Some Kernels

The following are the most commonly used kernels

- linear kernel: $\psi(\mathbf{x}) = \mathbf{x}$
- sigmoid: $K(\mathbf{x}, \mathbf{x}') = \tanh(\gamma\langle \mathbf{x}, \mathbf{x}' \rangle + \zeta)$ (for γ, $\zeta > 0$)
- degree-Q polynomial kernel
- Gaussian-radial basis function (RBF) kernel

Degree-Q polynomial kernel

Definition

For given constants $\gamma > 0, \zeta > 0$ and for $Q \in \mathbb{N}$, the *degree-Q polynomial kernel* is

$$K(\mathbf{x}, \mathbf{x}') = (\zeta + \gamma \langle \mathbf{x}, \mathbf{x}' \rangle)^Q$$

time? ~~$\mathcal{O}(d)$~~ ~~$\mathcal{O}(d+Q)$~~ $\mathcal{O}(d + \log Q)$

Example

For $Q = 2$:

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

$$\psi(\mathbf{x}) = [\zeta, \sqrt{2\zeta\gamma}x_1, \sqrt{2\zeta\gamma}x_2, \ldots, \sqrt{2\zeta\gamma}x_d, \gamma x_1 x_1, \gamma x_1 x_2, \ldots, \gamma x_d x_d]^T \in \mathbb{R}^{1+d+d^2}$$

In general: $\psi(\vec{x}) \in \mathbb{R}^{\mathcal{O}(d^Q)}$

Gaussian-RBF Kernel

Definition

For a given constant $\gamma > 0$ the *Gaussian-RBF kernel* is

$$K(\mathbf{x}, \mathbf{x}') = e^{-\gamma ||\mathbf{x}-\mathbf{x}'||^2}$$

What is $\psi(\mathbf{x})$? Assume $\gamma = 1$ and $\mathbf{x} = x \in \mathbb{R}$ for simplicity, then

$$\begin{aligned} K(x, x') &= e^{-||x-x'||^2} \\ &= e^{-x^2} e^{2xx'} e^{-(x')^2} \\ &= e^{-x^2} \left(\sum_{k=0}^{+\infty} \frac{2^k (x)^k (x')^k}{k!} \right) e^{-(x')^2} \end{aligned}$$

$$\Rightarrow \psi(x) = e^{-x^2} \left(1, \sqrt{\frac{2}{1!}} x, \sqrt{\frac{2^2}{2!}} x^2, \sqrt{\frac{2^3}{3!}} x^3, \dots \right)^T$$

$\Rightarrow \psi(x)$ has infinite number of dimensions!

Choice of Kernel

Notes

- polynomial kernel: usually used with $Q \leq 10$
- Gaussian-RBF kernel: usually $\gamma \in [0, 1]$
- many other choices are possible!

Mercer's condition

$K(\mathbf{x}, \mathbf{x}')$ is a valid kernel function if and only if the kernel matrix

$$K = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) \dots & K(\mathbf{x}_1, \mathbf{x}_m) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) \dots & K(\mathbf{x}_2, \mathbf{x}_m) \\ \vdots & \vdots & \vdots \\ K(\mathbf{x}_m, \mathbf{x}_1) & K(\mathbf{x}_m, \mathbf{x}_2) \dots & K(\mathbf{x}_m, \mathbf{x}_m) \end{bmatrix}$$

is always symmetric positive semi-definite for any given $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m$.

Support Vector Machines for Regression

$Y = \mathbb{R}$

SVMs can be also used for regression. The function to be minimized will be

$$\frac{\lambda}{2}||\mathbf{w}||^2 + \sum_{i=1}^{m} V_\varepsilon(y_i - \langle \mathbf{x}_i, \mathbf{w} \rangle - b)$$

ℓ_2-regularization

loss function

observed value

– prediction for $\vec{x}_i$ (by model $\vec{w}, b$)

where

$$V_\varepsilon(r) = \begin{cases} 0 & \text{if } |r| < \varepsilon \\ |r| - \varepsilon & \text{otherwise} \end{cases}$$

One can prove that the solution has the form:

$$\mathbf{w} = \sum_{i=1}^{m} (\alpha_i^* - \alpha_i)\mathbf{x}_i$$

in the Training set

and that the final model produced in output is

$$h(\mathbf{x}) = \sum_{i=1}^{m} (\alpha_i^* - \alpha_i)\langle \mathbf{x}_i, \mathbf{x} \rangle + b$$

if we use a transformation $\psi \Rightarrow \langle \psi(\vec{x}), \psi(\vec{x}') \rangle = K_\psi(\vec{x}, \vec{x}')$

where $\alpha_i^*, \alpha_i \geq 0$ and are the solution to a suitable QP.

