Introduction to Machine Learning - CS725

Instructor: Prof. Ganesh Ramakrishnan

Lecture 14 -Non-Parametric Regression, Algorithms for Optimizing SVR and Lasso

Kernels in SVR

- Recall:
  $max_{\alpha_i,\alpha_i^*} -\frac{1}{2}\sum_i\sum_j(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)K(\mathbf{x}_i,\mathbf{x}_j)-\epsilon\sum_i(\alpha_i+\alpha_i^*)+\sum_i y_i(\alpha_i-\alpha_i^*)$
  such that $\sum_i(\alpha_i-\alpha_i^*)=0$, $\alpha_i,\alpha_i^*\in[0,C]$ and the decision function:
  $f(\mathbf{x})=\sum_i(\alpha_i-\alpha_i^*)K(\mathbf{x}_i,\mathbf{x})+b$
  are all in terms of the kernel $K(\mathbf{x}_i,\mathbf{x}_j)$ only
- *One can now employ any mercer kernel in SVR or Ridge Regression to implicitly perform linear regression in higher dimensional spaces*
- Check out applet at `https://www.csie.ntu.edu.tw/~cjlin/libsvm/` to see the effect of non-linear kernels in SVR

Basis function expansion & Kernel: Part 1

Consider regression function $f(\mathbf{x}) = \sum_{j=1}^{p} w_j \phi_j(\mathbf{x})$ with weight vector $\mathbf{w}$ estimated as

$$\mathbf{w}_{Pen} = \underset{\mathbf{w}}{\operatorname{argmin}}\ \mathcal{L}(\phi, \mathbf{w}, \mathbf{y}) + \lambda\Omega(\mathbf{w})$$

It can be shown that for $p \in [0, \infty)$, under certain conditions on K, the following can be equivalent representations

- $$f(\mathbf{x}) = \sum_{j=1}^{p} w_j \phi_j(\mathbf{x})$$
- And
$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$
- For what kind of regularizers $\Omega(\mathbf{w})$, loss functions $\mathcal{L}(\phi, \mathbf{w}, \mathbf{y})$ and $p \in [0, \infty)$ will these dual representations hold?[1]

[1] Section 5.8.1 of Tibshi.

Basis function expansion & Kernel: Part 2

- We could also begin with (Eg: NadarayaWatson kernel regression)

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}, \mathbf{x}_i) = \frac{\sum_{i=1}^{m} y_i k_n(||\mathbf{x} - \mathbf{x}_i)||}{\sum_{i=1}^{m} k_n(||\mathbf{x} - \mathbf{x}_i||)}$$

A non-parametric kernel k_n is a non-negative real-valued integrable function satisfying the following two requirements: $\int_{-\infty}^{+\infty} k_n(u)du = 1$ and $k_n(-u) = k_n(u)$ for all values of u

[2]Section 2.8.2 of Tibshi

Basis function expansion & Kernel: Part 2

- We could also begin with (Eg: NadarayaWatson kernel regression)

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}, \mathbf{x}_i) = \frac{\sum_{i=1}^{m} y_i k_n(||\mathbf{x} - \mathbf{x}_i)||}{\sum_{i=1}^{m} k_n(||\mathbf{x} - \mathbf{x}_i||)}$$

  A non-parametric kernel k_n is a non-negative real-valued integrable function satisfying the following two requirements: $\int_{-\infty}^{+\infty} k_n(u)du = 1$ and $k_n(-u) = k_n(u)$ for all values of u
- *E.g.:* $k_n(x_i - x) = I(||x_i - x|| \leq ||x_{(k)} - x||)$ where $x_{(k)}$ is the training observation ranked k^{th} in distance from x and $I(S)$ is the indicator of the set S
- This is precisely the Nearest Neighbor Regression model
- Kernel regression and density models are other examples of such *local regression* methods[2]
- The broader class - Non-Parametric Regression: $y = g(\mathbf{x}) + \epsilon$ where functional form of $g(\mathbf{x})$ is not fixed

