1. Prove that the solution lies in the span of the training points (i.e. \( w = \sum_{i=1}^{n} \alpha_i x_i \) for some \( \alpha_i \))

2. Rewrite the algorithm and the classifier so that all training or testing inputs \( x_i \) are only accessed in inner-products with other inputs, e.g. \( x_i^\top x_j \)

3. Define a kernel function and substitute \( k(x_i, x_j) \) for \( x_i^\top x_j \)
Any positive semi-definite matrix is a well-defined kernel.

**Definition: Positive Semi-Definite Matrices**

A kernel matrix is positive-semidefinite is equivalent to any of the following statements:

1. All eigenvalues of $K$ are non-negative.
2. There exists a real matrix $P$ such that $K = P^\top P$.
3. For all real vectors $x$, $x^\top K x \geq 0$.

Common kernels:

- **Linear**: $k(x, z) = x^\top z$
- **RBF**: $k(x, z) = e^{-\frac{(x-z)^2}{\sigma^2}}$
- **Polynomial**: $k(x, z) = (1 + x^\top z)^d$
We can construct new kernels by recursively combining one or more rules from the following list:

1. \( k(x, z) = x^\top z \)
2. \( k(x, z) = ck_1(x, z) \)
3. \( k(x, z) = k_1(x, z) + k_2(x, z) \)
4. \( k(x, z) = g(k(x, z)) \)
5. \( k(x, z) = k_1(x, z)k_2(x, z) \)
6. \( k(x, z) = f(x)k_1(x, z)f(z) \)
7. \( k(x, z) = e^{k_1(x,z)} \)
8. \( k(x, z) = x^\top Az \)

where \( c \geq 0 \) and \( g() \) is a polynomial with positive coefficients.
Quiz Time!
Time to put your thinking caps on!

Quiz 1
Prove that the RBF kernel \( k(x, z) = e^{\frac{-(x-z)^2}{\sigma^2}} \) is a well-defined kernel matrix.

Quiz 2
Prove that the following kernel, defined on any two sets \( S_1, S_2 \subseteq \Omega \), is well-defined: \( k(S_1, S_2) = e^{|S_1 \cap S_2|} \).
Recap: Vanilla OLS regression minimizes the following squared loss regression loss function:

$$\min_w \sum_{i=1}^n (w^\top x_i - y_i)^2,$$

to find the hyper-plane $w$. The prediction at a test-point is simply $h(x) = w^\top x$.

If we let $X = [x_1, \ldots, x_n]$ and $y = [y_1, \ldots, y_n]^\top$, the solution of OLS can be written in closed form:

$$w = (XX^\top)^{-1}Xy$$
To kernelize the algorithm, we express the solution $\mathbf{w}$ as a linear combination of the training inputs:

$$
\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i = \mathbf{X} \vec{\alpha}.
$$

During testing, a test point is only accessed through inner-products with training inputs:

$$
h(\mathbf{z}) = \mathbf{w}^\top \mathbf{z} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i^\top \mathbf{z}.
$$

We can now immediately kernelize the algorithm by substituting $k(\mathbf{x}_i, \mathbf{z})$ for any inner-product $\mathbf{x}_i^\top \mathbf{z}$.
Kernelized Linear Regression

Derivation for Kernelized Ordinary Least Squares

Theorem

*Kernelized ordinary least squares has the solution \( \tilde{\alpha} = K^{-1}y \).*

Proof.

\[
X\tilde{\alpha} = w = (XX^T)^{-1}Xy \\
(X^TXX^T)X\tilde{\alpha} = (X^TXX^T)(XX^T)^{-1}Xy \\
(X^TX)(X^TX)\tilde{\alpha} = X^T(XX^T)(XX^T)^{-1}Xy \\
K^2\tilde{\alpha} = Ky \\
\tilde{\alpha} = K^{-1}y
\]
Kernel SVM
Kernelize your SVMs for more power and fun!

(Original) SVM Primal Form

\[
\min_{\xi_i \geq 0, w, b} \quad w^T w + C \sum_{i=1}^{n} \xi_i \\
\text{s.t. } \forall i, \quad y_i (w^T x_i + b) \geq 1 - \xi_i
\]

SVM Dual Form

\[
\min_{\alpha_1, \ldots, \alpha_n} \quad \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K_{ij} - \sum_{i=1}^{n} \alpha_i \\
\text{s.t. } 0 \leq \alpha_i \leq C \\
\sum_{i=1}^{n} \alpha_i y_i = 0
\]

Where \( w = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i) \) and the decision function is:

\[
h(x) = \text{sign} \left( \sum_{i=1}^{n} \alpha_i y_i k(x_i, x) + b \right).
\]
Question: What is the dual form of the hard-margin SVM?
Support vectors: only support vectors satisfy the constraint with equality: $y_i(w^T \phi(x_i) + b) = 1$. In the dual, these are the training inputs with $\alpha_i > 0$.

Recovering $b$: we can solve for $b$ from the support vectors using:

$$y_i(w^T \phi(x_i) + b) = 1$$

$$y_i \left( \sum_j y_j \alpha_j k(x_j, x_i) + b \right) = 1$$

$$\sum_j y_j \alpha_j k(x_j, x_i) + b = y_i$$

$$y_i - \sum_j y_j \alpha_j k(x_j, x_i) = b$$
Kernel SVM - The Smart Nearest Neighbor
Because who wants a dumb nearest neighbor?

KNN for binary classification problems

\[ h(z) = \text{sign} \left( \sum_{i=1}^{n} y_i \delta^{nn}(x_i, z) \right), \]

where \( \delta^{nn}(z, x_i) \in \{0, 1\} \) with \( \delta^{nn}(z, x_i) = 1 \) only if \( x_i \) is one of the \( k \) nearest neighbors of test point \( z \).

SVM decision function

\[ h(z) = \text{sign} \left( \sum_{i=1}^{n} y_i \alpha_i k(x_i, z) + b \right) \]

Kernel SVM is like a smart nearest neighbor: it considers all training points but kernel function assigns more weight to closer points. It also learns a weight \( \alpha_i > 0 \) for each training point and a bias \( b \), and sets many \( \alpha_i = 0 \) for useless training points.
Thank You!
I hope you had as much fun as I did!

That’s all folks! Have a great day and keep kernelizing!