Definition

Support vector: $\mathbf{x}_i$ such that $\alpha_i^* - \alpha_i \neq 0$

One can define kernels, similarly to SVM for classification.

Exercise 4

Assuming we have the following dataset ($x_i \in \mathbb{R}^2$) and by solving the SVM for classification we get the corresponding optimal dual variables:

i	x_i^T	y_i	α_i^*
1	[0.2 -1.4]	-1	0
2	[-2.1 1.7]	1	0
3	[0.9 1]	1	0.5
4	[-1 -3.1]	-1	0
5	[-0.2 -1]	-1	0.25
6	[-0.2 1.3]	1	0
7	[2.0 -1]	-1	0.25
8	[0.5 2.1]	1	0

Answer to the following:

(A) What are the support vectors?

(B) Draw a schematic picture reporting the data points (approximately) and the optimal separating hyperplane, and mark the support vectors. Would it be possible, by moving only two data points, to obtain the SAME separating hyperplane with only 2 support vectors? If so, draw the modified configuration (approximately).

Machine Learning

Regularization and Feature Selection

Fabio Vandin

November 27^{th}, 2023

Ridge Regression: Matrix Form

Linear regression: pick

$$\arg\min_{\mathbf{w}} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})$$

Ridge regression: pick

$$\arg\min_{\mathbf{w}} \left(\lambda ||\mathbf{w}||^2 + (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \right)$$

Want to find $\mathbf{w}$ which minimizes

$$f(\mathbf{w}) = \lambda||\mathbf{w}||^2 + (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}).$$

How? $2\lambda\vec{w}$ $\quad -2X^T(\vec{y} - X\vec{w})$

Compute gradient $\frac{\partial f(\mathbf{w})}{\partial \mathbf{w}}$ of objective function w.r.t $\mathbf{w}$ and compare it to 0.

$$\frac{\partial f(\mathbf{w})}{\partial \mathbf{w}} = 2\lambda\mathbf{w} - 2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w})$$

Then we need to find $\mathbf{w}$ such that

$$2\lambda\mathbf{w} - 2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) = 0$$

$$\lambda\vec{w} - X^T(\vec{y} - X\vec{w}) = 0$$

$$\lambda\vec{w} + X^TX\vec{w} = X^T\vec{y}$$

$$(\lambda I + X^TX)\vec{w} = X^T\vec{y} \Rightarrow \vec{w} = (\lambda I + X^TX)^{-1}X^T\vec{y}$$

$$2\lambda \mathbf{w} - 2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) = 0$$

is equivalent to

$$\left(\lambda \mathbf{I} + \mathbf{X}^T\mathbf{X}\right)\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

Note:

- $\mathbf{X}^T\mathbf{X}$ is positive semidefinite
- $\lambda\mathbf{I}$ is positive definite

$\Rightarrow \lambda\mathbf{I} + \mathbf{X}^T\mathbf{X}$ is positive definite

$\Rightarrow \lambda\mathbf{I} + \mathbf{X}^T\mathbf{X}$ is invertible

Ridge regression solution:

