IntroML - Lecture Notes Week 5

Ruben Schenk, ruben.schenk@inf.ethz.ch

April 6, 2022

1 Kernel Methods

1.1 Improving Polynomial Regression

1.1.1 Computational Complexity

How large is p to express a degree m polynomial for $x \in \mathbb{R}^d$? We can count the number of *monomials* of degree at most m:

Given any m-th degree monomial $1^{\alpha_0} x_{[1]}^{\alpha_1} \cdots x_{[d]}^{\alpha_d}$ with $\alpha_i \in \mathbb{N}$ and $\sum_{j=0}^{d} \alpha_j = m$, we can encode it as a $d+m$ binary string with d zeros and m ones:

Build the string: Start with empty string s. For $l = 0, ..., d$ do:

1. If $\alpha_l \geq 1$, append α_l ones to the string, i.e. $s \leftarrow (s, 1, ..., 1)$. If $l < d$, also add a zero, i.e. $s \leftarrow (s, 0)$.
2. Else if $\alpha_l = 0$, append $s \leftarrow (s, 0)$.

This gives you $m+1$ consecutive chunks of 1's – the number of 1's in the i-th chunk is the power of x_i.

Example: Let $d = 5$ and $m = 7$:

- $x_{[1]}^2 x_{[2]} x_{[3]}^3 \rightarrow 1011010011100$

Hence, each monomial corresponds to picking a set of m from $d+m$ numbers, yielding a total number of $p = \binom{d+m}{m} \simeq \frac{(d+m-1)\cdots d}{m \cdots 1}$ which turns into:

$$p = \begin{cases} \mathcal{O}(d^m) & \text{for large enough } d, \\ \mathcal{O}(m^d) & \text{for large enough } m. \end{cases}$$

For mth degree polynomial features, the total training set $\{(\phi(x_i), y_i)\}_{i=1}^n$ is of size $\mathcal{O}(nd^m)$.

1.1.2 Kernel Trick

For high-dimensional data in practice, e.g. $d \sim 10^5$ and $n \sim 10^5$, even choosing $m = 3$ to fit 3rd degree polynomials yields $\mathcal{O}(nd^m) \sim 10^{20}$ complexity. This is prohibitive from both the memory and the computational perspective.

The **kernel trick** is given, in short, as follows:

Kernel trick:

1. Save memory by noting that the training loss minimizer only depends on the feature vectors via their inner products (for polynomials: $\mathcal{O}(nd^m) \to \mathcal{O}(n^2)$ memory reduction!)
2. We can sometimes more efficiently compute the inner products, i.e. for polynomials of monomials, reduce polynomial to linear $\mathcal{O}(n^2 d^m) \to \mathcal{O}(n^2(d+m))$

Step 1: Minimizer only depends on inner product Remember for parameterized function $F_w = \{f : f_w \text{ with } w \in \mathbb{R}^p\}$, the minimizer $\arg\min_{f \in F} \frac{1}{n}\sum_{i=1}^n l(y_i, f(x_i))$ can be written as $\hat{f} = f_{\hat{w}}$ with $\hat{w} = \arg\min_{w \in \mathbb{R}^p} \frac{1}{n}\sum_{i=1}^n l(y_i, f(x_i))$.

Claim 1: Among the global minimizers in $\arg\min_{w \in \mathbb{R}^p} \frac{1}{n}\sum_{i=1}^n l(y_i, w^T\phi(x_i))$, on of them:

1. has the form $\hat{w} = \Phi^T\hat{\alpha}$ with $\hat{\alpha} \in \mathbb{R}^n$, such that $\hat{f}(x) = \sum_{i=1}^n \hat{\alpha}_i \langle \phi(x_i), \phi(x)\rangle$, and where
2. $\hat{\alpha}$ only depends on x_i via the inner products $\langle \phi(x_i), \phi(x_j)\rangle$ for $i, j = 1, ..., n$.

So far, we reduced the problem to $\hat{\alpha} = \arg\min_{\alpha \in \mathbb{R}^n} \frac{1}{n}\sum_{i=1}^n l(y_i, \alpha^T\Phi\phi(x_i)) =: \arg\min_{\alpha \in \mathbb{R}^n} \tilde{L}(\alpha)$.
We can use the inner products of the features to define a symmetric **kernel function:**

$$k : X \times X \to \mathbb{R},\ k(x, z) = \langle \phi(x), \phi(z)\rangle,$$

and *kernel matrix* $K \in \mathbb{R}^{n \times n}$ with $K = \Phi\Phi^T$ and $K_{ij} = k(x_i, x_j)$. The loss $\tilde{L}(\alpha)$ only depends on the entries of K, hence, we only need to keep memory of $\mathcal{O}(n^2)$ bits.