[2] Section 2.8.2 of Tibshi

Non-parametric Kernel weighted regression (Local Linear Regression): Tut 5, Prob 3

Given $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_i, y_i), \ldots, (\mathbf{x}_n, y_n)\}$, predict $f(\mathbf{x}') = (\mathbf{w}'^\top \phi(\mathbf{x}') + b)$ for each test (or query point) $\mathbf{x}'$ as:

$$(\mathbf{w}', b') = \underset{\mathbf{w}, b}{\operatorname{argmin}} \sum_{i=1}^{n} K(\mathbf{x}', \mathbf{x}_i) \left(y_i - (\mathbf{w}^\top \phi(x_i) + b) \right)^2$$

1. If there is a closed form expression for $(\mathbf{w}', b')$ and therefore for $f(x')$ in terms of the known quantities, derive it.
2. How does this model compare with linear regression and $k-$nearest neighbor regression? What are the relative advantages and disadvantages of this model?
3. In the one dimensional case (that is when $\phi(x) \in \Re$), graphically try and interpret what this regression model would look like, say when $K(.,.)$ is the linear kernel[3].

[3] Hint: What would the regression function look like at each training data point?

Answer to Question 1

The weighing factor $r_i^{x'}$ of each training data point $(\mathbf{x}_i, y_i)$ is now also a function of the query or test data point $(\mathbf{x}', ?)$, so that we write it as $r_i^{x'} = K(\mathbf{x}', \mathbf{x}_i)$ for $i = 1, \ldots, m$. Let $r_{m+1}^{x'} = 1$ and let R be an $(m+1) \times (m+1)$ diagonal matrix of $r_1^{x'}, r_2^{x'}, \ldots, r_{m+1}^{x'}$.

$$R = \begin{bmatrix} r_1^{x'} & 0 & \ldots & 0 & \\ 0 & r_2^{x'} & \ldots & 0 & \\ \ldots & \ldots & \ldots & \ldots & 1 \\ 0 & 0 & 0 & \ldots & r_{m+1}^{x'} \end{bmatrix}$$

Further, let

$$\Phi = \begin{bmatrix} \phi_1(x_1) & \ldots & \phi_p(x_1) & 1 \\ \ldots & \ldots & \ldots & 1 \\ \phi_1(x_m) & \ldots & \phi_p(x_m) & 1 \end{bmatrix}$$

and

Answer to Question 1 (contd.)

$$\widehat{\mathbf{w}} = \begin{bmatrix} w_1 \\ \dots \\ w_p \\ b \end{bmatrix}$$

and

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \dots \\ y_m \end{bmatrix}$$

The sum-square error function then becomes

$$\frac{1}{2}\sum_{i=1}^{m} r_i(y_i - (\widehat{\mathbf{w}}^T\phi(x_i) + b))^2 = \frac{1}{2}||\sqrt{R}\mathbf{y} - \sqrt{R}\Phi\widehat{\mathbf{w}}||_2^2$$

where $\sqrt{R}$ is a diagonal matrix such that each diagonal element of $\sqrt{R}$ is the square root of the corresponding element of R.

Answer to Question 1 (contd.)

The sum-square error function:

$$\frac{1}{2}\sum_{i=1}^{m} r_i(y_i - (\widehat{\mathbf{w}}^T\phi(x_i) + b))^2 = \frac{1}{2}||\sqrt{R}\mathbf{y} - \sqrt{R}\Phi\widehat{\mathbf{w}}||_2^2$$

This convex function has a global minimum at $\widehat{\mathbf{w}}_*^{x'}$ such that

