Machine Learning

Regularization and Feature Selection

Fabio Vandin  November 14, 2017
Regularized Loss Minimization

Assume $h$ is defined by a vector $w = (w_1, \ldots, w_d)^T \in \mathbb{R}^d$ (e.g., linear models)

**Regularization function** $R : \mathbb{R}^d \to \mathbb{R}$

Regularized Loss Minimization (RLM): pick $h$ obtained as

$$\arg \min_w (L_S(w) + R(w))$$

**Intuition:** $R(w)$ is a “measure of complexity” of hypothesis $h$ defined by $w$

$\Rightarrow$ regularization balances between low empirical risk and “less complex” hypotheses

We will see some of the most common regularization function
Regularization

\( \ell_1 \) Regularization

Regularization function: \( R(w) = \lambda \| w \|_1 \)

- \( \lambda \in \mathbb{R}, \lambda > 0 \)
- \( \ell_1 \) norm: \( \| w \|_1 = \sum_{i=1}^{d} |w_i| \)

Therefore the learning rule is: pick

\[
A(S) = \arg \min_w (L_S(w) + \lambda \| w \|_1)
\]

Intuition:
- \( \| w \|_1 \) measures the “complexity” of hypothesis defined by \( w \)
- \( \lambda \) regulates the tradeoff between the empirical risk \( (L_S(w)) \) or overfitting and the complexity \( (\| w \|_1) \) of the model we pick
LASSO

Linear regression with squared loss + $\ell_1$ regression $\Rightarrow$ LASSO (least absolute shrinkage and selection operator)

LASSO: pick

$$
\mathbf{w} = \arg \min_{\mathbf{w}} \lambda \| \mathbf{w} \|_1 + \sum_{i=1}^{m} (\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i)^2
$$

How?

Notes:

- no closed form solution!
- $\ell_1$ norm is a convex function and squared loss is a convex
  $\Rightarrow$ problem can be solved efficiently! (true for every convex
  loss function)
- $\ell_1$ regularization often induces sparse solutions
LASSO and Sparse Solution

Ridge Regression

LASSO
Ridge Regression vs LASSO

\[ \ell_1 \text{ regularization performs a sort of feature selection} \]
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 $y$
- $x_1 \sim U[-1, 1]$
- $y = x_1^2$
- $x_2 \sim y + U[-0.01, 0.01]$

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

**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!
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:

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

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

How can we solve it?
Subset Selection

How do we find the solution to the problem below?

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

Let:

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

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

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

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

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

**Complexity?** Learn $$\Theta \left( \binom{d}{k} \right) \in \Theta \left( d^k \right)$$ models $\Rightarrow$ exponential algorithm!
What about finding the best subset of features (of any size)?

```
for k ← 0 to d do
    P(k) ← \{ J ⊆ I : |J| = k \};
    foreach p ∈ P(k) do
        h_p ← \arg\min_{h ∈ H_p} L_S(h);
        h(k) ← \arg\min_{p ∈ P(k)} L_S(h_p);
    return \arg\min_{h ∈ \{h(0), h(1), \ldots, h(d)\}} L_S(h)
```

**Complexity?** Learn $\Theta(2^d)$ models!
Can we do better?

Proposition

The optimization problem of feature selection NP-hard.

What can we do?

Heuristic solution ⇒ 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 ← ∅;
while |sol| < k do
    foreach $i ∈ I \ sol$ do
        $p ← sol \cup \{i\}$;
        $h_p ← \arg \min_{h ∈ H_p} L_S(h)$;
        $sol ← sol \cup \arg \min_{i ∈ I \setminus sol} L_S(h_{sol∪\{i\}})$;
    return sol;
```

**Complexity?** Learns $Θ(kd)$ models
**Backward Selection**: start from the solution which includes all features, remove one features at the time, until solution has cardinality $k$

Pseudocode: analogous to forward selection [Exercize!]

**Complexity?** Learns $\Theta(kd)$ models
We have used only training set to select the best hypothesis...

⇒ 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.
Subset Selection with Validation Data

\[ S = \text{training data (from data split)} \]
\[ V = \text{validation data (from data split)} \]

Using training and validation:

\[
\text{for } k \leftarrow 0 \text{ to } d \text{ do} \\
\quad P^{(k)} \leftarrow \{ J \subseteq \mathcal{I} : |J| = k \} ; \\
\quad \text{foreach } p \in P^{(k)} \text{ do} \\
\quad \quad h_p \leftarrow \arg \min_{h \in \mathcal{H}_p} L_S(h) ; \\
\quad \quad h^{(k)} \leftarrow \arg \min_{p \in P^{(k)}} L_V(h_p) ; \\
\quad \text{return } \arg \min_{h \in \{ h^{(0)}, h^{(1)}, \ldots, h^{(d)} \}} L_V(h) \]

Forward Selection with Validation Data

Using training and validation:

\[ \text{sol} \leftarrow \emptyset; \]

while \(|\text{sol}| < k\) do

\[
\begin{align*}
\text{foreach } i \in \mathcal{I} \setminus \text{sol} \text{ do} \\
p &\leftarrow \text{sol} \cup \{i\}; \\
h_p &\leftarrow \arg\min_{h \in \mathcal{H}_p} L_S(h); \\
\text{sol} &\leftarrow \text{sol} \cup \arg\min_{i \in \mathcal{I} \setminus \text{sol}} L_V(h_{\text{sol} \cup \{i\}}); \\
\end{align*}
\]

return \(\text{sol}\);
Backward Selection with validation: similar [Exercize]

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