Foundations of Data Science, Fall 2020

12. Support Vector Machines II

Prof. Dan Olteanu

Nov 10, 2020

https://lms.uzh.ch/url/RepositoryEntry/16830890400

https://uzh.zoom.us/j/96690150974?pwd=cnZmMTduWUtCeWoxYW85Z3RMYnpTZz09

Support Vector Machines: Story So Far

- Primal Formulation of SVM: Minimise the hinge loss with regularisation
- Slack variables ζ_i for linearly (non-)separable data

minimise: $\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N}\zeta_i$

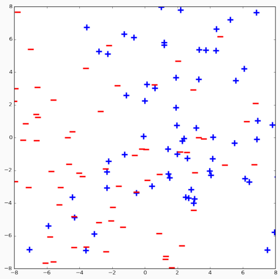
subject to: $y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) \geq 1 - \zeta_i$

$\zeta_i \geq 0$

for $1 \leq i \leq N$

Here $y_i \in \{-1, 1\}$

For the optimal solution:
$\zeta_i = \max\{0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0)\}$



Optimisation Background: Constrained Optimisation with Inequalities

Primal Form

$$\begin{aligned} &\text{minimise} && F(\mathbf{z}) && \\ &\text{subject to} && g_i(\mathbf{z}) \geq 0 && i = 1, \ldots, m \\ & && h_j(\mathbf{z}) = 0 && j = 1, \ldots, \ell \end{aligned}$$

Lagrange Function

$$\Lambda(\mathbf{z}; \boldsymbol{\alpha}, \boldsymbol{\mu}) = F(\mathbf{z}) - \sum_{i=1}^{m}\alpha_i g_i(\mathbf{z}) - \sum_{j=1}^{\ell}\mu_j h_j(\mathbf{z})$$

Primal variables: $\mathbf{z}$. Dual variables: $\boldsymbol{\alpha}, \boldsymbol{\mu}$.

Karush-Kuhn-Tucker (KKT) Conditions

Lagrange Function

$$\Lambda(\mathbf{z}; \boldsymbol{\alpha}, \boldsymbol{\mu}) = F(\mathbf{z}) - \sum_{i=1}^{m}\alpha_i g_i(\mathbf{z}) - \sum_{j=1}^{\ell}\mu_j h_j(\mathbf{z})$$

Karush-Kuhn-Tucker (KKT) conditions

$$\begin{aligned} &\text{Dual feasibility:} && \alpha_i \geq 0 && \text{for } i = 1, \ldots m \\ &\text{Primal feasibility:} && g_i(\mathbf{z}) \geq 0 && \text{for } i = 1, \ldots m \\ & && h_j(\mathbf{z}) = 0 && \text{for } j = 1, \ldots \ell \\ &\text{Complementary slackness:} && \alpha_i g_i(\mathbf{z}) = 0 && \text{for } i = 1, \ldots m \end{aligned}$$

- For convex problems: KKT conditions are **necessary and sufficient** for a critical point of Λ to be the minimum of the original constrained optimisation
- For non-convex problems: KKT are **necessary but not sufficient**

From Primal to Dual SVM Formulation: The Linearly Separable Case

minimise: $\frac{1}{2}\|\mathbf{w}\|_2^2$ subject to: $y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) \geq 1$ for $1 \leq i \leq N$

The Lagrange function and its gradients wrt the primal variables:

$$\Lambda(\mathbf{w}, w_0; \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|_2^2 - \sum_{i=1}^{N}\underbrace{\alpha_i}_{\text{dual variables}}\underbrace{(y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) - 1)}_{\text{constraints}}$$

$$\frac{\partial\Lambda}{\partial w_0} = -\sum_{i=1}^{N}\alpha_i y_i \qquad \nabla_{\mathbf{w}}\Lambda = \mathbf{w} - \sum_{i=1}^{N}\alpha_i y_i\mathbf{x}_i$$

At optimality, these gradients are 0:

$$\sum_{i=1}^{N}\alpha_i y_i = 0 \qquad \mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i\mathbf{x}_i$$

