Lecture 4

Regularization

Encouraging sparsity: l_0 regularization

Sparsity of w: Number of non-zero coefficients in w.

the good, the bad and the ugly

Choose $\Psi(w) = ||w||_0$

$$G(w) = \sum_{i=1}^{n}(w^T x_i - y_i)^2 + \lambda||w||_0$$

Good: "Information-theoretically" good! (need less data to learn)

Suppose weights in w are in $\{-w, -w+1, \cdots, 0, \cdots, w\}$

1.How many such s-sparse vector are there in dimensions?

We have $2w$ elements, need to fill s non-zero elements.

Answer: $\binom{d}{s}(2w)^s$ possibilities. $\binom{d}{s} = C_d^s$

2.How much data to learn?

About $\log(|F|)$ samples to learn (using the theorem from last time, note that we're ignoring ε, δ here):

$$\to \log\left(\binom{d}{s}(2w)^s\right) = s\log(\frac{d}{s}) + s\log(2w) \quad \because (\binom{d}{s} \approx (\frac{d}{s})^s)$$

3.How many free parameters?

$\rightarrow$ choose s coordinates: need $\log d$ bits (every bit is like a parameter you need to learn) per coordinate -> $s \log d$ in total.

$\rightarrow$ choose the value for non-zero coordinates: fix s values $= s \log d$ in total.

In contrast, without s-sparsity need about $\approx d$ samples in d dimensions (In this case, $d = s$).

$\therefore$ If $s \ll d$, need less data to learn!

Bad: $||w||_0$ is non-convex ($||w||_p, p < 1$ is non-convex)

minimizing $G(w) = \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \lambda ||w||_0$ is NP-hard.

Ugly: $||w||_0$ is highly-discontinuous

GD has no hopes!

l_1 regularization as a proxy for l_0

Choose $\Psi(w) = ||w||_1$

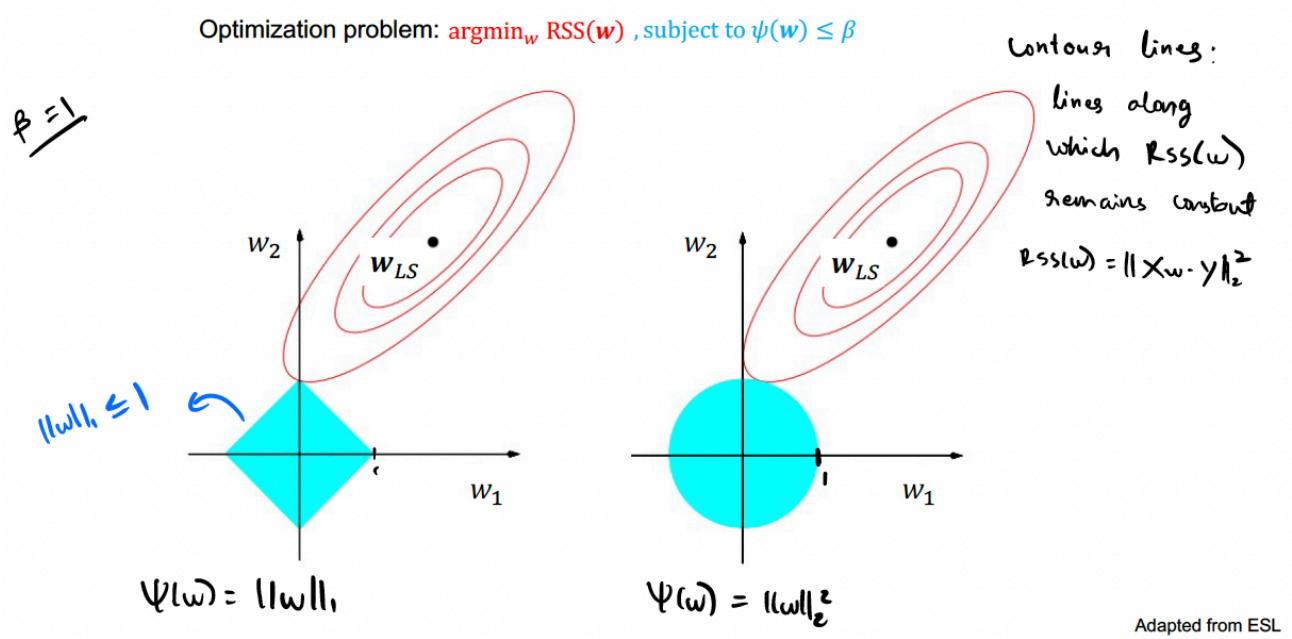
$$G(w) = \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \lambda ||w||_1$$

$||w||_1$ is convex. Can use GD / SGD to solve.

