Statistical Learning, Fall 2022-3

Class notes 12

Kernel methods

The general paradigm we have discussed, given modeling problem with $x \in \mathbb{R}^p$ low dimensional:

- Embed $x \to h(x) \in \mathbb{R}^q$ with $q >> p$.
- Fit a (possible linear model) in the high dimension $\hat{f}(x) = \sum_{j=1}^q h_j(x)\hat{\beta}_j$.
- Challenges:
  - Computational: how to fit in high dimension
  - Statistical: how to regularize in high dimension

Examples:

- Boosting:
  - Model space: all trees of given size
  - Computational trick: coordinate descent via gradient boosting
  - Regularization: sort of lasso (not discussed in class)
- DNN:
  - Model space: Not a linear model but linear combination of non-linear transformation of linear combinations...
  - Computational tricks: (stochastic) gradient descent,
  - Regularization: sort of ridge (gradient descent $\approx$ ridge, similarly dropout

Now we will discuss perhaps the primary example of this thinking, which was hugely important in ML in the past, lost some of its glamour: Kernel methods including (but not limited to) kernel SVM. We can think of the basic idea the same way, except now $x \to h(x)$ where $h_1, ...h_q$ (possibly $q = \infty$) is a basis of a Reproducing kernel Hilbert functional space (RKHS) $\mathcal{H}_K$. The space is defined indirectly through the kernel function

$$K(\cdot,\cdot) : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R} \text{ such that: } K(x,y) = \langle h(x), h(y)\rangle = \sum_{j=1}^q h_j(x)h_j(y).$$

We also naturally define for a function in $\mathcal{H}_K$, $f = \sum_j \beta_j h_j$, we naturally define $\|f\|^2_{\mathcal{H}_K} = \sum_j \beta_j^2$.

Kernel examples:

1. Linear Kernel ($q = p$): $K(x, y) = \langle x, y \rangle$. Here $\mathcal{H}_K$ is simply linear functions.
2. Polynomial kernel: $K_d(x, y) = (1 + x^t y)^d$. Here $q = \binom{p+d}{p}$ all polynomials in x_j, y_j up to degree d.
3. RBF (Gaussian) kernel: $K_\sigma(x, y) = \exp\left(-\|x - y\|^2/(2\sigma^2)\right)$. Here $q = \infty$ and we usually don't think about $h_1, ...$ explicitly, only about the kernel as measuring distance:
   - When σ is small, the kernel $K(x, \cdot)$ is very tight around x
   - When σ is big, the kernel $K(x, \cdot)$ becomes very spread and $K(x, y)$ remains big for $\|x - y\|$ big

   Since $q = \infty$ the function space $\mathcal{H}_K$ contains all nicely behaved functions regardless of σ, however we will see that the different nature of the kernel will play a role in model building (i.e. selecting among the functions in $\mathcal{H}_K$) through regularization.

Kernel machines

The Hilbert space comes with a norm attached and therefore a natural regularization term that controls that norm. Given a loss function our problem is:

$$\hat{f}_\lambda = \arg\min_{f \in \mathcal{H}_K} \sum_{i=1}^n L(y_i, f(x_i)) + \lambda \|f\|^2_{\mathcal{H}_K}.$$

We see here that the regularization term is where the specific kernel plays an important role: how functions in $\mathcal{H}_K$ are prioritized for fitting.

The most important result in this area is *the Representer theorem* (Kimmeldorf and Wahba 1970):
The optimal solution to the kernel regression problem above has the form:

$$\hat{f}_\lambda = \sum_{i=1}^n \alpha_i K(x_i, \cdot) \ , \ \ \|\hat{f}_\lambda\|^2_{\mathcal{H}_K} = \alpha^T K \alpha, \ \text{ where: } K_{ij} = K(x_i, x_j).$$

Thus we get that we can solve the problem in the n dimensional basis of the columns of K:

$$\hat{f}_\lambda = \arg\min_\alpha \sum_{i=1}^n L(y_i, \sum_{j=1}^n \alpha_j K(x_i, x_j)) + \lambda \alpha^T K \alpha.$$

For squared loss this *Kernal linear regression* problem can be nicely written:

$$\hat{f}_\lambda = \sum \hat{\alpha}_i K(x_i, \cdot) \ \text{ where: } \hat{\alpha} = \arg\min_\alpha \|\mathbb{Y} - K\alpha\|^2 + \lambda \alpha^T K \alpha,$$

a "generalized ridge regression" problem, with an algebraic solution:

$$\hat{\alpha} = (K + \lambda I_n)^{-1} \mathbb{Y}.$$

Now we can interpret what some of our kernels do in this context:

- Linear kernel: $K = \mathbb{X}\mathbb{X}^T$ and therefore $\hat{\alpha} = (\mathbb{X}\mathbb{X}^T + \lambda I_n)^{-1}\mathbb{Y}$. In this case we can easily show:

$$K\hat{\alpha} = \mathbb{X}\mathbb{X}^T(\mathbb{X}\mathbb{X}^T + \lambda I_n)^{-1}\mathbb{Y} = \mathbb{X}(\mathbb{X}^T\mathbb{X} + \lambda I_p)^{-1}\mathbb{X}^T\mathbb{Y} = \mathbb{X}\hat{\beta}_\lambda,$$

  the solution is the same as regular ridge regression!

- RBF Kernel with small σ:

$$K(x_i, x_j) = \exp\left(-\|x_i - x_j\|^2/(2\sigma^2)\right) \approx 0 \text{ when } x_i \neq x_j.$$

  Therefore the kernel regression problem is very much like penalized k-NN:

$$\|\mathbb{Y} - K\alpha\|^2 + \lambda\alpha^T K\alpha \approx \|\mathbb{Y} - \alpha\|^2 + \lambda\alpha^T\alpha.$$

The most important kernel machine was the one using the hinge loss (kernel SVM):

$$L(y, \hat{y}) = (1 - y\hat{y})_+,$$

and recall that we discussed how the sparsity of the solution $\hat{\alpha}$ helps in computing and finding solution.

For regression, the ML crowd who like loss functions that zero many $\hat{\alpha}$ came up with the ϵ-support vector regression loss, which is absolute loss with a *dontcare* region in the middle:

$$L(y, \hat{y}) = (|y - \hat{y}| - \epsilon)_+,$$

Now we can also describe kernel methods in the high dimensional modeling framework:

- Model space: all functions in the RKHS
- Computational trick: representer theorem, giving a problem of dimension n
- Regularization: RKHS norm, sort of ridge