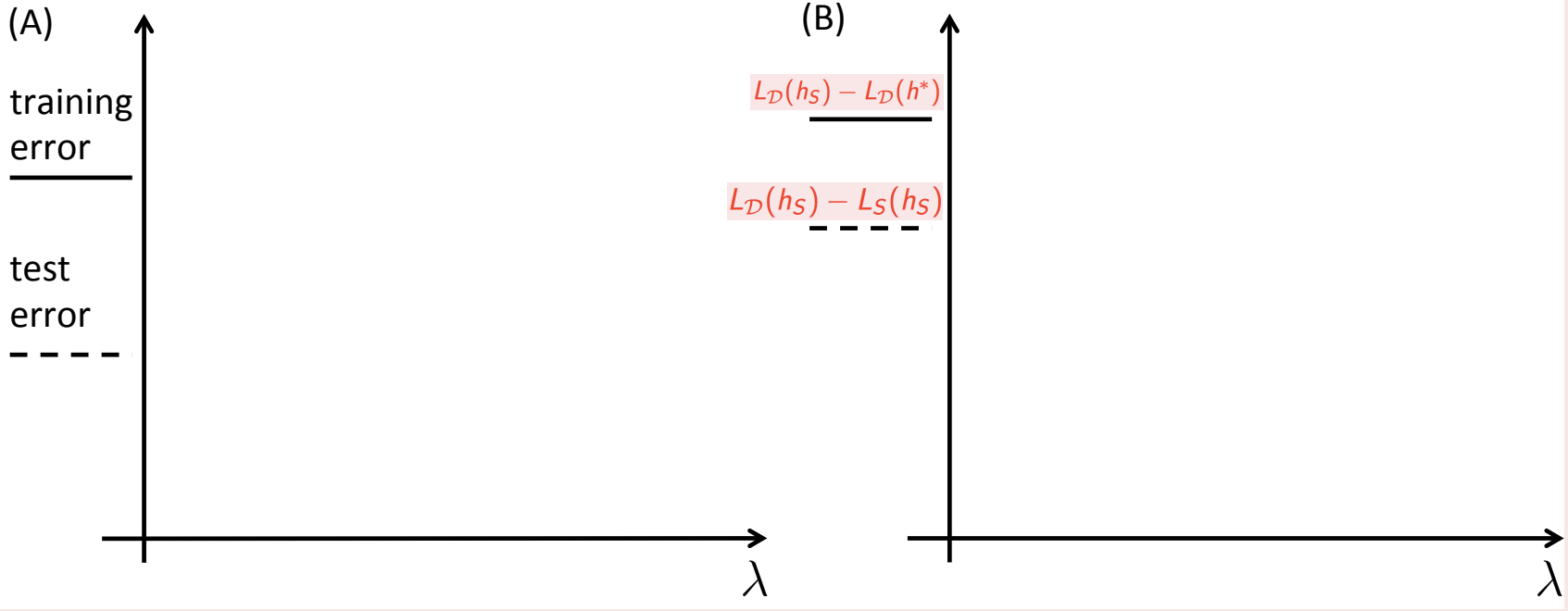
$$\mathbf{w} = \left(\lambda\mathbf{I} + \mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

Exercise 5

Consider the ridge regression problem $\arg\min_{\mathbf{w}} \lambda||\mathbf{w}||^2 + \sum_{i=1}^{m}(\langle \mathbf{w}, \mathbf{x_i}\rangle - y_i)^2$. Let: h_S be the hypothesis obtained by ridge regression on with training set S; h^* be the hypothesis of minimum generalization error among all linear models.

(A) Draw, in the plot below, a *typical* behaviour of (i) *the training error* and (ii) *the test/generalization error* of h_S as a function of λ.

(B) Draw, in the plot below, a *typical* behaviour of (i) $L_{\mathcal{D}}(h_S) - L_{\mathcal{D}}(h^*)$ and (ii) $L_{\mathcal{D}}(h_S) - L_S(h_S)$ as a function of λ.



Feature Selection

In general, in machine learning one has to decide what to use as features (= input) for learning.

Even if somebody gives us a representation as a feature vector, maybe there is a "better" representation?

What is "better"?

Example

- features x_1, x_2, output y
- $x_1 \sim Uniform(-1, 1)$
- $y = x_1^2$
- $x_2 \sim y + Uniform(-0.01, 0.01)$

If we want to predict y, which feature is better: x_1 or x_2?

- x_1: because with $(x_1)^2$ we perfectly predict y.
But what if you use x_1 as feature of a linear model?
Prediction is not great!
- x_2, because with linear models we predict y exactly up to noise (Uniform$(-0.01, 0.01)$). But we could do better by looking at x_1^2.

Feature Selection: Scenario

We have a large pool of features

Goal: select a small number of features that will be used by our (final) predictor

Assume $\mathcal{X} = \mathbb{R}^d$.

Goal: learn (final) predictor using $k << d$ predictors

Motivation?

- prevent overfitting: less predictors $\Rightarrow$ hypotheses of lower complexity!
- predictions can be done faster
- useful in many applications!