Minimizing $||w||_1$, often suffices to minimize $||w||_0$.

Theorem. Given n vectors $\{x_i \in \mathbb{R}^d, i \in [n]\}$ drawn i.i.d. from $N(0, I)$, let $y_i = w^{*T} x_i$ for some w^* with $||w||_0 = s$. Then for some fixed constant $C > 0$, the minimizer of $G(w)$ with $\psi(w) = ||w||_1$ will be w^* as long as $n > C \cdot s \log d$ (with high probability over the randomness in the training datapoints x_i).

Optimization problem: $\arg\min_w RSS(w)$, subject to $\psi(w) \leqslant \beta$.



Diving deeper: l_2 and l_1 regularization for the "isotropic" case

Isotropic assumption: $X^TX = I$

Isotropic informally means,

> all features have mean 0
>
> all features have variance 1
>
> features are uncorrelated

$\psi(w) = ||w||_2^2$

$$G(w) = \sum_{i=1}^{n}(x_i^Tw - y_i)^2 + \lambda||w||_2^2$$

$$w^* = (X^TX + \lambda I)^{-1}X^Ty$$

Now, $X^TX = I \Rightarrow w^* = \frac{1}{1+\lambda}X^Ty$

$$w_j^*(\cdot) = \frac{1}{1+\lambda}X_{(j)}^Ty$$

w_j^* menas jth coordinate of w^* , $X_{(j)}$ means jth row of X , $X_{(j)}^Ty$ menas correlation of jth feature with label.

l_2 regularization "shrinks" the estimated parameters.

Note: when features have unequal variance, l_2 regularization applies similar shrinkage to all of them.

$\therefore$ scaling features can be important.

$\psi(w) = ||w||_1$

$$G(w) = \sum_{i=1}^{n}(x_i^T w - y_i)^2 + \lambda||w||_1$$

what is gradient of $|w|$?

$$\frac{\partial |w|}{\partial w} = \begin{cases} 1 & w > 0 \\ -1 & w < 0 \end{cases}$$

At $w = 0$, we have a sub-gradient, ignore for now.

For $w_j \neq 0 (jth\ coordinate\ of\ w)$.

$$\frac{\partial G(w)}{\partial w_j} = 2\sum_{i=1}^{n}(x_i^T w - y_i)x_{i,j} + \lambda sign(w_j)$$

$x_{i,j}$ is jth coordinate of x_i

$$\begin{aligned} \frac{\partial G(w)}{\partial w_j} &= 2\sum_{i=1}^{n}(x_{i,j}x_i^T w) - 2\sum_{i=1}^{n} x_{i,j}y_i + \lambda sign(w_j) \\ &= 2w_j - 2X_{(i)}^T y + \lambda sign(w_j) \end{aligned}$$

$\therefore$ GD steps: $w_j \leftarrow w_j - \eta\big(2(w_j - X_{(i)}^T y) + \lambda sign(w_j)\big)$

Let's understand the gradient.

