# 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 $\alpha_l$ 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 $m$th 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

$\hat{w} = \text{argmin}_w ||y - Xw||^2 = X^\top\hat{\alpha}$

search in subspace $span(X^\top)$ → $\hat{\alpha} = \text{argmin}_\alpha ||y - XX^\top\alpha||^2$

$\to \hat{\alpha} = (XX^\top)^\dagger y$

$\hat{f}(x) = \hat{w}^\top x = y^\top(XX^\top)^\dagger Xx$

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

Kernelized

$\hat{w} = \text{argmin}_w ||y - \Phi w||^2 = \Phi^\top\hat{\alpha}$

$\hat{\alpha} = \text{argmin}_\alpha ||y - \Phi\Phi^\top\alpha||^2 = \text{argmin}_\alpha ||y - K\alpha||^2$

$\to \hat{\alpha} = (\Phi\Phi^\top)^\dagger y = K^\dagger y$

$\hat{f}(x) = \hat{w}^\top\phi(x) = y^\top K^\dagger\Phi\,\phi(x)$

Replace $X$ by $\Phi$, $x$ 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

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