Computing w_0 using the Optimal Values for Variables w and $\boldsymbol{\alpha}$

Optimal $\mathbf{w}$ obtained as $\mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i\mathbf{x}_i$

At optimality: $y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) = 1$, for all $1 \leq i \leq N$

We can obtain w_0 using the above equality constraint for any i:

$$y_i^2\left(\left(\sum_{j=1}^{N}\alpha_j y_j\mathbf{x}_j\right)\cdot\mathbf{x}_i + w_0\right) = y_i \;\Rightarrow\; w_0 = y_i - \sum_{j=1}^{N}\alpha_j y_j(\mathbf{x}_j\cdot\mathbf{x}_i)$$

It is numerically more stable to have w_0 the average over all possible values:

$$w_0 = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \sum_{j=1}^{N}\alpha_j y_j(\mathbf{x}_j\cdot\mathbf{x}_i)\right)$$

We expressed optimal $\mathbf{w}$ and w_0 as functions of $\boldsymbol{\alpha}$. How can we compute $\boldsymbol{\alpha}$?

Reduced Expression of the Lagrange Function

Plug the optimal $\mathbf{w}$ and constraint $\sum_{i=1}^N \alpha_i y_i = 0$ into Lagrangian:

$$g(\alpha) = \frac{1}{2}(\underbrace{\mathbf{w}}_{\sum_{i=1}^N \alpha_i y_i \mathbf{x}_i})^{\mathsf{T}} \underbrace{\mathbf{w}}_{\sum_{j=1}^N \alpha_j y_j \mathbf{x}_j} - \sum_{i=1}^N \alpha_i y_i \underbrace{\mathbf{w}}_{\sum_{j=1}^N \alpha_j y_j \mathbf{x}_j} \cdot \mathbf{x}_i - \underbrace{\sum_{i=1}^N \alpha_i y_i\, w_0}_{0} + \sum_{i=1}^N \alpha_i$$

$$= \frac{1}{2}\sum_{i=1}^N \sum_{j=1}^N \alpha_i y_i \alpha_j y_j (\mathbf{x}_i \cdot \mathbf{x}_j) - \sum_{i=1}^N \sum_{j=1}^N \alpha_i y_i \alpha_j y_j (\mathbf{x}_i \cdot \mathbf{x}_j) + \sum_{i=1}^N \alpha_i$$

$$= \sum_{i=1}^N \alpha_i - \frac{1}{2}\sum_{i=1}^N \sum_{j=1}^N \alpha_i y_i \alpha_j y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

To find critical points of Λ satisfying the KKT conditions,

it is sufficient to find the critical points of g that satisfy the constraints:

$\alpha_i \geq 0$ and $\sum_{i=1}^N \alpha_i y_i = 0$

Primal and Dual SVM Formulations in the Linearly Separable Case

Primal Form

minimise: $\frac{1}{2}\|\mathbf{w}\|_2^2$

subject to:

$y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1$

for $1 \leq i \leq N$

- Quadratic convex problem
- $D + 1$ variables
- Constraints define a complex polytope

Dual Form

maximise $\sum_{i=1}^N \alpha_i - \frac{1}{2}\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$

subject to:

$\sum_{i=1}^N \alpha_i y_i = 0$ and $\alpha_i \geq 0$

for $1 \leq i \leq N$

- Quadratic concave problem
- N variables
- Very simple box constraints + one zero-sum constraint

Why Prefer the SVM Dual Formulation?

Reason 1: $D \approx N$ or $D \gg N$: Basis expansion to capture non-linear discriminating boundaries

Reason 2: Natural kernelisation: Replace dot product $\mathbf{x}_i \cdot \mathbf{x}_j$ by kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j)$

Reason 3: The number of non-zero variables α_i can be much smaller in practice!