First, without l_1 regularization,

$$w_j \leftarrow w_j - \eta 2(w_j - X_{(i)}^T y)$$

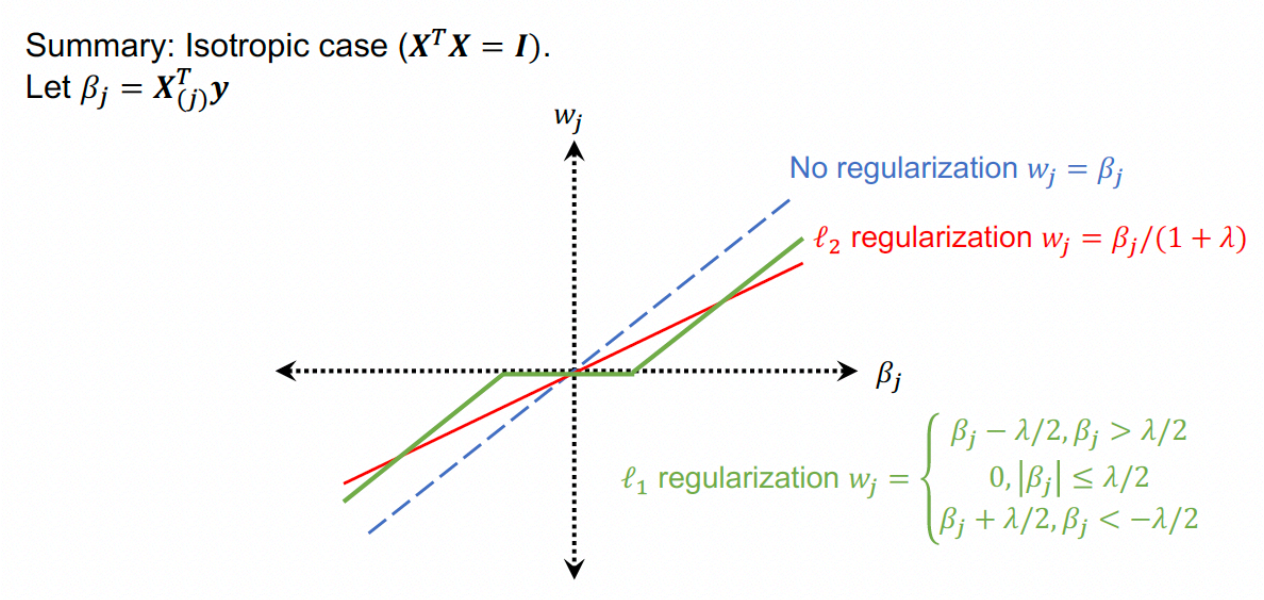
l_1 regularization is forcing you to zero!

with l_1 regularization: GD always has a shift of $-\eta\lambda sign(w_j)$, which pushes towards 0 .

Let $\beta_j = X_{(i)}^T y$

Using sub-gradients, we can show that for the l_1 regularized case:

$$w_j = \begin{cases} \beta_j - \frac{\lambda}{2},\ \beta_j > \frac{\lambda}{2} \\ 0,\ |\beta_j| \leqslant \frac{\lambda}{2} \\ \beta_j + \frac{\lambda}{2},\ \beta_j < -\frac{\lambda}{2} \end{cases}$$



Bias-variance tradeoff

The phenomenon of underfitting and overfitting is often referred to as the bias-variance tradeoff in the literature.

A model whose complexity is too small for the task will underfit. This is a model with a large bias because the model's accuracy will not improve even if we add a lot of training data.

A model whose complexity is too large for the amount of available training data will overfit. This is a model with high variance, because the model's predictions will vary a lot with the randomness in the training data (it can even fit any noise in the training data).

Kernels

Let's continue with regularized least squares with non-linear basis:

$$\begin{aligned} w^* &= \arg\min_w F(w) \\ &= \arg\min_w (||\Phi w - y||_2^2 + \lambda ||w||_2^2) \\ &= (\Phi^T \Phi + \lambda I)^{-1} \Phi^T y \end{aligned}$$

$\in \mathbb{R}^{n \times m}$. This operates in space $\mathbb{R}^m$ and m could be huge (and even infinite).

$$\Phi_{\in \mathbb{R}^{n \times m}} = \begin{pmatrix} \phi(x_1)^T \\ \phi(x_2)^T \\ \vdots \\ \phi(x_n)^T \end{pmatrix} \quad y_{\in \mathbb{R}^n} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

By setting the gradient of $F(w) = 0$:

$$\Phi^T(\Phi w^* - y) + \lambda w^* = 0$$

we know:

$$w^* = \frac{1}{\lambda}\Phi^T(y - \Phi w^*) = \Phi^T\alpha = \sum_{i=1}^{n} \alpha_i \phi(x_i)$$

Thus the least square solution is a linear combination of features of the data points!

This calculation does not show what α should be, but ignore that for now.

Why is this helpful?

Assuming we know α , the prediction of w^* on a new example x is

$$w^{*T}\phi(x) = \sum_{i=1}^{n} \alpha_i^T \phi(x_i)\phi(x)$$

Therefore, only inner products in the new feature space matter!

