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

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

How? gradient

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

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?

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

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

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

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

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

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

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

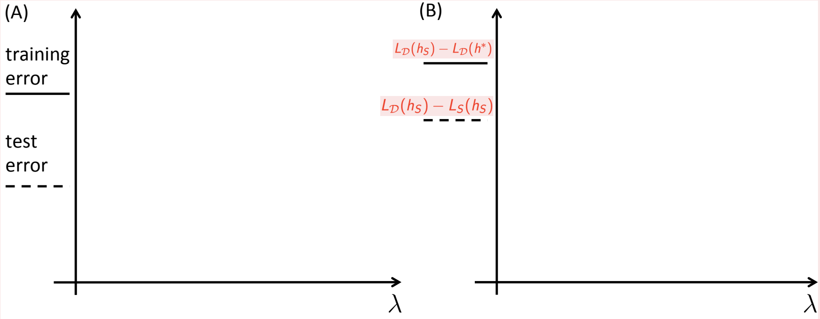
Ridge regression solution:
$\mathbf{w} = \left(\lambda \mathbf{I} + \mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$

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.

Example

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

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

No-free lunch...

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

(features)

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

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 $\vec{w} \in \mathbb{R}^d$

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

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

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

Feature Selection: Computational Problem

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

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

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

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

How can we solve it?

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!

≈ it is unlikely that there is a poly-time alg. to solve it.

Can we do better?

Proposition

The optimization problem of feature selection NP-hard.

Can we do better?

Proposition

The optimization problem of feature selection NP-hard.

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;

Backward Selection with validation: similar [Exercize]

Similar approach for all algorithms with cross-validation [Exercize]

Bibliography [UML]

Regularization and Ridge Regression: Chapter 12

- no Section 13.3;
- Section 13.4 only up to Corollary 13.8 (excluded)

Feature Selection and LASSO: Chapter 25

- only Section 25.1.2 (introduction and "Backward Elimination") and 25.1.3