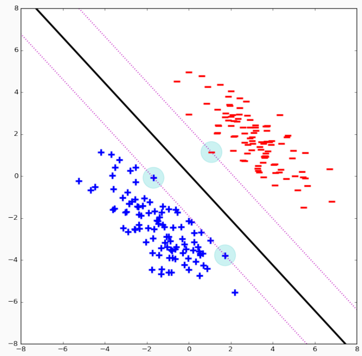
Complementary slackness conditions: $\alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) - 1\right) = 0$

- $\alpha_i = 0$ OR $y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) = 1$
- Only the points $\mathbf{x}_i$ with $\alpha_i > 0$ contribute to the solution $\mathbf{w}$:

$$\mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i$$

- Such vectors $\mathbf{x}_i$ form the support of the solution $\Rightarrow$ support vectors
- Such vectors x_i are points with *least* margin

Support Vectors



SVM Dual Formulation in the Non-Linearly Separable Case

SVM primal formulation

minimise:	$\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^N \zeta_i$	
subject to:	$y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1 - \zeta_i$	for $1 \leq i \leq N$
	$\zeta_i \geq 0$	for $1 \leq i \leq N$

Lagrange Function

$$\Lambda(\mathbf{w}, w_0, \zeta; \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^N \zeta_i - \sum_{i=1}^N \alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) - (1 - \zeta_i)\right) - \sum_{i=1}^N \mu_i \zeta_i$$

Gradient of Lagrange Function

Lagrange Function

$$\Lambda(\mathbf{w}, w_0, \zeta; \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^N \zeta_i - \sum_{i=1}^N \alpha_i (y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) - (1 - \zeta_i)) - \sum_{i=1}^N \mu_i \zeta_i$$

We write derivatives with respect to $\mathbf{w}$, w_0 and ζ_i:

$$\frac{\partial \Lambda}{\partial w_0} = -\sum_{i=1}^N \alpha_i y_i$$

$$\frac{\partial \Lambda}{\partial \zeta_i} = C - \alpha_i - \mu_i$$

$$\nabla_{\mathbf{w}} \Lambda = \mathbf{w} - \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i$$

For (KKT) dual feasibility constraints, we require $\alpha_i \geq 0$, $\mu_i \geq 0$

SVM Dual Formulation in the Non-Linearly Separable Case

Setting the derivatives to 0, substituting the resulting expressions in Λ (and simplifying), we get a function $g(\alpha)$ and some constraints

$$g(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

Constraints

$$0 \leq \alpha_i \leq C \qquad 1 \leq i \leq N$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

Finding critical points of Λ satisfying the KKT conditions corresponds to finding the maximum of $g(\alpha)$ subject to the above constraints

Primal and Dual SVM Formulations in the Non-Linearly Separable Case

Primal Form

minimise: $\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N}\zeta_i$

subject to:

$y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) \geq 1 - \zeta_i$

$\zeta_i \geq 0$

for $1 \leq i \leq N$

- Quadratic convex problem
- $D + N + 1$ variables
- Constraints define a complex polytope

Dual Form

maximise $\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i\cdot\mathbf{x}_j)$

subject to:

$\sum_{i=1}^{N}\alpha_i y_i = 0$

$0 \leq \alpha_i \leq C$

for $1 \leq i \leq N$

- Quadratic concave problem
- N variables
- Very simple box constraints + one zero-sum constraint

Making Predictions using SVM Dual

Recall the optimal solution for $\mathbf{w}$ expressed using the dual variables α:

$$\mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i$$

The bias w_0 is also expressible using α.

Prediction on a new point $\mathbf{x}_{new}$ requires inner products with the support vectors:

$$\mathbf{w}\cdot\mathbf{x}_{new} = \sum_{i=1}^{N}\alpha_i y_i(\mathbf{x}_i\cdot\mathbf{x}_{new})$$

We can as well using blackbox access to a function $\kappa(\cdot,\cdot)$ that maps two inputs $\mathbf{x}, \mathbf{x}'$ to their inner product $\mathbf{x}\cdot\mathbf{x}'$. This is a kernel function.