Kernel methods are exactly about computing inner products without explicitly computing ϕ .

Solving for α

$\alpha = \frac{1}{\lambda}(y - \Phi w^*)$

Solving for α, Step 1: Kernel matrix

Plugging in $w = \Phi^T\alpha$ into $F(w)$ gives

$$\begin{aligned} H(\alpha) &= F(\Phi^T\alpha) \\ &= ||\Phi\Phi^T\alpha - y||_2^2 + \lambda||\Phi^T\alpha||_2^2 \\ &= ||K\alpha - y||_2^2 + \lambda\alpha^T K\alpha K \end{aligned}$$

$K = \Phi\Phi^T \in R^{n\times n}$ is called Gram matrix or kernel matrix where the $(i,j)_{th}$ entry is:

$$K_{(i,j)} = \phi(x_i)^T\phi(x_j)$$

$\Phi\Phi^T$: $n * n$ dimensions, entry$(i,j) = \phi(x_i)^T\phi(x_j)$

$\Phi^T\Phi$: $m * m$ dimensions, entry$(i,j) = \sum_{k=1}^{n} \phi(x_k)_i\phi(x_k)_j$

both are symmetric & positive semi definite (psd)

*psd: Any matrix $A = UU^T$ is psd:

$$x^TAx = x^TUU^Tx = ||U^Tx||_2^2 \geqslant 0$$

Solving for α, Step 2: Minimize the dual

Minimize (the so-called dual formulation)

$$H(\alpha) = ||K\alpha - y||_2^2 + \lambda\alpha^T K\alpha$$

Setting the derivative to 0 we have

$$0 = (K^2 + \lambda K)\alpha - Ky = K((K + \lambda I)\alpha - y)$$

Thus $\alpha = (K + \lambda I)^{-1}y$ **is a minimizer** and we obtain

$$w^* = \Phi^T\alpha = \Phi^T(K + \lambda I)^{-1}y$$

The kernel trick

Minimizing $F(w)$ gives $w^* = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T y$ $(\Phi^T\Phi \; is \; covariance)$.

Minimizing $H(\alpha)$ gives $w^* = \Phi^T(\Phi\Phi^T + \lambda I)^{-1}y$ $(\Phi\Phi^T \; is \; kernel)$.

Note I has different dimensions in these two formulas.

Natural question: are the two solutions the same or different?

They have to be the same because $F(w)$ has a unique minimizer!

And they are:

$$\begin{aligned}
&(\Phi^T\Phi + \lambda I)^{-1}\Phi^T y \\
&\quad = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T(\Phi\Phi^T + \lambda I)(\Phi\Phi^T + \lambda I)^{-1}y \\
&\quad = (\Phi^T\Phi + \lambda I)^{-1}(\Phi^T\Phi\Phi^T + \lambda\Phi^T)(\Phi\Phi^T + \lambda I)^{-1}y \\
&\quad = (\Phi^T\Phi + \lambda I)^{-1}(\Phi^T\Phi + \lambda I)\Phi^T(\Phi\Phi^T + \lambda I)^{-1}y \\
&\quad = \Phi^T(\Phi\Phi^T + \lambda I)^{-1}y
\end{aligned}$$

If the solutions are the same, then what is the difference?

First, computing $(\Phi\Phi^T + \lambda I)^{-1}$ can be more efficient than computing $(\Phi^T\Phi + \lambda I)^{-1}$ when $n < m$ ($O(n^3)$ vs $O(m^3)$).

More importantly, computing $\alpha = (K + \lambda I)^{-1}y$ also only requires computing inner products in the new feature space Φ !

Now we can conclude that the exact form of $\phi(\cdot)$ is not essential; all we need to do is know the inner products $\phi(x)^T\phi(x')$.

For some "it is indeed possible to compute $\phi(x)^T\phi(x')$ without computing / knowing ϕ". This is the kernel trick.

