CSE 546: Machine Learning **Lecture 5**

Feature Selection: Part 1

Instructor: Sham Kakade

1 Regression in the high dimensional setting

How do we learn when the number of features d is greater than the sample size n? In the previous lecture, we examined ridge regression and provided a dimension free rate of convergence. Now let us examine feature selection.

2 Feature Selection

Let us suppose there are s relevant features out of the d possible features. Throughout this analysis, let us assume that:

$$\mathbf{Y} = \mathbf{X}w_* + \eta,$$

where $\mathbf{Y} \in R^n$ and $\mathbf{X} \in \mathbb{R}^{n\times d}$. We assume that the support of w_* (the number of non-zero entries) is s.

2.1 Loss Minimization (Empirical Risk Minimization)

Define our "empirical loss" as:

$$\hat{L}(w) = \frac{1}{n}\|\mathbf{X}w - \mathbf{Y}\|^2$$

which has no expectation over $\mathbf{Y}$.

Suppose we knew the support size s. One algorithm is to simply find the estimator which minimizes the empirical loss and has support only on s coordinates. In particular, consider the estimator:

$$\hat{w}_{\text{subset selection}} = \arg\min_{\text{support}(w)\leq s} \hat{L}(w)$$

where the $\inf$ is over vectors with support size s. Computing this estimator is not computationally tractable in general (the naive algorithm runs in time d^s). Furthermore, finding the best subset is known to be an NP-hard problem.

How much better is this estimator better than the naive estimator? Recall the risk is:

$$R(\hat{w}_{\text{subset selection}}) = E_{\mathbf{Y}}\|\hat{w}_{\mathcal{T}} - w_*\|^2_{\mathbf{\Sigma}}$$

where the expectation is over $\mathbf{Y}$.

We have the following theorem:

Theorem 2.1. *Suppose the support of w_* is bounded by s. We have that the risk is bounded as:*

$$R(\hat{w}_{\textit{subset selection}}) \leq c\frac{s\log d}{n}\sigma^2$$

(where c isa universal constant).

2.2 Coordinate dependence?

Clearly, the coordinates system is important here, as the "support" is defined with respect to this coordinate system. However, note that the 'scale' in each coordinate is irrelevant here. In contrast, note the empirical risk minimization does not depend on the coordinate system.

3 Norms

The ℓ_p of a vector x is:

$$\|x\|_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$$

The ℓ_0 norm is defined as:

$$\|x\|_p = |\{i | x_i \neq 0\}|$$

which is the number of non-zero entries in x. Technically, the ℓ_0 norm is not a norm.

4 Lasso

Let us view the the subset selection problem as a regularized problem. A relaxed version of a hard constraint on the size of the subset would be to minimize:

$$\hat{L}(w) + \lambda\|w\|_0$$

One can show that for an appropriate choice of λ this algorithm also enjoys the same risk guarantee of the hard constrained subset selection algorithm (up to constants). Unfortunately, minimizing this objective function is also not computationally tractable.

A natural convex relaxation is to instead consider minimizing the following:

$$F(w) = \hat{L}(w) + \lambda\|w\|_1$$

which can be viewed as a convex relaxation to the ℓ_0 problem. This is referred to as the *Lasso*.

4.1 Coordinate Scalings

Often it is a good idea to transform the data so that the variance along each coordinate is 1. In other words, for each coordinate j, it often makes sense to do the following transformation:

$$X_{i,j} \leftarrow X_{i,j}/Z_i \text{ where } Z_j = \frac{1}{n}\sum_i X_{i,j}^2$$

Intuitively, this is to remove an arbitrary scale factor. A more precise reason for this will be discussed in the next lecture.

4.2 Optimization & Coordinate Descent

The 1-dimensional case: Suppose that we are in in the 1-dimensional case where each x_i is a scalar and so w is a scalar. The lasso problem is then to minimize:

$$\sum_i (y_i - wx_i)^2 + \lambda |w|$$

where $|w|$ is the absolute value function. To minimize this function, we can again set the gradient to 0 and solve.

A subtlety here is that the absolute value function is non-differentiable at 0. Note that for any $w \neq 0$ the gradient is:

$$-2 \sum_i x_i(y_i - wx_i) + \lambda \text{sign}(w) \tag{1}$$

where sign(w) is 1 if w is positive and -1 if w is negative.

There are three cases to check. If the minimizer w_* is positive, then we know that the first order condition implies that:

$$w_* = \frac{\sum_i y_i x_i - \lambda/2}{\sum_i x_i^2}$$

If we compute the right hand side and it is positive, then indeed this value is the minimizer.

Now suppose the minimizer w_* is negative, then we know that the first order condition implies that:

$$w_* = \frac{\sum_i y_i x_i + \lambda/2}{\sum_i x_i^2}$$

If we compute the right hand side and it is negative, then indeed this value is the minimizer.

Now suppose w_* is 0. Note that $|w|$ is not differentiable at 0. Here, one can show that we must have that:

$$2 \sum_i y_i x_i \in [-\lambda, \lambda]$$

So if we compute the left hand side and it is in the interval $[-\lambda, \lambda]$ then $w = 0$ is a minimizer. To see this, consider any small perturbation so that $w = \epsilon$. Suppose $\epsilon > 0$. For sufficiently small ϵ, the first term in (1) will still be in the interval $[-\lambda, \lambda]$ and so the gradient will be strictly positive (for small ϵ). Thus gradient descent will push us back to 0. Similarly, for $\epsilon < 0$, we will move back to 0.

Formally, the "sub-gradient" of $|w|$ can take any value in $[-1, 1]$, which is a valid tangent plane (see the wikipedia definition).

Coordinate Ascent:

The coordinate ascent algorithm for minimizing an objective function $F(w_1, w_2, \ldots w_n)$ is as follows:

1. Initialize: $w = 0$
2. choose a coordinate i (e.g. at random)
3. update w_i as follows:
   $$w_i \leftarrow \arg\min_{z \in R} F(w_1, \ldots, w_{i-1}, z, w_{i+1}, \ldots w_d)$$
   where the optimization is over the i-th coordinate (holding the other coordinates fixed). Then return to step 2.

Clearly, many natural variants are possible.

5 Relationship of Lasso to Compressed Sensing

As we discussed earlier, regression can be viewed as finding an approximate solution to an (inconsistent) linear system of equations. In compressed sensing, we are dealing with the setting were the system of equations is consistent, i.e. $Aw = b$ has a solution. Suppose we are in the case where A is of size $n \times d$ and $d > n$, so there are multiple solutions. In particular, we seek the sparsest solution:

$$\min_w \|w\|_0 \text{ s.t. } Ax = b$$

As before, this problem is not computationally tractable.

The convex relaxation is the following optimization problem:

$$\min_w \|w\|_1 \text{ s.t. } Ax = b$$

Similar to the case of lasso, under certain assumptions this can recover the solution to the ℓ_0 problem.

6 Greedy Algorithms

There are variety of greedy algorithms and numerous naming conventions for these algorithms. These algorithms must rely on some stopping condition (or some condition to limit the sparsity level of the solution).

6.1 Stagewise Regression / Matching Pursuit / Boosting

Here, we typically do no regularize our objective function and, instead, directly deal with the empirical loss $\hat{L}(w_1, w_2, \ldots w_n)$. This class of algorithms for minimizing an objective function $\hat{L}(w_1, w_2, \ldots w_n)$ is as follows:

1. Initialize: $w = 0$
2. choose the coordinate i_* which can result in the greatest decrease in error, i.e.
$$i_* \in \arg\min_i \min_{z \in R} F(w_1, \ldots, w_{i-1}, z, w_{i+1}$$
3. update w_{i_*} as follows:
$$w_i \leftarrow \arg\min_{z \in R} F(w_1, \ldots, w_{i_*-1}, z, w_{i_*+1}, \ldots w_d)$$
where the optimization is over the i-th coordinate (holding the other coordinates fixed).
4. While some termination condition is not met, return to step 2. This termination condition can be looking at the error on some holdout set or simply just running the algorithm for some predetermined number of steps.

Variants: Clearly, many variants are possible. Sometimes (for loss functions other than the square loss) it is costly to do the minimization exactly so we sometimes choose i_* based on another method (e.g. the magnitude of the gradient of a coordinate). We could also re-optimize all the weights of all those features which were are currently added. Also, sometimes we do *backward steps* where we try to prune away some of the features which are added.

Relation to boosting: In boosting, we sometimes do not explicitly enumerate the set of all features. Instead, we have a "weak learner" which provides us with a new feature. The importance of this viewpoint is that sometimes it is difficult to enumerate the set of all features (e.g. our features could be decision trees, so our feature vector x could be of dimension the number of possible tress). Instead, we just assume some oracle which in step 2 which provides us with a feature.

There are numerous variants.

6.2 Stepwise Regression / Orthogonal Matching Pursuit

Note that the previous algorithm finds i_* by only checking the improvement in performance keeping all the other variables fixed. At any given iteration, we have some subset $\mathcal{S}$ of features whose weights are not 0. Instead, when determining which coordinate i to add, we could look improvement based on reoptimizing the weights on the full set $\mathcal{S} \cup \{i\}$. This is a more costly procedure computationally, though there are some ways to reduce the computational cost.