Mercer Kernels

A function κ is a kernel that computes the dot product for some expansion ϕ

$$\kappa(\mathbf{x}',\mathbf{x}) : \mathcal{X}\times\mathcal{X}\to\mathbb{R}, \qquad \kappa(\mathbf{x}',\mathbf{x}) = \phi(\mathbf{x}')\cdot\phi(\mathbf{x})$$

$\kappa(\mathbf{x},\mathbf{x}')$ is some measure of similarity between $\mathbf{x}$ and $\mathbf{x}'$

Gram matrix

$$\mathbf{K} = \begin{bmatrix} \kappa(\mathbf{x}_1,\mathbf{x}_1) & \kappa(\mathbf{x}_1,\mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_1,\mathbf{x}_N) \\ \kappa(\mathbf{x}_2,\mathbf{x}_1) & \kappa(\mathbf{x}_2,\mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_2,\mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_N,\mathbf{x}_1) & \kappa(\mathbf{x}_N,\mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_N,\mathbf{x}_N) \end{bmatrix}$$

Mercer or positive definite kernel: The Gram matrix is always positive semi-definite

SVM Dual Formulation using Mercer Kernel

maximise $\sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{K}_{i,j}$ s.t.: $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{N}\alpha_i y_i = 0$

where the Gram matrix $\mathbf{K}$ is

$$\mathbf{K} = \begin{bmatrix} \kappa(\mathbf{x}_1,\mathbf{x}_1) & \kappa(\mathbf{x}_1,\mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_1,\mathbf{x}_N) \\ \kappa(\mathbf{x}_2,\mathbf{x}_1) & \kappa(\mathbf{x}_2,\mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_2,\mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_N,\mathbf{x}_1) & \kappa(\mathbf{x}_N,\mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_N,\mathbf{x}_N) \end{bmatrix}$$

To make prediction on new $\mathbf{x}_{new}$, we compute $\kappa(\mathbf{x}_i,\mathbf{x}_{new})$ for support vectors $\mathbf{x}_i$

Which Function Constitutes a Kernel?

Kernel engineering: We can build kernels using simpler kernels as building blocks

Given kernels κ_1, κ_2, the following are kernels:

1. $\kappa(\mathbf{x},\mathbf{x}') = c\kappa_1(\mathbf{x},\mathbf{x}')$ — $c > 0$
2. $\kappa(\mathbf{x},\mathbf{x}') = f(\mathbf{x})\kappa_1(\mathbf{x},\mathbf{x}')f(\mathbf{x}')$ — f is a function
3. $\kappa(\mathbf{x},\mathbf{x}') = q(\kappa_1(\mathbf{x},\mathbf{x}'))$ — q is a polynomial with non-negative coefficients
4. $\kappa(\mathbf{x},\mathbf{x}') = \exp(\kappa_1(\mathbf{x},\mathbf{x}'))$
5. $\kappa(\mathbf{x},\mathbf{x}') = \kappa_1(\mathbf{x},\mathbf{x}')\kappa_2(\mathbf{x},\mathbf{x}')$
6. $\kappa(\mathbf{x},\mathbf{x}') = \kappa_a(\mathbf{x}_a,\mathbf{x}_a')\kappa_b(\mathbf{x}_b,\mathbf{x}_b')$ — $\mathbf{x} = (\mathbf{x}_a,\mathbf{x}_b)$, $\mathbf{x}' = (\mathbf{x}_a',\mathbf{x}_b')$, κ_a, κ_b are kernels
7. $\kappa(\mathbf{x},\mathbf{x}') = \kappa_a(\mathbf{x}_a,\mathbf{x}_a') + \kappa_b(\mathbf{x}_b,\mathbf{x}_b')$
8. $\kappa(\mathbf{x},\mathbf{x}') = \mathbf{x}^{\mathsf{T}}\mathbf{A}\mathbf{x}'$ — $\mathbf{A}$ is a symmetric positive semi-definite matrix

Example: Kernel Engineering

Recall the Gaussian/RBF kernel: $\kappa_{\text{RBF}}(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/2\sigma^2)$