Feature Selection: Computational Problem

Assume that we use the Empirical Risk Minimization (ERM) procedure.

Assumption: an hypothesis $h \in \mathcal{H}$ corresponds to a vector of weights $\mathbf{w} \in \mathbb{R}^d$

The problem of selecting k features that minimize the empirical risk can be written as:

$$\min_{\mathbf{w}} L_S(\mathbf{w}) \quad \text{subject to} \quad ||\mathbf{w}||_0 \leq k$$

where $||\mathbf{w}||_0 = |\{i : w_i \neq 0\}|$

Subset Selection

How do we find the solution to the problem below?

$$\min_{\mathbf{w}} L_S(\mathbf{w}) \text{ subject to } ||\mathbf{w}||_0 \leq k$$

Note: the solution will always include k features

Let:

- $\mathcal{I} = \{1, \ldots, d\}$;
- given $p = \{i_1, \ldots, i_k\} \subseteq \mathcal{I}$: $\mathcal{H}_p$ = hypotheses/models where only features $w_{i_1}, w_{i_2} \ldots, w_{i_k}$ are used

$P^{(k)} \leftarrow \{J \subseteq \mathcal{I} : |J| = k\}$;

foreach $p \in P^{(k)}$ **do**

  $h_p \leftarrow \arg\min_{h \in \mathcal{H}_p} L_S(h)$;

return $p_k \leftarrow \arg\min_{p \in P^{(k)}} L_S(h_p)$;

Complexity? Learn $\Theta\left(\binom{d}{k}\right) \in \Theta\left(d^k\right)$ models $\Rightarrow$ exponential algorithm!

Can we do better?

Proposition

The optimization problem of feature selection NP-hard.

↳ Not likely to find a polynomial algorithm

What can we do?

Heuristic solution $\Rightarrow$ greedy algorithms

Greedy Algorithms for Feature Selection

Forward Selection: start from the empty solution, add one feature at the time, until solution has cardinality k

$sol \leftarrow \emptyset$;
while $|sol| < k$ **do**
  foreach $i \in \mathcal{I} \setminus sol$ **do**
    $p \leftarrow sol \cup \{i\}$;
    $h_p \leftarrow \arg\min_{h \in \mathcal{H}_p} L_S(h)$;
  $sol \leftarrow sol \cup \arg\min_{i \in \mathcal{I} \setminus sol} L_S(h_{sol \cup \{i\}})$;
return sol;

Complexity? Learns $\Theta(kd)$ models

Backward Selection: start from the solution which includes all features, remove one feature at the time, until solution has cardinality k

Pseudocode: analogous to forward selection [Exercize!]

Complexity? Learns $\Theta\left((d-k)d\right)$ models

Notes

We have used only training set to select the best hypothesis...

$\Rightarrow$ we may overfit!

Solution? Use validation! (or cross-validation)

Split data into training data and validation data, learn models on training, evaluate (= pick among different hypothesis models) on validation data. Algorithms are similar.

Note: now the best model (in terms of validation error) may include less than k features!

Subset Selection with Validation Data

S = training data (from data split)
V = validation data (from data split)

Using training and validation:

for $\ell \leftarrow 0$ *to* k **do**
    $P^{(\ell)} \leftarrow \{J \subseteq \mathcal{I} : |J| = \ell\}$;
    foreach $p \in P^{(\ell)}$ **do**
        $h_p \leftarrow \arg\min_{h \in \mathcal{H}_p} L_S(h)$;
    $p_\ell \leftarrow \arg\min_{p \in P^{(\ell)}} L_V(h_p)$;
return $\arg\min_{p \in \{p_0, p_1, \ldots, p_k\}} L_V(h_p)$

Forward Selection with Validation Data

Using training and validation:

$sol \leftarrow \emptyset$;
while $|sol| < k$ **do**
  foreach $i \in \mathcal{I} \setminus sol$ **do**
    $p \leftarrow sol \cup \{i\}$;
    $h_p \leftarrow \arg\min_{h \in \mathcal{H}_p} L_S(h)$;
  $sol \leftarrow sol \cup \arg\min_{i \in \mathcal{I} \setminus sol} L_V(h_{sol \cup \{i\}})$;
return sol;