EE445 Mod2-Lec4: Kernel Regression

What is Kernel Regression?

$\begin{bmatrix} X & \boxed{y} \end{bmatrix}$

→ prediction → lst. sqrs. , feature maps
→ classification → — " —

- We have been talking about supervised ML in the context of data fitting with least squares
- Cost function paradigm for supervised machine learning
  - Features x
  - Output/response y
  - Goal: Find $f(x)$ such that $f(x^{(i)}) \approx y^{(i)}$
  - objective/cost function $F(\theta) = \|A\theta - y\|^2$

Kernel Motivation

- But what we really want are flexible non-linear classifers/predictors!
- We can get this via a linear model using the kernel trick
- Note that feature maps are already all non-linear

$x_1 \cdot x_2$

$$x \mapsto 1, x, x^2, \ldots$$

- Yet, we want something a little more automatic that implicitly captures nonlinearities without expanding out data to many times the original size
- Kernels give us this

Kernel Trick: Starting Point

$x^{(i)}$: feature vectors

- [A2]:

$$\theta = \sum_{i=1}^{m} \alpha_i x^{(i)} \quad \text{for some} \quad \alpha_1, \ldots, \alpha_m \in \mathbb{R}$$

  i.e., θ is in the span of the feature vectors

- We will see shortly how to find these α_i's

Kernel Trick: Linear Regression

- [A2]: $\theta = \sum_{i=1}^{m} \alpha_i x^{(i)}$ for some $\alpha_1, \ldots, \alpha_m \in \mathbb{R}$

$$f(x) = \theta^T x = \left(\sum_{i=1}^{m} \alpha_i x^{(i)}\right)^T x = \sum_{i=1}^{m} \alpha_i \cdot (x^{(i)})^T x$$

$$= \sum_{i=1}^{m} \alpha_i K(x^{(i)}, x)$$

- **Kernel function**: $K(x, z) = x^\top z$
- Predictions only depend on training data through kernel function which is just a dot product.

Linear Regression: Objective Function

$K = K^T$

- [A2]: $\theta = \sum_{i=1}^{m} \alpha_i x^{(i)}$ for some $\alpha_1, \ldots, \alpha_m \in \mathbb{R}$
- The predictor has the form

$K(z, x) = x^T z$

$$f(x) = \sum_{i=1}^{m} \underbrace{\alpha_i K(x^{(i)}, x)}$$

$\langle x^{(i)}, x^{(j)} \rangle$

$K = \left[K(x^{(j)}, x^{(i)}) \right]$

- The objective function has the form

$$\frac{1}{2} \sum_{i=1}^{m} \big(\underbrace{f(x^{(i)})}_{\theta^T x^{(i)}} - y^{(i)} \big)^2 = \frac{1}{2} \sum_{i=1}^{m} \Big(\sum_{j=1}^{m} \alpha_j K(x^{(j)}, x^{(i)}) - y^{(i)} \Big)^2 =: F(\alpha)$$

- Objective function only depends on training data through kernel function which is just dot products
- Choose α by minimizing $F(\alpha)$

$\nabla F(\alpha) = 0$

Kernel Trick: Take-Aways

- Predictor and objective only depend on training data through the kernel which is itself just dot products
- Hence, if we only have the ability to do dot product operations, then we can still suprisingly train a model (i.e., find a prediction of y)

Kernelized Linear Regression

$$f(x) = \theta^\top x$$

- Rewrite linear regression as a different linear regression model:

$$f(x) = \sum_{i=1}^{m} \alpha_i K(x^{(i)}, x) = \alpha^\top k(x)$$

where

$$\alpha^\top = \begin{bmatrix} \alpha_1 & \cdots & \alpha_m \end{bmatrix} \quad \text{and} \quad k(x) = \begin{bmatrix} K(x^{(1)}, x) \\ \vdots \\ K(x^{(m)}, x) \end{bmatrix}$$

- i.e., we map x to a new "feature vector" $k(x)$ (= kernel evaluation between x and each training feature vector).

What happens to original data matrix X under this mapping?

$$\begin{bmatrix} - & x^{(1)} & - \\ & \vdots & \\ - & x^{(n)} & - \end{bmatrix} \xrightarrow{K(\cdot,\cdot)} K$$

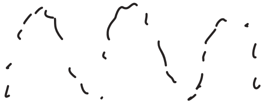
- Recall: i–th row of X is i–th feature vector $x^{(i)}$
- Kernel Matrix: new "data matrix" K such that the i–th row contains dot products between $x^{(i)}$ and every other training point:

$$K_{ij} = K(x^{(i)}, x^{(j)}) = (x^{(i)})^\top x^{(j)}$$

$$\begin{bmatrix} (x^{(1)})^T x^{(1)} & (x^{(1)})^T x^{(2)} & \cdots \\ & & \\ & & \end{bmatrix}$$

- Sometimes this is called the Kernel Trick.
- Take-Away: you can learn an equivalent linear model using the kernel matrix in place of the original data matrix.
- this equivalence is only exact without regularization (I will talk about this shortly)

