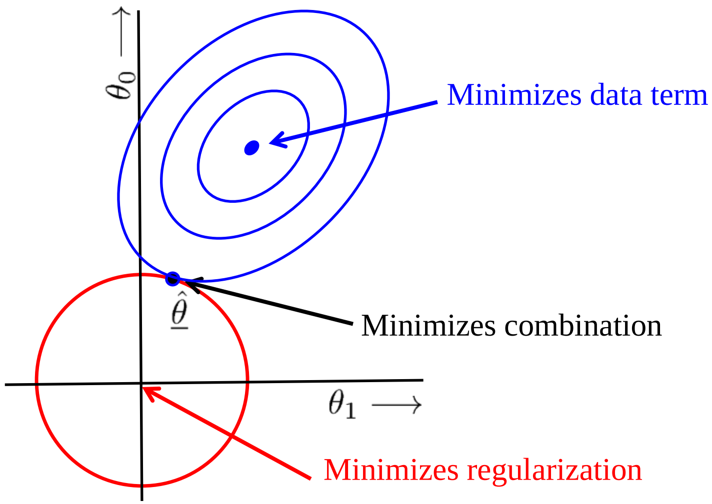
L_2 norm

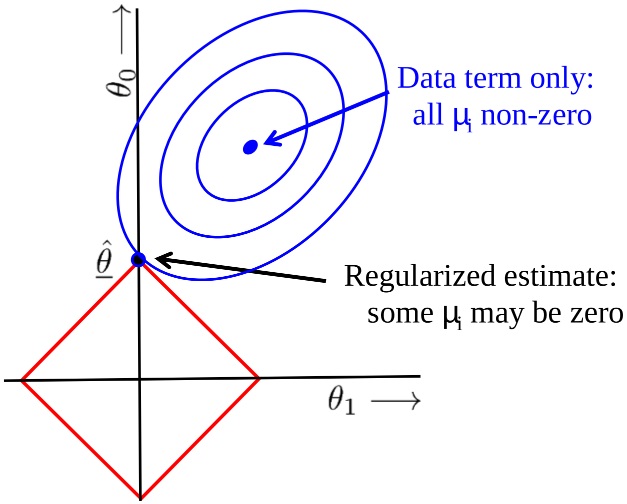


L_1 norm

- Similar to the L_2 regularization, the L_1 regularization also prefers to have small values for the model parameters

- But the L_1 regularization often leads to sparse solutions for $\boldsymbol{w}$, that is, many elements in $\boldsymbol{w}$ are zeros





Homework1

作业题目：实现并对比线性分类器与非线性分类器

作业要求：

1. 实现Lecture 2线性分类器（多类分类采用softmax函数）

2. 通过基函数非线性化步骤1的线性分类器，得到不含正则化的非线性分类器（基函数的选择不限）

3. 通过L1和L2范数分别对步骤2的非线性分类器进行正则化，正则化系数$\lambda = 1$，分别得到含L1和L2正则化的非线性分类器。

4. 在UCI Machine Learning Repository（https://archive.ics.uci.edu/ml/datasets.php）找到自己认为合适的数据集对比：线性分类器、不含正则化的非线性分类器、含L1正则化的非线性分类器、含L2正则化的非线性分类器。

5. 对比指标采用分类精度，即报告每一个分类器在测试集$\{(f(\mathbf{x}^{(i)}), y^{(i)}), i = 1, \ldots, m\}$上得到的ACC

$$ACC = \frac{1}{m}\sum_{i=1}^{m} \delta\left(f(\mathbf{x}^{(i)}), y^{(i)}\right)$$

其中$\delta\left(f(\mathbf{x}^{(i)}), y^{(i)}\right) = 1$，若$f(\mathbf{x}^{(i)}) = y^{(i)}$；否则为0。

6. 提交代码+数据集+详细实验报告及分析（编程语言不限、报告字数不限，需要透彻分析），压缩包提交：学号+姓名。

7. 提交日期：4月8日。提交邮箱：sysumldm2022@163.com