Stat 205: Introduction to Nonparametric Statistics

Lecture 11: Kernelization/Feature Maps

Instructor David Donoho; TA: Yu Wang

Curse of dimensionality

- High-dimensional space is mostly empty
- Even Nearest Neighbors are far apart
- Example: $x_i = (x_{i,1}, \ldots, x_{i,p})$
- Entries uniform

$$x_i \sim_{iid} unif[0,1]$$

- Maximum coordinate distance (ℓ_∞)

$$d(x_1, x_2) = \max_{k=1}^{p} |x_{1k} - x_{2k}|$$

- Chance of two specific points within δ-distance

$$\begin{aligned} P\{d(x_1, x_j) < \delta\} &= P(\cap_{k=1}^{p} \{|x_{1k} - x_{2k}| < \delta\}) \\ &= (1 - 2\delta)^p \end{aligned}$$

- Chance x_1 has no neighbors within δ-distance

$$P\{\min_{j=2}^{n} d(x_1, x_j) > \delta\} = P(\cap_{j=2}^{n} \{d(x_1, x_j) > \delta\}) = (1 - (1 - 2\delta)^p)^n.$$

- Can vary n,p, δ in these formulas (Homework).

Feature Maps

- Terms of art: Kernel- { Regression, NN, 'Trick' }
- Data $Y = (y_i)_{i=1}^n$; "response", "target"
- Data $X = (x_i)_{i=1}^n$; $x_i \in \mathbf{R}^p$; predictors.
- Feature map: x_i inputs, $\phi(x_i)$ features.

$$\phi : \mathbf{R}^p \mapsto \mathbf{R}^m$$

- Dimension Raising: $m \gg p$:
  many more synthetic features than input dimension.
- After raising dimension can use any standard method:
  - Linear regression
  - Logistic regression classifier
  - Nearest neighbor regression, classifier

Feature Map Examples

- $p = 1$ Step functions

$$\phi(x) = (1_{[a_1,a_2]}(x), 1_{(a_2,a_3]}(x), \ldots, 1_{(a_{m-1},a_m]}(x))^T$$

- $p = 1$ Polynomial features

$$\phi(x) = (1, x_1, \ldots, x_1^{m-1})^T$$

- $p > 1$ Quadratic multivariate polynomial features

$$\phi(x) = (1, x_1, \ldots, x_p, x_1^2, ..., x_p^2, x_1x_2, \ldots, x_{p-1}x_p)^T$$

Here $m = 1 + p + p(p+1)/2$.

- $p > 1$ Random features:

$$\phi(x) = (\sigma(w_1^T x), \ldots, \sigma(w_m^T x))$$

  - $\sigma()$ specific nonlinearity, eg relu
  - (w_j) direction vectors in $\mathbf{R}^p$

Kernel Regression

- $\phi : \mathbf{R}^p \mapsto \mathbf{R}^m$ feature map
- Linear predictor $\mu(x;\theta) = \phi^T(x)\theta$.
- $n \times m$ matrix of features
$$\Phi = [\phi(x_i)^T]_{i=1}^n$$
- Squared error Loss
$$PMSE^{train}(\theta) = L(\theta) = \|Y - \Phi\theta\|_2^2$$
- Least-squares estimate
$$\hat{\theta} = \text{argmin} L(\theta)$$
- Least-squares solution
$$\hat{\theta} = (\Phi^T\Phi)^\dagger \Phi^T Y$$
  - M has full column rank $M^\dagger v = M^{-1} v$
  - M rank deficient: $M^\dagger v$ solves minimum norm problem

$$\min \|\theta\|_2^2 \text{ subject to } M\theta = v.$$

- Setting of most interest: $m \gg p$.
  - Problem is underdetermined
  - Generalized inverse is used

Surprises in Dimension-Raising, 1/3

In the $m \gg n$ setting, we often can do all computations in dimension $n = min(m, n)$!

- SVD $\Phi = UDV^T$,
  - Φ: $n \times m$,
  - U, D: $n \times n$
  - V: $m \times n$
- Define $n \times n$ kernel matrix

$$K = \Phi\Phi^T = [\phi(x_i)^T \phi(x_j)]_{i,j=1}^n$$

  Kernel properties

$$K = \Phi\Phi^T = UD^2U^T; \qquad K^\dagger = UD^{-2}U^T.$$

- Define $m \times m$ Gram matrix

$$G = \Phi^T\Phi = VD^2V^T; \qquad G^\dagger = VD^{-2}V^T.$$

- **Special Property** relating G, Φ and K.

$$\Phi G^\dagger \Phi^T = UDV^T VD^{-2}VV^T DU^T = UU^T$$

  Here U depends on Φ since $\Phi = UDV^T$
  Here U depends on K since $K = UD^2U^T$
  Doesn't (need to) involve Φ.

Zero-Training Loss

Typically when $m \gg n$ zero training error:

- $\hat{\mu}(x) = \mu(x; \hat{\theta}) = \phi^T(x)\hat{\theta}$.

$$\hat{Y} = [\hat{\mu}(x_i)^T]_{i=1}^n$$

$$PMSE^{train}(\hat{\theta}) = \mathcal{L}(\hat{\theta}) = n^{-1}\|Y - \hat{Y}\|_2^2 = 0.$$

- Consider predictions

$$\hat{Y} = \Phi\hat{\theta} = HY$$

where [from previous slide]

$$H = \Phi(\Phi^T\Phi)^{-1}\Phi^T = \Phi G^{\dagger}\Phi^T = UU^T$$

In the full column rank case, $UU^T = I_n$, the $n \times n$ identity, so

$$\hat{\mu} = \Phi\hat{\theta} = HY = Y$$

'Zero training loss' in the full column rank case.

Regularized Kernel Regression, 1

- In the case where $m \gg n$ we generally get $\hat{\mu} = Y$:
  predictions just reproduce the data in-sample.
- Statistical Theory tells us: if any noise in the data, then $\hat{\mu} = Y$ is *overfit*.
- Indeed $\hat{\mu} = Y$ means:
  our predictions in-sample are exactly *as noisy as* the data;
  when used out of sample on test data, $\Phi(X^{test})$ actually *noisier than* the data.
- So to avoid making hyper-noisy predictions, we *regularize*
  Regularized coefficients:

$$\begin{aligned} \hat{\theta}_\lambda &= (\Phi^T\Phi + \lambda I)^{-1}\Phi^T Y \\ &= (G + \lambda I)^{-1}\Phi^T Y \end{aligned}$$

  Regularized predictions:

$$\begin{aligned} \hat{\mu}_\lambda &= \Phi\hat{\theta}_\lambda \\ &= \Phi(G + \lambda I)^{-1}\Phi^T Y \end{aligned}$$

- You have probably seen this before, in the guise of *ridge regression*..
  The linear algebra is the same; but with a new interpretation.
- Penalized squared error loss:

$$\mathcal{L}_\lambda(\theta|X, Y) = \frac{1}{n}\sum_i (y_i - \phi(x_i)^T\theta)^2 + \lambda\|\theta\|_2^2$$

  the second term is called a *penalty term*.
- Regularized coefficients $\hat{\theta}_\lambda$ listed earlier are minimizing the penalized squared-error loss.

Surprises in Dimension-Raising, 2/3

In the $m \gg n$ setting, we often can do all computations in dimension $n = min(m, n)$!

- Regularized predictions:

$$\begin{aligned} \hat{\mu}_\lambda &= \Phi\hat{\theta}_\lambda \\ &= \Phi(G + \lambda I)^{-1}\Phi^T Y \end{aligned}$$

- The matrix 'sandwich' $\Phi(G + \lambda I)^\dagger \Phi^T$ appears to go into m dimensions: G is $m \times m$.
- Actually we saw a few slides ago a special property:

$$\Phi G^\dagger \Phi^T = UDV^T VD^{-2}V^T UDU^T = UU^T$$

  all terms on RHS: n times n.

- Also

$$\Phi(G + \lambda I)^\dagger \Phi^T = UDV^T V(U^T UDV^T = U(D^2(D^2 + \lambda I)^{-1})U^T$$

  again all terms on RHS: n times n.

- We can compute full regularization path: $\hat{\mu}_\lambda = U^T diag(\frac{d_i^2}{d_i^2+\lambda})U^T Y$.

We can compute many things without ever going into m dimensions!

Fast Generalized Cross-Validation

- Sum of squared errors as a function of λ

$$RSS(\lambda) = \|Y - H_\lambda Y\|_2^2 = \sum_{i=1}^{n}(y_i - \hat{\mu}_\lambda(x_i))^2$$
$$= \|diag(\frac{\lambda}{d_i^2 + \lambda})\tilde{Y}\|_2^2$$

  where $diag()$ means diagonal $n \times n$ matrix and $D = diag(d_i)$ and $\tilde{Y} \equiv U^T Y$. Hence, with a startup cost of one single SVD, we get $O(n)$ cost per evaluation.

- Generalized Cross-validation score:

$$GCV(\lambda) = \frac{1}{1 - tr(H_\lambda)/n} \cdot \|Y - H_\lambda Y\|_2^2$$

- Incremental cost $O(n)$ per evaluation of GCV:
  - $tr(H_\lambda) = \sum_{i=1}^{n} \frac{\lambda}{d_i^2+\lambda}$
  - $\|Y - H_\lambda Y\|_2^2 = \sum_{i=1}^{n} \tilde{y}_i^2 \cdot \frac{\lambda^2}{(d_i^2+\lambda)^2}$
- Evaluate on grid.

We can compute many things in $O(n)$ without ever going into m dimensions!

Explicit Kernel Functions

- Given feature map $\phi(x)$, define $\mathcal{K}(x, z) = \phi(x)^T \phi(z)$
- Often $\mathcal{K}(x, z)$ computable without going into m dimensions!
- Example: Quadratic polynomial features

$$\begin{aligned}
\phi(x)^T \phi(z) &= 1 + \sum_{i=1}^{p} x_i z_i + \sum_{i,j=1}^{p} x_i x_j z_i z_j \\
&= 1 + x^T z + (\sum_{i=1}^{p} x_i z_i)(\sum_{j=1}^{p} x_j z_j) \\
&= 1 + x^T z + (x^T z)^2 \\
&= \mathcal{K}_2(x, z),
\end{aligned}$$

  say.

- Example: Cubic Polynomial Features

$$\phi(x)^T \phi(z) = 1 + x^T z + (x^T z)^2 + (x^T z)^3 = \mathcal{K}_3(x, z),$$

  say.

- In these cases the feature map dimension $m \gg p$, but calculating $\mathcal{K}$ takes $O(\min(n, p))$ operations.
- We can leverage this to save on regularized kernel regression

Surprises in Dimension-Raising, 3/3

- In the $m \gg n$ setting, we often can do all computations in dimension $n = min(m, n)$!

$$\hat{\mu}_\lambda \quad = \quad \Phi(G + \lambda I)^{-1}\Phi^T Y$$

- The matrix 'sandwich' $\Phi(G + \lambda I)^\dagger \Phi^T$ appears to go into m dimensions: G is $m \times m$.
- Actually we saw a few slides ago a special property:

$$\Phi(G + \lambda I)^\dagger \Phi^T = UDV^T V(U^T UDV^T = U(D^2(D^2 + \lambda I)^{-1})U^T$$

all terms on RHS: n times n.

- Another way to put it:

$$\Phi(G + \lambda I)^\dagger \Phi^T \quad = \quad K(K + \lambda I)^{-1}$$

appears to involve G which is m times m but is actually a function of the n times n matrix K!

- Indeed

$$K = UD^2U^T.$$

- Now when there is an explicit Kernel $\mathcal{K}(x, z)$

$$K_{ij} = \mathcal{K}(x_i, x_j)$$

- Hence we can obtain the matrix K without ever going into m dimensions!

With an explicit Kernel compute many things without ever going into m dimensions!

Kernel Examples

So we often specify $\mathcal{K}$ only; never explicitly construct our feature map Φ!
Examples of $\mathcal{K}$ we can specifiy

- Power Kernels, $\ell = 1, 2, \ldots$.
$$\mathcal{K}_\ell(x, z) = (x^T z)^\ell$$
- Gaussian Kernel (often called Radial Basis Function).
$$\mathcal{K}_G(x, z) = \exp(-\frac{\|x - z\|_2^2}{2\sigma^2})$$
- Laplacian Kernel (often called Radial Basis Function).
$$\mathcal{K}_L(x, z) = \exp(-\alpha\|x - z\|_2)$$
- Sums of Kernels with positive coefficients
$$\mathcal{K}(x, z) = c_1\mathcal{K}_1(x, z) + c_2\mathcal{K}_2(x, z)$$
$c_i > 0$, $i = 1, 2$.
- Products of Kernels with positive integer coefficients
$$\mathcal{K}(x, z) = \mathcal{K}_1(x, z)^{\ell_1} \cdot \mathcal{K}_2(x, z)^{\ell_2}$$
$\ell_i \in \{1, 2, 3, \ldots\}$, $i = 1, 2$.
- Many others.

Standing behind explicit kernel is implicit feature map

Definition. *Let $\mathcal{X} \subset \mathbf{R}^p$. The symmetric function $\mathcal{K}(x, z)$ on $\mathcal{X} \times \mathcal{X}$ is called PSD if, for every n and every $(c_i)_{i=1}^n$ with $c_i \in \mathbf{R}$ and every $(x_i)_{i=1}^n$ with $x_i \in \mathcal{X}$:*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(x_i, x_j) \geq 0.$$

Mercer's Theorem:
if K is positive semidefinite, then $K(x, z) = \phi(x)^T \phi(z)$ for some feature map $\phi(x)$ from $\mathbf{R}^p$ into $\mathbf{R}^m$ for some $m \geq n$.