The kernel trick: Example 1

Consider the following polynomial basis $\phi : \mathbb{R}^2 \to \mathbb{R}^3$:

$$\phi(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

What is the inner product between $\phi(\boldsymbol{x})$ and $\phi(\boldsymbol{x}')$?

$$\begin{aligned} \phi(\boldsymbol{x})^{\mathrm{T}}\phi(\boldsymbol{x}') &= {x_1}^2{x'_1}^2 + 2x_1x_2x'_1x'_2 + {x_2}^2{x'_2}^2 \\ &= (x_1x'_1 + x_2x'_2)^2 = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}')^2 \end{aligned}$$

Therefore, *the inner product in the new space is simply a function of the inner product in the original space*.

The kernel trick: Example 2

$\phi : \mathbb{R}^d \to \mathbb{R}^{2d}$ is parameterized by θ:

$$\phi_\theta(\boldsymbol{x}) = \begin{pmatrix} \cos(\theta x_1) \\ \sin(\theta x_1) \\ \vdots \\ \cos(\theta x_m) \\ \sin(\theta x_m) \end{pmatrix}$$

What is the inner product between $\phi_\theta(\boldsymbol{x})$ and $\phi_\theta(\boldsymbol{x}')$?

$$\begin{aligned} \phi_\theta(\boldsymbol{x})^{\mathrm{T}}\phi_\theta(\boldsymbol{x}') &= \sum_{m=1}^{d} \cos(\theta x_m)\cos(\theta x'_m) + \sin(\theta x_m)\sin(\theta x'_m) \\ &= \sum_{m=1}^{d} \cos(\theta(x_m - x'_m)) \qquad \text{(trigonometric identity)} \end{aligned}$$

Once again, *the inner product in the new space is a simple function of the features in the original space.*

The kernel trick: Example 3

Based on ϕ_θ, define $\phi_L : \mathbb{R}^d \to \mathbb{R}^{2d(L+1)}$ for some integer L:

$$\phi_L(\boldsymbol{x}) = \begin{pmatrix} \phi_0(\boldsymbol{x}) \\ \phi_{\frac{2\pi}{L}}(\boldsymbol{x}) \\ \phi_{2\frac{2\pi}{L}}(\boldsymbol{x}) \\ \vdots \\ \phi_{L\frac{2\pi}{L}}(\boldsymbol{x}) \end{pmatrix}$$

θ varies from $(0, \frac{2\pi}{L}, \ldots, 2\pi)$

What is the inner product between $\phi_L(\boldsymbol{x})$ and $\phi_L(\boldsymbol{x}')$?

$$\begin{aligned} \phi_L(\boldsymbol{x})^{\mathrm{T}} \phi_L(\boldsymbol{x}') &= \sum_{\ell=0}^{L} \phi_{\frac{2\pi\ell}{L}}(\boldsymbol{x})^{\mathrm{T}} \phi_{\frac{2\pi\ell}{L}}(\boldsymbol{x}') \\ &= \sum_{\ell=0}^{L} \sum_{m=1}^{d} \cos\left(\frac{2\pi\ell}{L}(x_m - x'_m)\right) \end{aligned}$$

The kernel trick: Example 4

When $L \to \infty$, even if we cannot compute $\phi(x)$ (since it's a vector of *infinite dimension*), we can still compute inner product:

change order of summation & integral

$$\begin{aligned} \phi_\infty(\boldsymbol{x})^{\mathrm{T}} \phi_\infty(\boldsymbol{x}') &= \int_0^{2\pi} \sum_{m=1}^{d} \cos(\theta(x_m - x'_m))\, d\theta \\ &= \sum_{m=1}^{d} \frac{\sin(2\pi(x_m - x'_m))}{x_m - x'_m} \end{aligned}$$

Again, a simple function of the original features.

Note that when using this mapping in linear regression, we are *learning a weight $\boldsymbol{w}^*$ with infinite dimension!*

Kernel Functions

Definition: a function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is called a kernel function if there exists a function $\phi : \mathbb{R}^d \to \mathbb{R}^m$ so that for any $x, x' \in \mathbb{R}^d$,

$$k(x, x') = \phi(x)^T \phi(x')$$

Examples:

$$k(x, x') = (x^T x')^2$$

$$k(x, x') = (x^T x' - 1)^2$$

$$k(x, x') = \sum_{m=1}^{d} \frac{\sin(2\pi(x_m - x'_m))}{x_m - x'_m}$$

$$k(x, x') = \exp(-||x - x'||_2^2)$$

Choosing a nonlinear basis ϕ becomes equivalent to choosing a kernel function.