We can show that κ_{RBF} is a kernel as follows:

$$\|\mathbf{x} - \mathbf{x}'\|^2 = \mathbf{x}^\top \mathbf{x} - 2\mathbf{x}^\top \mathbf{x}' + (\mathbf{x}')^\top \mathbf{x}'$$

$$\kappa_{\text{RBF}}(\mathbf{x}, \mathbf{x}') = \underbrace{\exp(-\mathbf{x}^\top \mathbf{x}/2\sigma^2)}_{f(\mathbf{x})} \underbrace{\exp(\mathbf{x}^\top \mathbf{x}'/\sigma^2)}_{\exp(c\kappa_1(\mathbf{x},\mathbf{x}'))} \underbrace{\exp(-(\mathbf{x}')^\top \mathbf{x}'/2\sigma^2)}_{f(\mathbf{x}')}$$

We can replace $\kappa_1(\mathbf{x}, \mathbf{x}')$ with a non-linear kernel, so RBF is not restricted to Euclidean distance:

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2\sigma^2}(-2\kappa_1(\mathbf{x}, \mathbf{x}') + \kappa_1(\mathbf{x}, \mathbf{x}) + \kappa_1(\mathbf{x}', \mathbf{x}'))\right)$$

Recall: Expansion Function for RBF Kernel

What is ϕ_{RBF} such that $\kappa_{\text{RBF}}(\mathbf{x}', \mathbf{x}) = \exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2) = \phi_{\text{RBF}}(\mathbf{x}) \cdot \phi_{\text{RBF}}(\mathbf{x}')$?

We use the following:

$$\|\mathbf{x} - \mathbf{x}'\|^2 = \langle \mathbf{x} - \mathbf{x}', \mathbf{x} - \mathbf{x}' \rangle = \langle \mathbf{x}, \mathbf{x} \rangle + \langle \mathbf{x}', \mathbf{x}' \rangle - 2\langle \mathbf{x}, \mathbf{x}' \rangle = \|\mathbf{x}\|^2 + \|\mathbf{x}'\|^2 - 2\mathbf{x} \cdot \mathbf{x}'$$

$$\exp(\mathbf{x} \cdot \mathbf{x}') = \sum_{k=0}^{\infty} \frac{1}{k!} \underbrace{(\mathbf{x} \cdot \mathbf{x}')^k}_{\phi_{poly}(\mathbf{x}) \cdot \phi_{poly}(\mathbf{x}')} \qquad \texttt{Taylor expansion:} \quad \exp(z) = \sum_{k=0}^{\infty} \frac{z^k}{k!}$$

Without loss of generality, assume $\gamma = 1$. Then,

$$\exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2) = \sum_{k=0}^{\infty} \underbrace{\left(\sqrt{\frac{1}{k!}} \exp(-\|\mathbf{x}\|^2)\phi_{poly}(\mathbf{x})\right)}_{\text{row } k \text{ in } \phi_{\text{RBF}}(\mathbf{x})} \cdot \underbrace{\left(\sqrt{\frac{1}{k!}} \exp(-\|\mathbf{x}'\|^2)\phi_{poly}(\mathbf{x}')\right)}_{\text{row } k \text{ in } \phi_{\text{RBF}}(\mathbf{x}')}$$

$\phi_{\text{RBF}} : \mathbb{R}^d \to \mathbb{R}^\infty$ projects vectors into an infinite dimensional space!

- Not feasible to compute $\kappa_{\text{RBF}}(\mathbf{x}', \mathbf{x})$ using ϕ_{RBF}!

Recall: Expansion Function for Polynomial Kernel

Find expansion function for kernel $\kappa_{\text{poly}}(\mathbf{x}', \mathbf{x}) = (\mathbf{x} \cdot \mathbf{x}')^d$, where $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^k$

We use $(z_1 + \cdots + z_k)^d = \sum_{n_i \geq 0, \sum_i n_i = d} \underbrace{\frac{d!}{n_1! \cdots n_k!}}_{multinomial\ coeff.\ C(d; n_1, \ldots, n_k)} z_1^{n_1} \cdots z_k^{n_k}$

C = # of ways to distribute d balls into k bins, where the j-th bin holds $n_j \geq 0$ balls