Nonlinear Feature Maps



$f(x) = \theta^T x$, $K(x, z) = x^T z$

Mod2-Lec2

- Suppose we want to do feature mappings before learning such as

$$f(x) = \theta^\top \phi(x), \quad \phi : \mathbb{R}^n \to \mathbb{R}^p$$

- **Kernel corresponding to ϕ:** To solve the learning problem and make predictions, we only need to be able to compute

$$K(x, z) = \phi(x)^\top \phi(z)$$

$$\begin{bmatrix} & & \\ & \phi(x^{(i)})^T \phi(x^{(j)}) & \\ & & \end{bmatrix}$$

Examples: Polynomial Kernel

$\underbrace{\phi(x)^T\phi(z)}$

Note: we can often compute kernel without actually doing the expansion

- Consider $K(x,z) = (x^\top z)^2$
- **What is $\phi : \mathbb{R}^2 \to \mathbb{R}^4$?**

$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \phi(x) \in \mathbb{R}^4$

$$\phi(x) = (x_1^2, x_1x_2, x_2x_1, x_2^2)$$

- Check:

$$K(x,z) = (x^T z)^2 = \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}\right)^2 = (x_1z_1 + x_2z_2)^2 = x_1^2z_1^2 + 2z_1x_1 \cdot x_2z_2 + z_2^2x_2^2$$

$$\phi(x)^T\phi(z) = \begin{bmatrix} x_1^2 \\ x_1x_2 \\ x_2x_1 \\ x_2^2 \end{bmatrix}^T \begin{bmatrix} z_1^2 \\ z_1z_2 \\ z_2z_1 \\ z_2^2 \end{bmatrix} =$$

Examples: Polynomial Kernel

$\phi(x)^\top \phi(z)$

$x \in \mathbb{R}^n$

Note: computational complexity is lower

- Consider $K(x,z) = (x^\top z + 1)^2$
- **What is** $\phi : \mathbb{R}^2 \to \mathbb{R}^4$**?**

$$\phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_1x_2, x_2x_1, x_2^2)$$

- complexity of $\phi(x)^\top \phi(z)$: $O(n^2)$
- complexity of $x^\top z$: $O(n)$
- complexity of $(x^\top z + 1)^2$: $O(n)$
- If using kernel trick, can implement a non-linear feature expansion at no additional cost
- **More general**: $K(x,z) = (x^\top z + 1)^d$
  - complexity of computing corresponding features with ϕ: $O(n^d)$
  - complexity of computing K: $O(n)$

Example: Gaussian Kernel

$$K(x, z) = \exp(-\gamma\|x - z\|^2)$$

Some observations:

- non-linear kernel with a lot of flexibility
- corresponds to an infinite dimensional ϕ—i.e., cannot implement the corresponding feature mapping ϕ.

In Practice: Regularization

$\theta = \sum \alpha_i x^{(i)}$

$(A^T A)^{-1}$

- We often introduce a regularization term in practice:

$K(x,z) = \phi^T(x)\phi(z)$

$$\left(y^{(k)} - \theta^T \phi(x^{(k)})\right)^2$$

$$F(\theta) = \frac{1}{2}\sum_{k=1}^{m}(\theta^\top\phi(x^{(k)}) - y^{(k)})^2 + \frac{\lambda}{2}\|\theta\|_2^2 \rightarrow \text{argmin is } \hat{\theta}$$

- **why?**: Regularization improves the conditioning of the problem and reduces the variance of the estimates.
- Taking derivatives and setting them to zero we have

$$\sum_{k=1}^{m}\left(y^{(k)} - \theta^T \phi(x^{(k)})\right)\phi(x^{(k)})^T = \lambda\theta$$

$$\Rightarrow \hat{\theta} = \left(\lambda I + \sum_{k=1}^{m}\phi(x^{(k)})\phi(x^{(k)})^T\right)^{-1}\sum_{j=1}^{m}\phi(x^{(j)})y^{(j)}$$

Deriving the α-dependent regularization term

Recall that we converted $F(\theta)$ to a cost in terms of α. We will do the same thing for the regularized cost.

- [A2]: $\theta = \cancel{\sum_{i=1}^m \alpha_i x^{(i)}}$ for some $\alpha_1, \ldots, \alpha_m \in \mathbb{R}$

$$\sum_{i=1}^m \alpha_i \phi(x^{(i)}) = \theta$$

$$\|\theta\|^2 = \theta^T\theta = \left(\sum_{i=1}^m \alpha_i \phi(x^{(i)})\right)^T\left(\sum_{i=1}^m \alpha_i \phi(x^{(i)})\right)$$

$$= \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j \underbrace{\phi(x^{(i)})^T \phi(x^{(j)})}_{\underbrace{K(x^{(i)}, x^{(j)})}} = \alpha^T K \alpha$$

$$K_{ij} = K(x^{(i)}, x^{(j)})$$

Kernelized Regression Regularized Cost (Ridge Regression)

Dan Sheldon
CMU

LASSO Regression

$$F(\alpha) = \frac{1}{2}\|K\alpha - y\|^2 + \frac{\lambda}{2}\alpha^\top K\alpha$$

$\| \cdot \|_1$

$$K(K\alpha - y) + \lambda K\alpha = 0 \iff K(K + \lambda I)\alpha = Ky$$

$$\iff \alpha = (K + \lambda I)^{-1} y$$

$$f(x) = \left(\sum_{i=1}^{m} \alpha_i \phi(x^{(i)})\right)^\top \phi(x)$$

- Choose λ via cross validation!