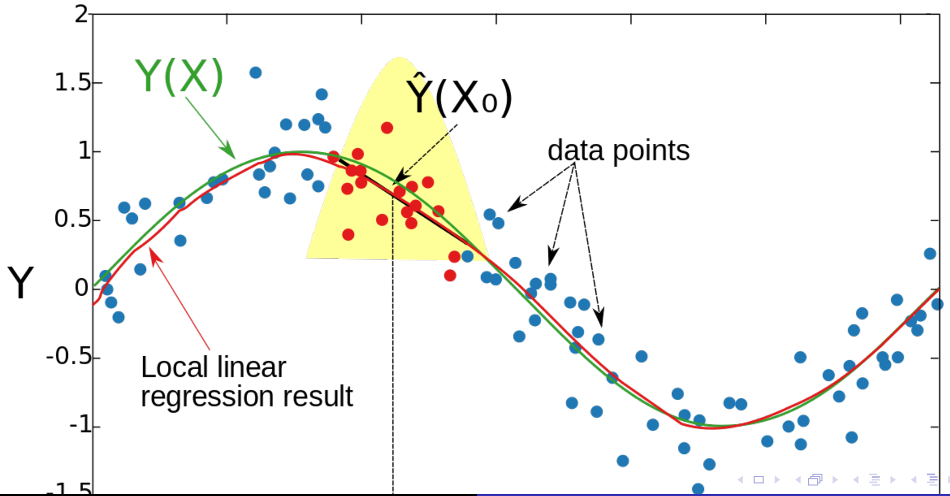
$$\widehat{\mathbf{w}}_*^{x'} = (\Phi^T R\Phi)^{-1}\Phi^T R\mathbf{y}$$

This is referred to as local linear regression (Section 6.1.1 of Tibshi).

Answer to Question 2

1. Local linear regression gives more importance (than linear regression) to points in $\mathcal{D}$ that are closer/similar to $\mathbf{x}'$ and less importance to points that are less similar.
2. Important if the regression curve is supposed to take different shapes in different parts of the space.
3. Local linear regression comes close to k-nearest neighbor. But unlike k-nearest neighbor, local linear regression gives you a smooth solution

Answer to Question 3



Solving the SVR Dual Optimization Problem

- The SVR dual objective is:
  $max_{\alpha_i,\alpha_i^*} - \frac{1}{2}\sum_i \sum_j (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(x_i, x_j)$
  $-\epsilon \sum_i (\alpha_i + \alpha_i^*) + \sum_i y_i(\alpha_i - \alpha_i^*)$ such that $\sum_i(\alpha_i - \alpha_i^*) = 0,\ \alpha_i, \alpha_i^* \in [0, C]$
- This is a linearly constrained quadratic program (LCQP), just like the constrained version of Lasso
- There exists no closed form solution to this formulation
- Standard QP (LCQP) solvers[4] can be used
- Question: Are there more specific and efficient algorithms for solving SVR in this form?

[4] https://en.wikipedia.org/wiki/Quadratic_programming#Solvers_and_scripting_.28programming.29_languages

Sequential Minimial Optimization Algorithm for Solving SVR

Solving the SVR Dual Optimization Problem

- It can be shown that the objective:
  $max_{\alpha_i,\alpha_i^*} - \frac{1}{2}\sum_i \sum_j (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(x_i, x_j)$
  $-\epsilon \sum_i (\alpha_i + \alpha_i^*) + \sum_i y_i(\alpha_i - \alpha_i^*)$
- can be written as:
  $max_{\beta_i} - \frac{1}{2}\sum_i \sum_j \beta_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j) - \epsilon \sum_i |\beta_i| + \sum_i y_i \beta_i$
  s.t.
    - $\sum_i \beta_i = 0$
    - $\beta_i \in [-C, C], \forall i$
- Even for this form, standard QP (LCQP) solvers[5] can be used
- Question: How about (iteratively) solving for two β_i's at a time?
    - This is the idea of the Sequential Minimal Optimization (SMO) algorithm

[5] https://en.wikipedia.org/wiki/Quadratic_programming#Solvers_and_scripting_.28programming.29_languages