Now assume $z_i = x_i x_i'$ in the above formula. Then,

$$(\mathbf{x} \cdot \mathbf{x}')^d = \underbrace{\sum_{n_i \geq 0, \sum_i n_i = d}}_{\text{dim of } \phi_{\text{poly}}} \underbrace{\sqrt{C(d; n_1, \ldots, n_k)} x_1^{n_1} \cdots x_k^{n_k}}_{\text{one row in } \phi_{\text{poly}}(\mathbf{x})} \underbrace{\sqrt{C(d; n_1, \ldots, n_k)} (x_1')^{n_1} \cdots (x_k')^{n_k}}_{\text{one row in } \phi_{\text{poly}}(\mathbf{x}')}$$

The dimension of the vectors $\phi_{\text{poly}}(\mathbf{x})$ and $\phi_{\text{poly}}(\mathbf{x}')$ is $O(k^d)$.

Complexity of computing $\kappa(\mathbf{x}', \mathbf{x})$: $O(k^d)$ using ϕ_{poly} vs. $O(k \log d)$ using $(\mathbf{x} \cdot \mathbf{x}')^d$!

Examples: Expansion Function for Polynomial Kernel

For $\mathbf{x} = [x_1\ x_2]^\top$ and $\mathbf{x}' = [x_1'\ x_2']^\top$ find ϕ such that $\kappa(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}')$

$$\kappa(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}')^2 = (x_1 x_1' + x_2 x_2')^2 = x_1^2 (x_1')^2 + 2 x_1 x_1' x_2 x_2' + x_2^2 (x_2')^2 = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}')$$

$$\phi(\mathbf{x}) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2} x_1 x_2 \end{bmatrix} \qquad \phi(\mathbf{x}') = \begin{bmatrix} (x_1')^2 \\ (x_2')^2 \\ \sqrt{2} x_1' x_2' \end{bmatrix}$$

$$\begin{aligned} \kappa(\mathbf{x}, \mathbf{x}') &= (\mathbf{x} \cdot \mathbf{x}' + \theta)^2 \\ &= (x_1 x_1' + x_2 x_2' + \theta)^2 = x_1^2 (x_1')^2 + 2 x_1 x_1' x_2 x_2' + x_2^2 (x_2')^2 + 2 x_1 x_1' \theta + 2 x_2 x_2' \theta + \theta^2 \end{aligned}$$

$$\phi(\mathbf{x}) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2} x_1 x_2 \\ \sqrt{2\theta} x_1 \\ \sqrt{2\theta} x_2 \\ \theta \end{bmatrix} \qquad \phi(\mathbf{x}') = \begin{bmatrix} (x_1')^2 \\ (x_2')^2 \\ \sqrt{2} x_1' x_2' \\ \sqrt{2\theta} x_1' \\ \sqrt{2\theta} x_2' \\ \theta \end{bmatrix}$$

Kernels on Discrete Data: String Kernel

Let alphabet $\mathcal{A} = \{A, R, N, D, C, E, Q, G, H, I, L, K, M, F, P, S, T, W, Y, V\}$

Each letter denotes one of the 20 aminoacids

Let $\mathbf{x}$ and $\mathbf{x}'$ be strings over $\mathcal{A}$:

```
IPTSALVKETLALLSTHRTLLIANETLRIPVPVHKNHQLCTEEIFQGIGTLESQTVQGGTV
ERLFKNLSLIKKYIDGQKKKCGEERRRVNQFLDYLQEFLGVMNTEWI
```

```
PHRRDLCSRSIWLARKIRSDLTALTESYVKHQGLWSELTEAERLQENLQAYRTFHVLLA
RLLEDQQVHFTPTEGDFHQAIHTLLLQVAAFAYQIEELMILLEYKIPRNEADGMLFEKK
LWGLKVLQELSQWTVRSIHDLRFISSHQTGIP
```

These strings encode proteins. They have the string LQE in common.

The kernel defines similarities on strings: $\kappa(\mathbf{x}, \mathbf{x}') = \sum_s w_s \phi_s(\mathbf{x}) \phi_s(\mathbf{x}')$

$\phi_s(\mathbf{x})$ is the number of times s appears in $\mathbf{x}$ as substring

w_s is the weight associated with substring s