There are many many explicit kernels! The feature map may surprise you!

Examples of Implicit Feature maps associated with PD Kernels

domain $\mathcal{X}$	kernel	feature span	dim m
$\mathbf{R}^p$	$\mathcal{K}_2(x,z) = 1 + x'z + (x'z)^2$	quadratic polyn's on $\mathbf{R}^p$	$1 + p + \frac{p(p+1)}{2}$
$\mathbf{R}^p$	$\mathcal{K}_G(x,z) = \exp(-\alpha\|x-z\|_2^2)$	all polyn's on $\mathbf{R}^p$	∞
$[0,1]$	$\mathcal{K}_S(u,v) = (1-\max(u,v))\min(u,v)$	polyn's vanishing at $0,1$	∞
$\mathcal{D} = \{\|z\| \le 1\}$	$\mathcal{K}_D(z,w) = \frac{1}{1-\bar{z}w}$	all polyn's in z	∞

Often many equivalent feature maps, related by

$$\psi = \Omega\phi$$

where Ω is an orthogonal mapping of feature space $\mathbf{R}^m$.
Only the span and normalization are fixed.

Reproducing Kernels

- Reproducing Kernel Hilbert space $\mathcal{H}$
- Kernel Membership property:

$$k_x(\cdot) \equiv \mathcal{K}(x, \cdot) \in \mathcal{H}, \qquad \forall x \in \mathcal{X}.$$

- Reproducing property:

$$(f, k_x)_{\mathcal{H}} = f(x), \qquad f \in \mathcal{H}, \qquad \forall x \in \mathcal{X}.$$

- **Theorem.** Every positive definite kernel $\mathcal{K}$ induces a unique RKHS $\mathcal{H}_{\mathcal{K}}$

Representer Theorem

- Reproducing Kernel Hilbert space $\mathcal{H}$
- Penalized Least-squares

$$\mathcal{L}_\lambda(f) = n^{-1}\|Y - (f(x_i))\|_2^2 + \lambda\|f\|_{\mathcal{H}}^2$$

- Optimizer

$$\hat{f}^* = \text{argmin}_f \mathcal{L}_\lambda(f)$$

- Every solution

$$\hat{f}^*(\cdot) = g(\cdot) + \sum_i \theta_i^* \mathcal{K}(\cdot, x_i)$$

g in null space of $\mathcal{H}$: $\|g\|_{\mathcal{H}} = 0$.

- Coefficients determined by

$$\theta^* = \text{argmin}_\vartheta\{\|Y - K\vartheta\|_2^2 + \lambda\vartheta^T K\vartheta\}$$

- Solution

$$\theta^* = (K^T K + \lambda K)^\dagger K^T Y$$

$n \times n$ system!

Example:

- Domain $\mathcal{X} = [0, 1]$
- Kernel $\mathcal{K}_S(u, v) = (1 - max(u, v))min(u, v)$
- Hilbert space: Sobolev space W_0^1.
- Representer:

$$k_u(v) = \begin{cases} (1-u)v & v < u \\ (1-v)u & v > u \end{cases}.$$

  Piecewise linear in v with one knot at $v = u$!
- Optimal penalized-MSE solution

$$\hat{f}_\lambda^*(v) = a + bv + \sum_i \theta_i^* k_{x_i}(v)$$

  Piecewise linear with one knot at every x_i!
- Penalization by W_0^1 (semi-) norm

$$\|f\|_{W_0^1}^2 = \int_0^1 |f'(t) - \bar{f}'|^2 dt.$$

  where $\bar{f}' = \int_0^1 f'(t)dt$.
- Penalized kernel regression based on the Sobolev Kernel $\mathcal{K}_S$ finds the linear spline which best balances a tradeoff between smoothness $\|f\|_{W_0^1}^2$ and lack-of-fit $\|Y - f\|_2^2$.

Conclusion

- Modern machine learning involves implicitly or explicitly *feature manufacturing*
- Dimension m of feature map can be far greater than $\max(n, p)$ of dataset
- Two approaches:
  - Explicit: specify feature map $\phi(x)$
  - Implicit: specify kernel $\mathcal{K}(x, z)$
- Advantages of Implicit [Kernel]:
  - Computations always in dimension n.
  - Explicit kernel representation $\hat{\mu}(z) = \sum_i \theta_i K(z, x_i)$
- Explicit, massive learned feature maps, [Deepnet] $m \gg n$.
  - Advantage: outperform kernel methods
  - Disadvantage: costly in GPU etc.