As long as computing the kernel function is more efficient, we should apply the kernel trick.

Gram/kernel matrix becomes:

$$K = \Phi\Phi^T = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \cdots & \ddots & \cdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{pmatrix}$$

Determin Kernel

In fact, k is a kernel if and only if K is positive semi-definite for any n and any $x_1, x_2, \cdots, x_n$ (**Mercer theorem**).

For any function $f : \mathbb{R}^d \to \mathbb{R}$, $k(x, x') = f(x)f(x')$ is a kernel.

What is ϕ ? $\phi : \mathbb{R}^d \to \mathbb{R}, \phi(x) = f(x)$.

If $k_1(\cdot, \cdot)$ and $k_2(\cdot, \cdot)$ are kernels, then the following are also kernels:

- conical combination: $\alpha k_1(\cdot, \cdot) + \alpha k_2(\cdot, \cdot),\ if\ \alpha, \beta \geqslant 0$
- product: $k_1(\cdot, \cdot) \cdot k_2(\cdot, \cdot)$
- exponential: $e^{k(\cdot, \cdot)}$

These are not kernels:

- $-k(\cdot, \cdot)$
- $\ln(k(\cdot, \cdot))$
- $k_1 - k_2$

How to determine a function is a kernel?

First, calculate $K = \Phi\Phi^T = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \cdots & \ddots & \cdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{pmatrix}$. K must be positive semi-definite.

If not, then it isn't a kernel.

Function

$$k(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2$$

is *not a kernel*, why?

If it is a kernel, the kernel matrix for two data points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$: this entry is $\|x-x\|_2^2 = 0$

$$\boldsymbol{K} = \begin{pmatrix} 0 & \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2 \\ \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2 & 0 \end{pmatrix}$$

must be positive semidefinite, *but is it?*

$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ is not psd. why?

$$\begin{pmatrix} 1 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{pmatrix} -1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = -2$$

How to prove psd? Math in ML: p118

- All A's leading principal minor determinant should $\geqslant 0$.
- All A eigenvalue $\geqslant 0$
- There exists a invertible P , $A = P^T P$

Popular kernels

1.Polynomial kernel:

$$k(x, x') = (x^T x' + c)^M$$

What is the corresponding ϕ ?

$c = 0, M = 2$, we saw earlier: $\phi(x) = \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{pmatrix}$.

The case of larges m can be obtained by applying this seperately.

2.Gaussian kernel or Radial basis function (RBF) kernel:

$$k(x, x') = \exp\left(-\frac{||x - x'||_2^2}{2\sigma^2}\right)$$

for some $\sigma > 0$.

What is the corresponding ϕ ?

$$k(x, x') = \exp\left(-\frac{||x||_2^2}{2\sigma^2}\right) \exp\left(-\frac{||x'||_2^2}{2\sigma^2}\right) \exp\left(-\frac{x^T x'}{2\sigma^2}\right)$$

$$k(x, x') = f(x) f(x'), \ where \ f(x) = \exp\left(-\frac{||x||_2^2}{2\sigma^2}\right)$$

transformation for the product.

$$\exp(\frac{x^T x'}{\sigma^2}) = 1 + \frac{x^T x'}{\sigma^2} + \frac{1}{2!}\frac{(x^T x')^2}{(\sigma^2)^2} + \frac{1}{3!}\frac{(x^T x')^3}{(\sigma^2)^3} + \cdots$$

each of these is a polynomial kernel.

∞ dimensional feature space.

Prediction with kernels

As long as $w^* = \sum_{i=1}^n \alpha_i \phi(x_i)$, prediction on a new example x becomes

$$w^{*T}\phi(x) = \sum_{i=1}^n \alpha_i^T \phi(x_i)\phi(x) = \sum_{i=1}^n \alpha_i k(x_i, x)$$

This is known as a **non-parametric method**. Informally speaking, this means that there is no fixed set of parameters that the model is trying to learn (remember w^* could be infinite). Nearest-neighbors is another non-parametric method we have seen.

LR, logistic, bayesian, NN and perceptron are parametric methods.

SVM, knn, decision tree, and algorithm with kernels are non-parametric methods.