Step 2: Efficient Computation For the feature vector $\phi(x) = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2]$, the inner product reads:

$$\langle \phi(x), \phi(z)\rangle = 1 + 2x_1z_1 + 2x_2z_2 + 2x_1z_1x_2z_2 + x_1^2z_1^2 + x_2^2z_2^2 = (1 + \langle x, z\rangle)^2 =: k(x, z).$$

More generally, for appropriate scaling of monomials and cross terms, the inner product of m-th degree polynomial features in any dimension d can be written as:

$$\langle \phi(x), \phi(z)\rangle = k(x, z) = (1 + \langle x, z\rangle)^m$$

1.2 Kernelized Regression for Polynomials

Linear	Kernelized
$\hat{w} = \text{argmin}_w \|\|y - Xw\|\|^2 = X^\top\hat{\alpha}$	$\hat{w} = \text{argmin}_w \|\|y - \Phi w\|\|^2 = \Phi^\top\hat{\alpha}$
search in subspace $span(X^\top)$ $\to$ $\hat{\alpha} = \text{argmin}_\alpha \|\|y - XX^T\alpha\|\|^2$	$\hat{\alpha} = \text{argmin}_\alpha \|\|y - \Phi\Phi^T\alpha\|\|^2 = \text{argmin}_\alpha \|\|y - K\alpha\|\|^2$
$\to \hat{\alpha} = (XX^\top)^\dagger y$	$\to \hat{\alpha} = (\Phi\Phi^\top)^\dagger y = K^\dagger y$
$\hat{f}(x) = \hat{w}^\top x = y^\top(XX^\top)^\dagger Xx$	$\hat{f}(x) = \hat{w}^\top\phi(x) = y^\top K^\dagger\Phi\,\phi(x)$

can check that $\hat{w} = (X^\top X)^\dagger X^\top y = X^\top(XX^\top)^\dagger y$

Replace X by Φ, x by $\phi(x)$, and $XX^\top$ by $\Phi\Phi^\top = K$ kernel matrix with $K_{i,j} = k(x_i, x_j)$

Claim 2 (Representer Theorem): The global minimizer(s) $\arg\min_{w \in \mathbb{R}^p} \frac{1}{n}\sum_{i=1}^n l(y_i, w^T\phi(x_i)) + \lambda||w||^2$ have the form $\hat{w} = \Phi^T\hat{\alpha}$ with some $\hat{\alpha} \in \mathbb{R}^n$, such that $\hat{f} = \sum_{i=1}^n \hat{\alpha}_i\langle \phi(x_i), \phi(x)\rangle$.

Using the kernel trick, for ridge regression $\frac{1}{n}||y - \Phi w||^2 + \lambda||w||^2$, using $w = \Phi^T\alpha$ and $K = \Phi\Phi^T$, we obtain

$$\frac{1}{n}||y - \Phi w||^2 + \lambda||w||^2 = \frac{1}{n}||y - \Phi\Phi^T\alpha||^2 + \lambda||\Phi^T\alpha||^2 = \frac{1}{n}||y - K\alpha||^2 + \lambda\alpha^T K\alpha.$$

1.3 Infinite-Dimensional Features & RBF Kernels

1.3.1 Infinite-Dimensional Feature Maps

Finite-dimensional	Infinite-dimensional
$\phi(x) = \left(\phi_1(x), \dots, \phi_p(x)\right) \in R^p$	$(\phi_1(x), \phi_2(x), \dots) \in \ell^2$ with $\sum_{j=1}^{\infty} \phi_j(x)^2 < \infty$
$k(x,z) = \langle\phi(x), \phi(z)\rangle = \sum_{j=1}^{p} \phi_j(x)\phi_j(z)$	$k(x,z) = \langle\phi(x), \phi(z)\rangle_{\ell^2} = \sum_{j=1}^{\infty} \phi_j(x)\phi_j(z)$
$F = \{f \mid f(x) = \sum_{j=1}^{p} w_j\phi_j(x) = \langle w, \phi(x)\rangle, w \in R^p\}$	$F = \{f \mid f(x) = \sum_{j=1}^{\infty} w_j\phi_j(x) = \langle w, \phi(x)\rangle_{\ell^2}, w \in \ell^2\}$

$p \to \infty$

$$||w||_{\ell^2}^2 = \sum_{j=1}^{\infty} w_j^2 < \infty$$

For every $x, \phi(x)$ is a countable sequence. For every j, ϕ_j is a function!