Sequential Minimal Optimization (SMO) for SVR

- Consider:
  $max_{\beta_i} -\frac{1}{2}\sum_i \sum_j \beta_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j) - \epsilon \sum_i |\beta_i| + \sum_i y_i \beta_i$
  s.t.
  - $\sum_i \beta_i = 0$
  - $\beta_i \in [-C, C], \forall i$
- The SMO subroutine can be defined as:
  1. Initialise $\beta_1, \ldots, \beta_n$ to some value $\in [-C, C]$
  2. Pick β_i, β_j to estimate closed form expression for next iterate (i.e. β_i^{new}, β_j^{new})
  3. Check if the KKT conditions are satisfied
     - If not, choose β_i and β_j that worst violate the KKT conditions and reiterate

Iterative Soft Thresholding Algorithm for Solving Lasso

Lasso: Recap Midsem Problem 2

- $$\mathbf{w}^* = \underset{\mathbf{w}}{\textbf{argmin}} \|\phi\mathbf{w} - \mathbf{y}\|^2 \;\; s.t. \;\; \|\mathbf{w}\|_1 \le \eta, \tag{1}$$

  where

  $$\|\mathbf{w}\|_1 = \left(\sum_{i=1}^{n} |w_i|\right) \tag{2}$$

- Since $\|\mathbf{w}\|_1$ is not differentiable, one can express (2) as a set of constraints

  $$\sum_{i=1}^{n} \xi_i \le \eta, \;\; w_i \le \xi_i, \;\; -w_i \le \xi_i$$

- The resulting problem is a linearly constrained Quadratic optimization problem (LCQP):

  $$\mathbf{w}^* = \underset{\mathbf{w}}{argmin} \|\phi\mathbf{w} - \mathbf{y}\|^2 \;\; s.t. \;\; \sum_{i=1}^{n} \xi_i \le \eta, \;\; w_i \le \xi_i, \;\; -w_i \le \xi_i \tag{3}$$

Lasso: Continued

- KKT conditions:

$$2(\phi^T\phi)\mathbf{w} - 2\phi^T y + \sum_{i=1}^{n}(\theta_i - \lambda_i) = 0$$

$$\beta(\sum_{i=1}^{n}\xi_i - \eta) = 0$$

$$\forall\ i,\ \theta_i(\mathbf{w}_i - \xi_i) = 0 \ \mathit{and}\ \lambda_i(-\mathbf{w}_i - \xi_i) = 0$$

- Like Ridge Regression, an equivalent Lasso formulation can be shown to be:

$$\mathbf{w}^* = \underset{\mathbf{w}}{argmin} \left\|\phi\mathbf{w} - \mathbf{y}\right\|^2 + \lambda \left\|\mathbf{w}\right\|_1 \tag{4}$$

- The justification for the equivalence between (2) and (4) as well as the solution to (4) requires *subgradient*[6].

[6] https://www.cse.iitb.ac.in/~cs709/notes/enotes/lecture27b.pdf

Iterative Soft Thresholding Algorithm (Proximal Subgradient Descent) for Lasso

- Let $\varepsilon(\mathbf{w}) = \|\phi\mathbf{w} - \mathbf{y}\|_2^2$
- **Iterative Soft Thresholding Algorithm:**
  Initialization: Find starting point $\mathbf{w}^{(0)}$
  - Let $\widehat{\mathbf{w}}^{(\mathbf{k+1})}$ be a next iterate for $\varepsilon(\mathbf{w}^k)$ computed using using any (gradient) descent algorithm
  - Compute $\mathbf{w}^{(k+1)} = \underset{\mathbf{w}}{argmin}||\mathbf{w} - \widehat{\mathbf{w}}^{(\mathbf{k+1})}||_{\mathbf{2}}^{\mathbf{2}} + \lambda\mathbf{t}||\mathbf{w}||_{\mathbf{1}}$ by:
    1. If $\widehat{w}_i^{(k+1)} > \lambda t$, then $w_i^{(k+1)} = -\lambda t + \widehat{w}_i^{(k+1)}$
    2. If $\widehat{w}_i^{(k+1)} < \lambda t$, then $w_i^{(k+1)} = \lambda t + \widehat{w}_i^{(k+1)}$
    3. 0 otherwise.
  - Set $k = k + 1$, **until** stopping criterion is satisfied (such as no significant changes in $\mathbf{w}^k$ w.r.t $\mathbf{w}^{(k-1)}$)

Next few optional slides: Extra Material on Subgradients and Justification Behind Iterative Soft Thresholding

(Optional) Subgradients

- An equivalent condition for convexity of $f(\mathbf{x})$:

$$\forall\ \mathbf{x}, \mathbf{y} \in \mathbf{dmn(f)},\ \ \mathbf{f(y)} \geq \mathbf{f(x)} + \nabla^\top \mathbf{f(x)(y - x)}$$

- $\mathbf{g_f}(\mathbf{x})$ is a *subgradient* for a function f at $\mathbf{x}$ if

$$\forall\ \mathbf{y} \in \mathbf{dmn(f)},\ \ \mathbf{f(y)} \geq \mathbf{f(x)} + \mathbf{g_f(x)}^\top \mathbf{(y - x)}$$

- Any convex (even non-differentiable) function will have a subgradient at any point in the domain!
- If a convex function f is differentiable at $\mathbf{x}$ then $\nabla f(\mathbf{x}) = \mathbf{g_f}(\mathbf{x})$
- $\mathbf{x}$ is a point of minimum of (convex) f if and only if $\mathbf{0}$ is a subgradient of f at $\mathbf{x}$

(Optional) Subgradients and Lasso

- Claim (out of syllabus): If $\mathbf{w}^*(\eta)$ is solution to (2) and $\mathbf{w}^*(\lambda)$ is solution to (4) then
  - Solution to (2) with $\eta = ||\mathbf{w}^*(\lambda)||$ is also $\mathbf{w}^*(\lambda)$ and
  - Solution to (4) with λ as solution to $\phi^T(\phi\mathbf{w} - y) = \lambda g_{\mathbf{x}}$ is also $\mathbf{w}^*(\eta)$
- The unconstrained form for Lasso in (4) has no closed form solution
- But it can be solved using a generalization of gradient descent called *proximal subgradient descent*[7]

[7] https://www.cse.iitb.ac.in/~cs709/notes/enotes/lecture27b.pdf

(Optional) Proximal Subgradient Descent for Lasso[a]

[a] https://www.cse.iitb.ac.in/~cs709/notes/enotes/lecture27b.pdf

- Let $\varepsilon(\mathbf{w}) = ||\phi\mathbf{w} - \mathbf{y}||_2^2$
- **Proximal Subgradient Descent Algorithm:**
  Initialization: Find starting point $\mathbf{w}^{(0)}$
  - Let $\widehat{\mathbf{w}}^{(\mathbf{k+1})}$ be a next gradient descent iterate for $\varepsilon(\mathbf{w}^k)$
  - Compute $\mathbf{w}^{(k+1)} = \underset{\mathbf{w}}{\mathbf{argmin}}||\mathbf{w} - \widehat{\mathbf{w}}^{(\mathbf{k+1})}||_2^2 + \lambda\mathbf{t}||\mathbf{w}||_1$ by setting subgradient of this objective to $\mathbf{0}$. This results in:
    1. If $\widehat{w}_i^{(k+1)} > \lambda t$, then $w_i^{(k+1)} = -\lambda t + \widehat{w}_i^{(k+1)}$
    2. If $\widehat{w}_i^{(k+1)} < \lambda t$, then $w_i^{(k+1)} = \lambda t + \widehat{w}_i^{(k+1)}$
    3. 0 otherwise.
  - Set $k = k + 1$, **until** stopping criterion is satisfied (such as no significant changes in $\mathbf{w}^k$ w.r.t $\mathbf{w}^{(k-1)}$)