We can write $f(x) = \langle w, \phi(x)\rangle_{l^2}$ and again define $S = \text{span}\{\phi(x_1), \dots, \phi(x_n)\} \subset l^2$. For ridge regression, by orthogonal decomposition, we again have $||w||^2 = ||\Pi_S w||^2 + ||\Pi_{S^\perp} w||^2$, and hence can use the same arguments to obtain:

$$||y - \Phi w||^2 + \lambda||w||^2 = ||y - K\alpha||^2 + \lambda\alpha^T K\alpha$$

1.3.2 Valid Kernels & Compositions

But how do we find infinite dimensional features ϕ such that k is easy to compute? No general recipe! But we can do some reverse engineering. We start with a computable kernel first and then find the features:

1. Define some bivariate function $x : X \times X \to \mathbb{R}$ with $X \subset \mathbb{R}^d$
2. Then, find ϕ such that $\langle\phi(x), \phi(y)\rangle = k(x, y)$ where $\phi(x) \in l^2$

For which k do such ϕ exist? The necessary conditions for k to be inner products of feature vectors are as follows:

- Kernel functions must be symmetric because they are inner products ($k(x, z) = k(z, x)$)
- The kernel function k is positive-semidefinite, i.e. the kernel matrix K with $(K)_{ij} = k(x_i, x_j)$ is positive-semidefinite for any choice of inputs $x_1, \dots, x_n$ and any $n \in \mathbb{N}$

Inner product kernels $k(x, z) = h(\langle x, z\rangle)$ are positive-semidefinite if the Taylor series $h(\langle x, z\rangle) = \sum_{j=0}^{\infty} a_j(\langle x, z\rangle)^j$ has $a_j \geq 0$ for all $j \geq 0$.

Radial Basis Functions (RBF) kernel $k(x, z) = h(||x - z||)$ are often positive-semidefinite, e.g. such as α-exponential kernels $k(x, z) = \exp(-\frac{||x-z||^\alpha}{\tau})$ with bandwidth parameter τ such the

- Gaussian kernel with $\alpha = 2$
- Laplacian kernel with $\alpha = 1$

What are the features corresponding to a given bivariate valid kernel function?

Mercer's Theorem: For kernel $k : X \times X \to \mathbb{R}$ on a compact domain $X \in \mathbb{R}^d$, we can find a sequence $\{\mu_j\}_{j=1}^{\infty}$ and a basis $\{\phi_j\}_{j=1}^{\infty}$ of $L_2(X)$ (a sequence of functions) such that $k(x, y) = \sum_{j=1}^{\infty} \mu_j \phi_j(x)\phi_j(y) = \langle \tilde{\phi}(x), \tilde{\phi}(y)\rangle_{l_2}$ for all $x, y \in X$. The features $\tilde{\phi}(x)$ are induced by a kernel with $\tilde{\phi}_j(x) = \sqrt{\mu_j}\phi_j(x)$.

1.3.3 Function Space Induced by Kernels

More generally, a kernel $k : X \times X \to \mathbb{R}$ induces a reproducing kernel Hilbert space (RKHS):

$$F_k = \{f : f(x) = \sum_{j=1}^{m} \beta_j k(x_j, x) \text{ for } x_j \in X, m \in \mathbb{N}\}$$

We can define an inner product for this function space: For any two functions $f(x) = \sum_{l=1}^{m} \alpha_l k(x_l, x)$ and $g(x) = \sum_{r=1}^{q} \beta_r k(x_r, x)$, the *inner product* reads:

$$\langle f, g\rangle_F = \sum_r \sum_l \alpha_l \beta_r k(x_r, x_l)$$

Hence, the *Hilbert norm* of $f(x) = \sum_{l=1}^{m} \alpha_l k(x_l, x)$ reads:

$$||f||_F^2 = \langle f, f\rangle_F = \sum_{l=1}^{m} \alpha_i \alpha_j k(x_i, x_j) = \alpha^T K \alpha$$

1.4 Other Non-Linear Methods

1.4.1 K-Nearest Neighbor (kNN)

With the method for **k-Nearest Neighbors** there is actually no training involved! The classification is done during test time:

1. Given training set D
2. Pick k and distance metric d in X
3. For given x, find among $x_1, ..., x_n \in D$ the k closest to $x \to x_{i_1}, ..., x_{i_k}$
4. Output the majority vote of labels $y_{i_1}, ..., y_{i_k}$

1.4.2 Decision Trees & Random Forests (RF)

Decision Trees are another method of classification. They:

- Return a partition of X with sets aligned with canonical coordinates
- Each x in a set S of the partition is labeled $\hat{y}$ = majority class in set S
- Sets in partition are determined by *leaf nodes* of a binary tree that contains data points that satisfy decision rules along the path

The creation of binary trees $f : X \to Y$ is done as follows:

- each tree node v is associated with a splitting ruler $r_v : X \to \{0, 1\}$
- each leaf node returns $\hat{y}$ = majority class in set S

The decision rule at tree node v has the following characteristics:

- Decision rule at some node v is usually of the form $r_v(x) = \mathbb{I}\{x_i > t_i\}$
- For each node v, choose feature i and threshold t_i that minimizes the uncertainty u on sets $D(v_1) = \{x \in D(v) : r_v(x) = 1\}$ and $D(v_0) = \{x \in D(v) : r_v(x) = 0\}$

Example: