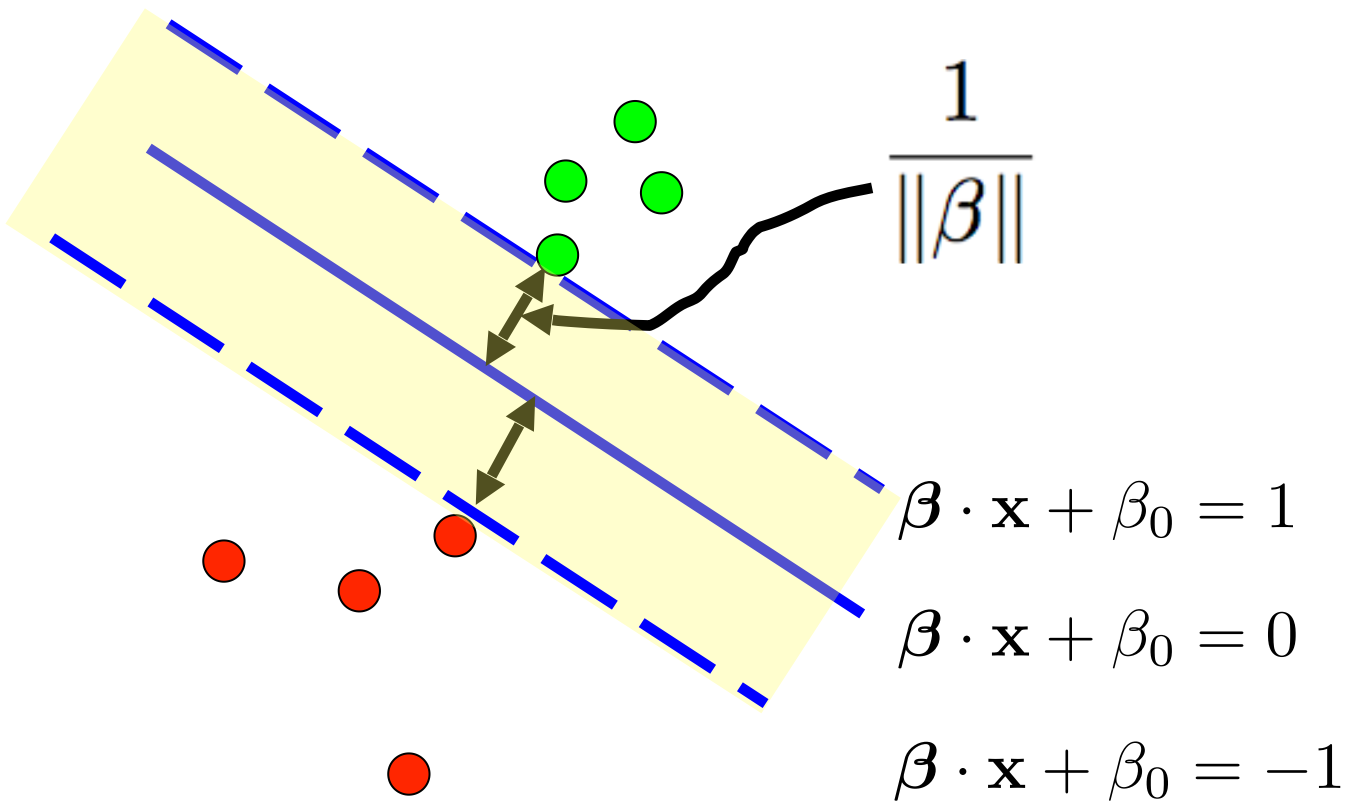
Hard Margin

Linear SVM for Separable Data

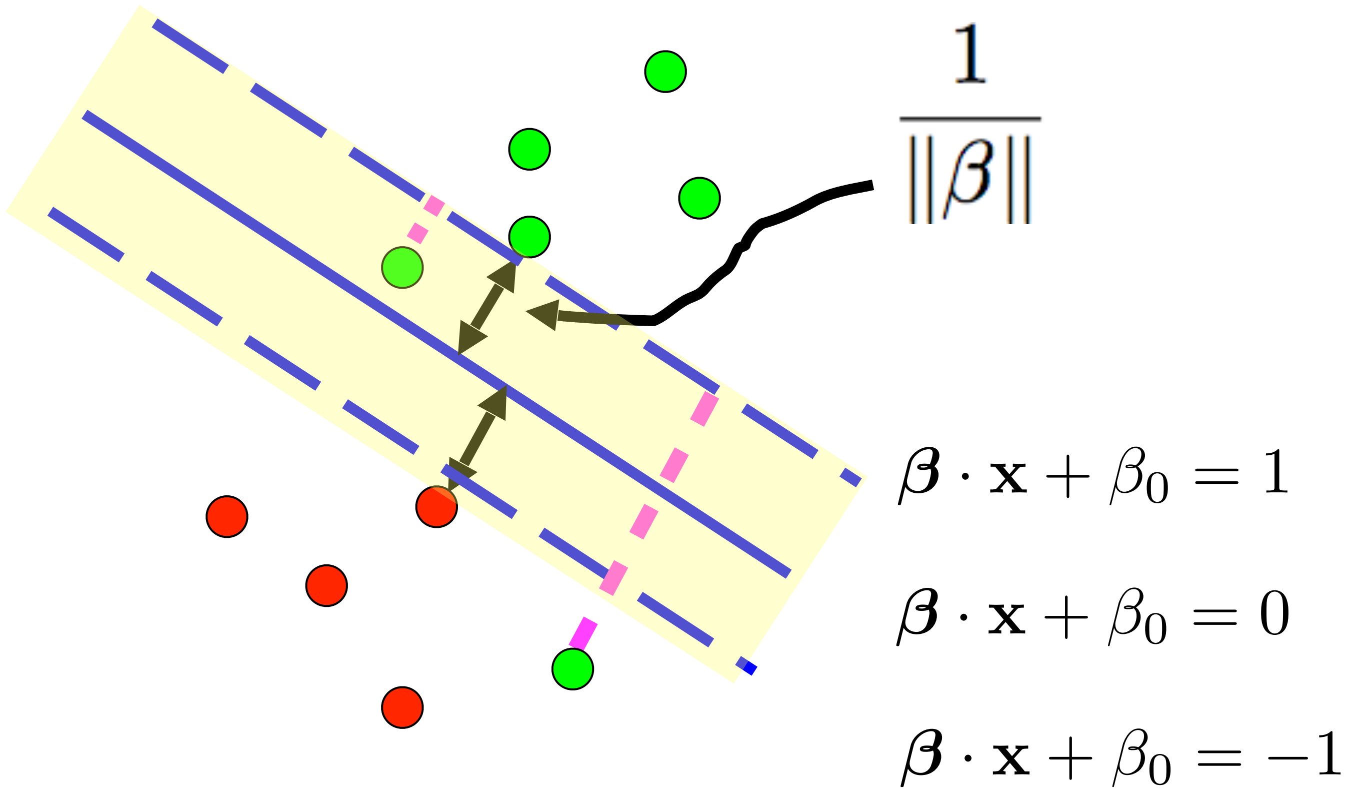


Kernel Machine

Nonlinear SVM for Separable/Non-separable Data

Soft Margin

Linear SVM for Non-separable/Separable Data



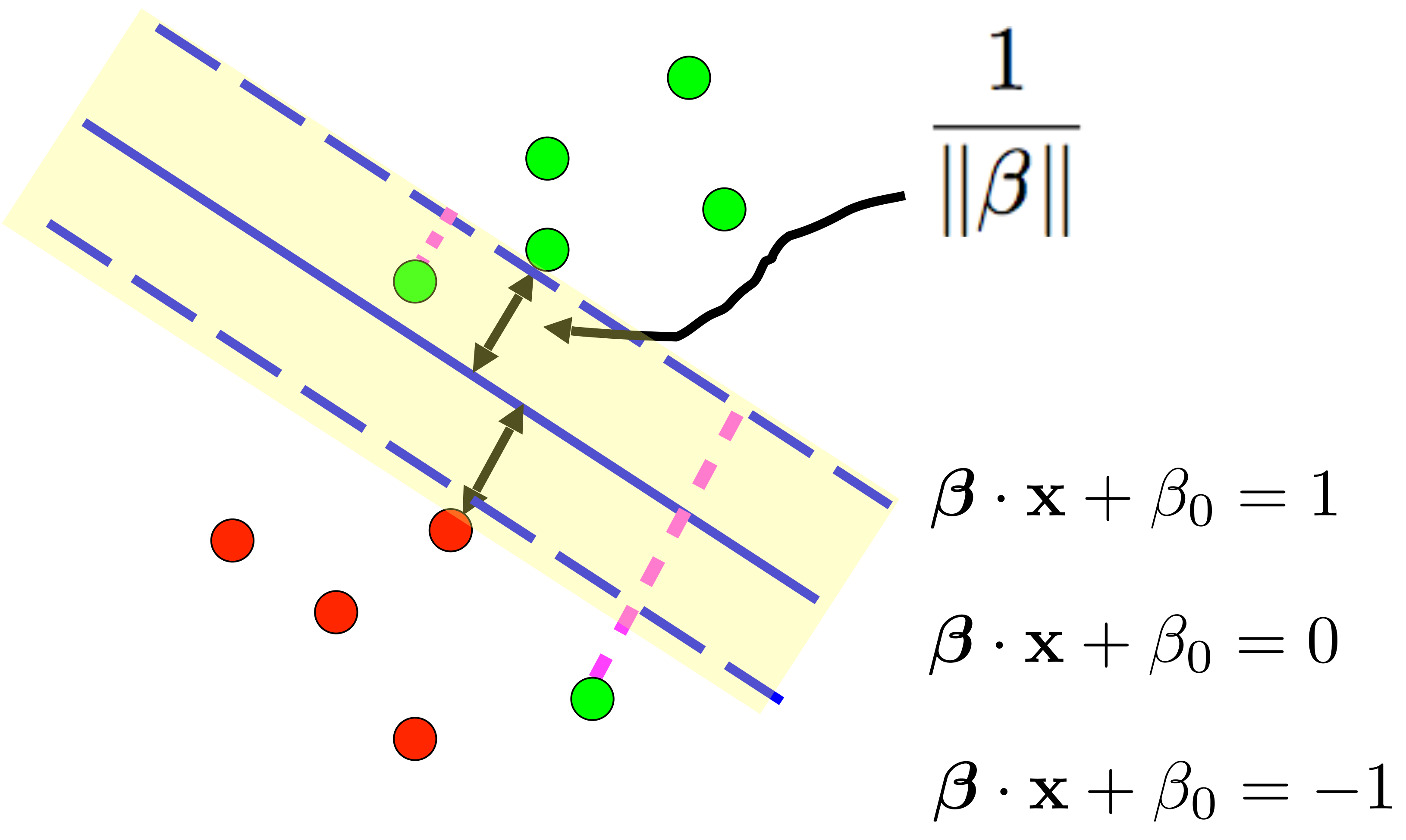
1. Formulate the **Primal** Problem (dim = p+1)
2. Solve the **Dual** Problem (dim = n)
3. **KKT Conditions** link the two sets of solutions
4. **SV**: data points on the dashed lines or on the wrong side of the datelines

Some Practical Issues

1. Binary decision to probability
2. Multiclass SVM

Some Practical Issues

1. **From binary decision to probability**



Run a logistic regression wrt f(x_i).

$$f(\mathbf{x}) = \boldsymbol{\beta} \cdot \mathbf{x} + \beta_0$$

Some Practical Issues

Consider MNIST Data

- **One-vs-all** Fit 10 SVMs
- **One-vs-one** Fit 45 SVMs

Can we formulate the concept of margin as some kind of area/volume of the ball (or some kind of convex region) that separate the K classes? Not a fan of this idea.

2. Multi-Class SVM

Recall how logistic regression and QDA/LDA/NB handle multi-class?

Vanilla extension to multiclass

- **One-vs-all**
- **One-vs-one**

Formulate a multi-class SVM

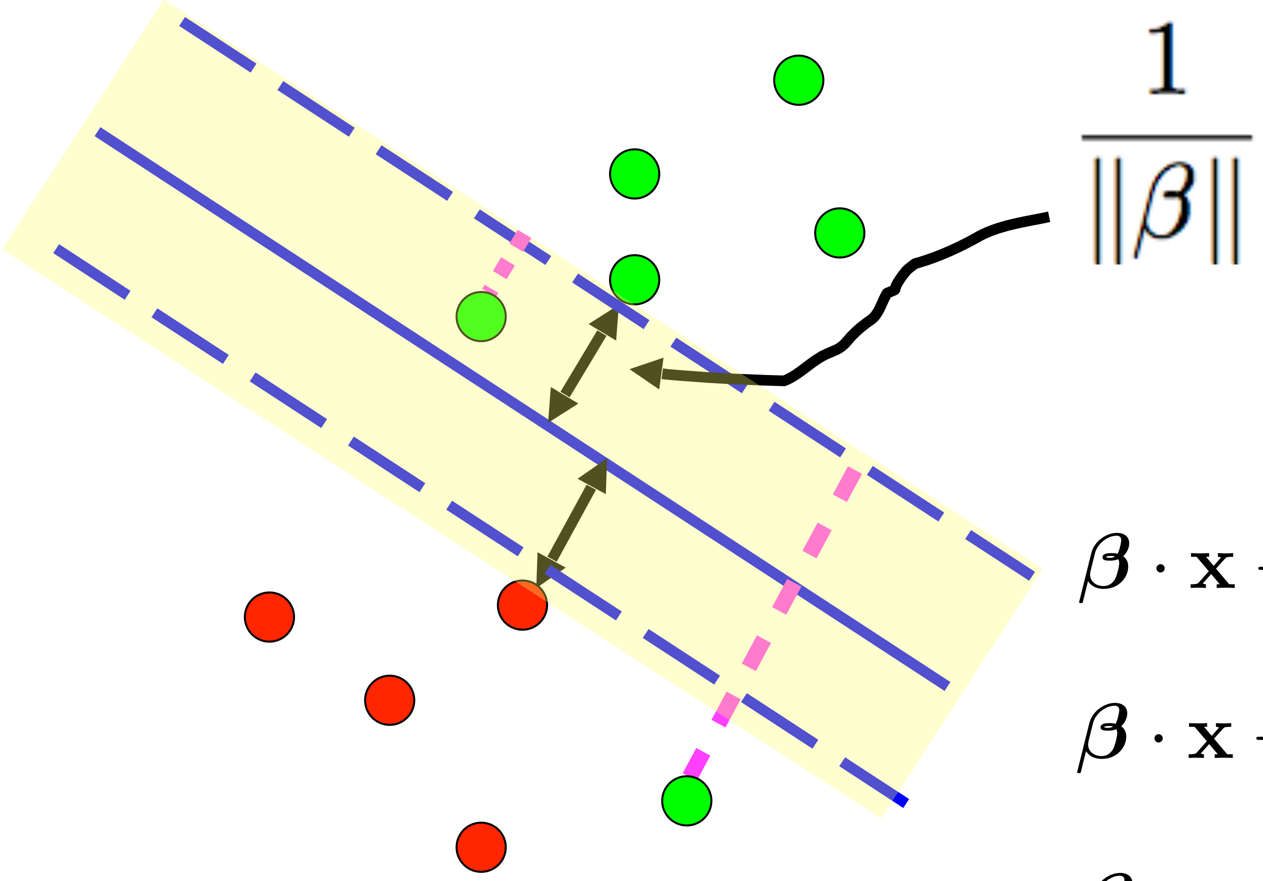
$$\min_{\boldsymbol{\beta},\beta_0,\xi_{1:n}} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \gamma \sum \xi_i$$
$$\text{subj to } y_i(\mathbf{x}_i \cdot \boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i,$$
$$\xi_i \geq 0$$

$$f_k(\mathbf{x}) = \boldsymbol{\beta}_k \cdot \mathbf{x} + \beta_{k0}$$

$$\min_{\boldsymbol{\beta},\beta_0,\xi_{1:n}} \frac{1}{2}\sum_{k=1}^{K} \|\boldsymbol{\beta}_k\|^2 + \gamma \sum \xi_i$$
$$\text{subj to } f_{y_i}(\mathbf{x}_i) - f_y(\mathbf{x}_i) \geq 1 - \xi_i,$$
$$\xi_i \geq 0$$

Some Practical Issues

1. From binary decision to probability



$\beta \cdot \mathbf{x} + \beta_0 = 1$

$\beta \cdot \mathbf{x} + \beta_0 = 0$

$\beta \cdot \mathbf{x} + \beta_0 = -1$

Run a logistic regression wrt f(x_i).

$$f(\mathbf{x}) = \boldsymbol{\beta} \cdot \mathbf{x} + \beta_0$$

2. Multi-Class SVM

Recall how logistic regression and QDA/LDA/NB handle multi-class?

Vanilla extension to multiclass

- **One-vs-all**
- **One-vs-one**

Formulate a multi-class SVM

$$\min_{\boldsymbol{\beta}, \beta_0, \xi_{1:n}} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \gamma \sum \xi_i$$
$$\text{subj to } y_i(\mathbf{x}_i \cdot \boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i,$$
$$\xi_i \geq 0$$

$$f_k(\mathbf{x}) = \boldsymbol{\beta}_k \cdot \mathbf{x} + \beta_{k0}$$

$$\min_{\boldsymbol{\beta}, \beta_0, \xi_{1:n}} \frac{1}{2}\sum_{k=1}^{K}\|\boldsymbol{\beta}_k\|^2 + \gamma \sum \xi_i$$
$$\text{subj to } f_{y_i}(\mathbf{x}_i) - f_y(\mathbf{x}_i) \geq 1 - \xi_i,$$
$$\xi_i \geq 0$$

Linear SVM

Primal

$$\min_{\boldsymbol{\beta},\beta_0,\xi_{1:n}} \quad \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \gamma \sum \xi_i$$

$$\text{subj to } y_i(\mathbf{x}_i \cdot \boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Dual

$$\max_{\lambda_{1:n}} \quad \sum \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i \lambda_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\text{subj to } \sum \lambda_i y_i = 0, \ \gamma \geq \lambda_i \geq 0$$

Prediction

$$\text{sign}\Big(\sum_{i \in N_s} \lambda_i y_i (\mathbf{x}_i \cdot \mathbf{x}_*) + \hat{\beta}_0\Big)$$

Note that we **do not need to compute beta's**. In practice, we just need to solve for lambda_i's from the Dual, and then use lambda_i's to make predictions.

Linear SVM

Primal

$$\min_{\boldsymbol{\beta},\beta_0,\xi_{1:n}} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \gamma \sum \xi_i$$

$$\text{subj to } y_i(\mathbf{x}_i \cdot \boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i,$$

$$\xi_i \geq 0$$

Dual

$$\max_{\lambda_{1:n}} \sum \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i \lambda_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\text{subj to } \sum \lambda_i y_i = 0, \ \ \gamma \geq \lambda_i \geq 0$$

Prediction

$$\text{sign}(\sum_{i \in N_s} \lambda_i y_i (\mathbf{x}_i \cdot \mathbf{x}_*) + \hat{\beta}_0)$$

Nonlinear SVM

Nonlinear Feature Mapping

$$(x_1, x_2) \Longrightarrow (x_1, x_2, x_1 x_2, x_1^2, x_2^2)$$

$$\mathbf{x} \Longrightarrow \Phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \dots)$$

Kernel Trick: We do not even need to construct the mapping. All we need is $K_\Phi(\mathbf{x}, \mathbf{z}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle$

Dual

$$\max_{\lambda_{1:n}} \sum \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i \lambda_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subj to } \sum \lambda_i y_i = 0, \ \ \gamma \geq \lambda_i \geq 0$$

Prediction

$$\text{sign}(\sum_{i \in N_s} \lambda_i y_i K(\mathbf{x}_i, \mathbf{x}_*) + \hat{\beta}_0)$$

The Kernel Function K

The bivariate function **K** is often referred to as the **reproducing kernel** (r.k.) function. We can view K(x, z) as a similarity measure between x and z, which generalizes the ordinary Euclidean inner product between x and z.

Popular kernel functions

- *d*-th degree polynomial

$$K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x} \cdot \mathbf{z})^d$$

- Gaussian kernel

$$K(\mathbf{x}, \mathbf{z}) = \exp(-\sigma \|\mathbf{x} - \mathbf{z}\|^2)$$

Kernel Trick: We only need the feature space to exist, as well as the K function.

How to Choose the K function?

1. Construct the feature mapping then we have the K function

The Kernel Function K

The bivariate function **K** is often referred to as the **reproducing kernel** (r.k.) function. We can view K(x, z) as a similarity measure between x and z, which generalizes the ordinary Euclidean inner product between x and z.

Popular kernel functions

- *d*-th degree polynomial

$$K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x} \cdot \mathbf{z})^d$$

- Gaussian kernel

$$K(\mathbf{x}, \mathbf{z}) = \exp(-\sigma \|\mathbf{x} - \mathbf{z}\|^2)$$

Kernel Trick: We only need the feature space to exist, as well as the K function.

How to Choose the K function?

1. Construct the feature mapping then we have the K function

2. Can we use any symmetric bivariate function as K? K must satisfy the **Mercer's condition**: **symmetric, semi-positive definite function**

$$K(x, z) = \exp(-\sigma x^2) \exp(-\sigma z^2) \exp(2\sigma x z)$$
$$= \exp(-\sigma x^2) \exp(-\sigma z^2) \sum_{k=0}^{\infty} \frac{2^k x^k z^k}{k!}$$

The Kernel Function K

The bivariate function **K** is often referred to as the **reproducing kernel** (r.k.) function. We can view K(x, z) as a similarity measure between x and z, which generalizes the ordinary Euclidean inner product between x and z.

Popular kernel functions

- *d*-th degree polynomial

$$K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x} \cdot \mathbf{z})^d$$

- Gaussian kernel

$$K(\mathbf{x}, \mathbf{z}) = \exp(-\sigma \|\mathbf{x} - \mathbf{z}\|^2)$$

Kernel Trick: We only need the feature space to exist, as well as the K function.

How to Choose the K function?

1. Construct the feature mapping then we have the K function

2. Can we use any symmetric bivariate function as K? K must satisfy the **Mercer's condition**: **symmetric, semi-positive definite function**

$$K(x, z) = \exp(-\sigma x^2) \exp(-\sigma z^2) \exp(2\sigma x z)$$

$$= \exp(-\sigma x^2)\exp(-\sigma z^2) \sum_{k=0}^{\infty} \frac{2^k x^k z^k}{k!}$$

3. Who cares. Use any symmetric function that can capture the similarity between x and z for your application/task (check our discussion on distance for KNN)

Convex Optimization

Primal

$$\min_{\boldsymbol{\beta},\beta_0,\xi_{1:n}} \quad \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \gamma \sum \xi_i$$

$$\text{subj to } y_i(\mathbf{x}_i \cdot \boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i,$$

$$\xi_i \geq 0$$

Dual

$$\max_{\lambda_{1:n}} \quad \sum \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\text{subj to } \sum \lambda_i y_i = 0, \;\; \gamma \geq \lambda_i \geq 0$$

Prediction

$$\text{sign}\Big(\sum_{i \in N_s} \lambda_i y_i (\mathbf{x}_i \cdot \mathbf{x}_*) + \hat{\beta}_0\Big)$$

Loss + Penalty

Primal

$$\min_{\boldsymbol{\beta},\beta_0} \sum_{i=1}^{n} \left[1 - y_i f(\mathbf{x}_i)\right]_+ + \nu \|\boldsymbol{\beta}\|^2$$

$$\boxed{f(\mathbf{x}) = \boldsymbol{\beta} \cdot \mathbf{x} + \beta_0}$$

Dual

$$f(\mathbf{x}) = \sum_i \lambda_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + \beta_0 \qquad \boldsymbol{\beta} = \sum_i \lambda_i y_i \mathbf{x}_i$$

$$f(\mathbf{x}) = \sum_i \lambda_i y_i K(\mathbf{x}_i, \mathbf{x}) + \beta_0$$

$$= \sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + \alpha_0 \qquad \|\boldsymbol{\beta}\|^2 = \boldsymbol{\alpha}^t \mathbf{K}_{n \times n} \boldsymbol{\alpha}$$

$$= \alpha_1 K(\mathbf{x}_1, \mathbf{x}) + \cdots + \alpha_n K(\mathbf{x}_n, \mathbf{x}) + \alpha_0$$

The Kernel Machine

Kernel Model

$$f(\mathbf{x}) = \alpha_1 K(\mathbf{x}_1, \mathbf{x}) + \cdots + \alpha_n K(\mathbf{x}_n, \mathbf{x}) + \alpha_0$$

Matrix Representation

$$\begin{pmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_n) \end{pmatrix} = \begin{pmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \cdots & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \cdots & K(\mathbf{x}_2, \mathbf{x}_n) \\ \cdots & \cdots & \cdots & \cdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & \cdots & K(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} + \alpha_0$$

$$= \mathbf{K}\boldsymbol{\alpha} + \alpha_0$$

Parameter Estimation via Regularization

$$\min_{\boldsymbol{\alpha}, \alpha_0} \sum_{i=1}^{n} \frac{1}{n} L\big(y_i, f(\mathbf{x}_i)\big) + \nu \boldsymbol{\alpha}^t \mathbf{K} \boldsymbol{\alpha}$$

The Kernel Machine

For this formulation, given a similarity function K(x, z) that doesn't need to satisfy the **Mercer's condition**, we just assume our model like this, and then estimate coefficients alpha's with a (generalized) ridge penalty.

Kernel Model

$$f(\mathbf{x}) = \alpha_1 K(\mathbf{x}_1, \mathbf{x}) + \cdots + \alpha_n K(\mathbf{x}_n, \mathbf{x}) + \alpha_0$$

Matrix Representation

$$\begin{pmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_n) \end{pmatrix} = \begin{pmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \cdots & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \cdots & K(\mathbf{x}_2, \mathbf{x}_n) \\ \cdots & \cdots & \cdots & \cdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & \cdots & K(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} + \alpha_0$$

$$= \mathbf{K}\boldsymbol{\alpha} + \alpha_0$$

Parameter Estimation via Regularization

$$\min_{\boldsymbol{\alpha}, \alpha_0} \sum_{i=1}^{n} \frac{1}{n} L\big(y_i, f(\mathbf{x}_i)\big) + \nu \boldsymbol{\alpha}^t \mathbf{K} \boldsymbol{\alpha}$$

For SVM, we have
Hinge-Loss + Ridge-Penalty.
For SVM, the sparsity is from Hinge-Loss

The Kernel Machine

For this formulation, given a similarity function K(x, z) that doesn't need to satisfy the **Mercer's condition**, we just assume our model like this, and then estimate coefficients alpha's with a (generalized) ridge penalty.

Kernel Model

$$f(\mathbf{x}) = \alpha_1 K(\mathbf{x}_1, \mathbf{x}) + \cdots + \alpha_n K(\mathbf{x}_n, \mathbf{x}) + \alpha_0$$

Matrix Representation

$$\begin{pmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_n) \end{pmatrix} = \begin{pmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \cdots & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \cdots & K(\mathbf{x}_2, \mathbf{x}_n) \\ \cdots & \cdots & \cdots & \cdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & \cdots & K(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} + \alpha_0$$

$$= \mathbf{K}\boldsymbol{\alpha} + \alpha_0$$

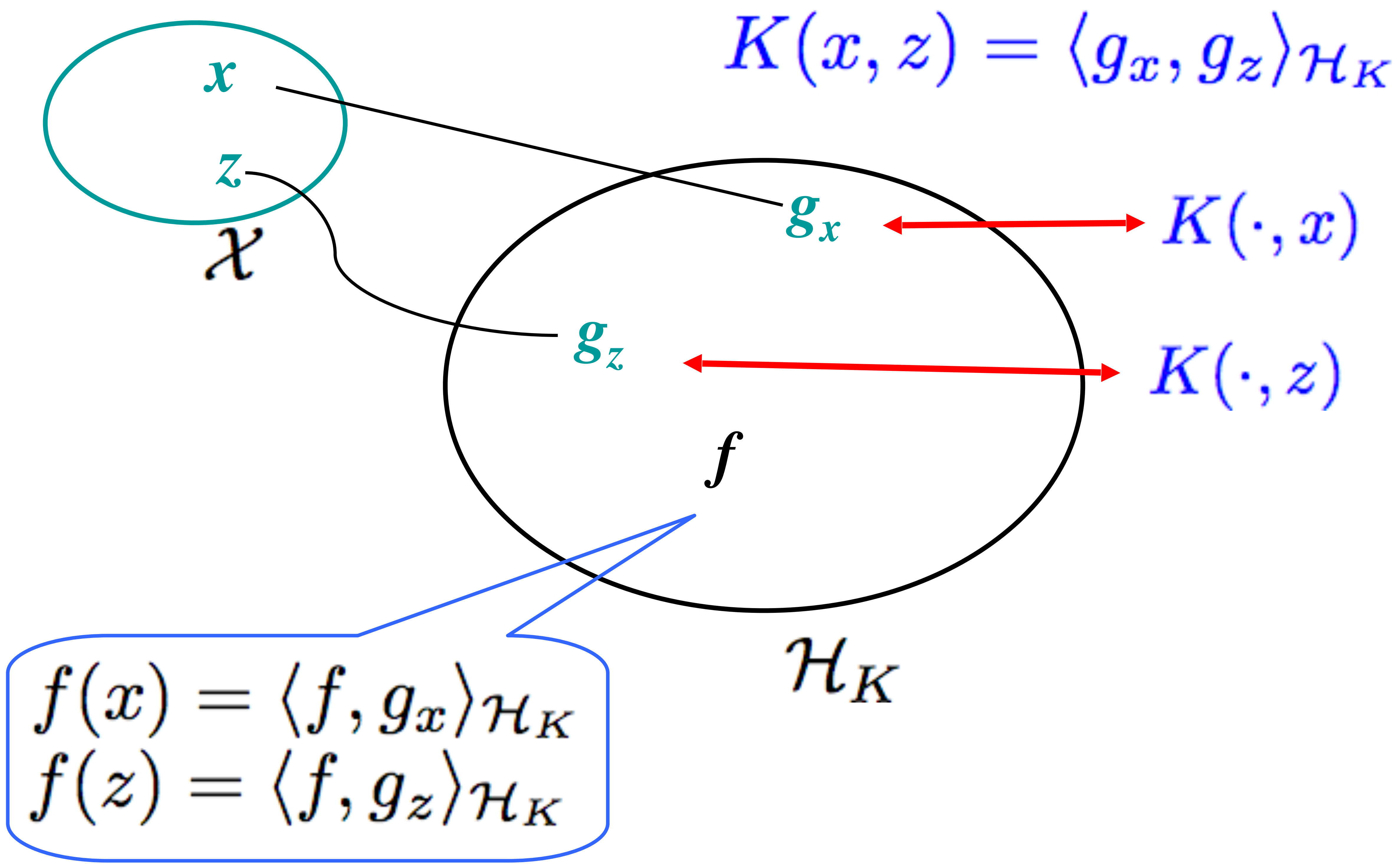
If K satisfies the Mercer's condition, what more we can say about this framework?

Parameter Estimation via Regularization

$$\min_{\boldsymbol{\alpha}, \alpha_0} \sum_{i=1}^{n} \frac{1}{n} L(y_i, f(\mathbf{x}_i)) + \nu \boldsymbol{\alpha}^t \mathbf{K} \boldsymbol{\alpha}$$

For SVM, we have
Hinge-Loss + Ridge-Penalty.
For SVM, the sparsity is from Hinge-Loss

Reproducing Kernel Hilbert Space (RKHS)



Reproducing Property

Representer Theorem

$$\arg\min_{f \in \mathcal{H}_K} \sum_{i=1}^{n} \frac{1}{n} L(y_i, f(\mathbf{x}_i)) + \nu \|f\|_K^2$$
$$= \alpha_1 K(\mathbf{x}_1, \mathbf{x}) + \cdots + \alpha_n K(\mathbf{x}_n, \mathbf{x}) + \alpha_0$$

Proof : Let $\mathcal{H}_1 = \text{span}\{K(\cdot, x_1), \ldots, K(\cdot, x_n)\}$ and $\mathcal{H}_2 = \mathcal{H}_1^{\perp}$. Then for any function $f \in \mathcal{H}_K$, we can write

$$f = f_1 + f_2, \quad \text{where } f_1 \in \mathcal{H}_1 \text{ and } f_2 \in \mathcal{H}_2.$$

Then we have the following

1. $\|f\|^2 \geq \|f_1\|^2$;
2. $f(x_i) = f_1(x_i)$ for $i = 1, \ldots, n$, because

$$\langle f, K(\cdot, x_i) \rangle_{\mathcal{H}_K} = \langle f_1 + f_2, K(\cdot, x_i) \rangle_{\mathcal{H}_K} = \langle f_1, K(\cdot, x_i) \rangle_{\mathcal{H}_K}.$$

That is $\Omega(f) \geq \Omega(f_1)$. So to minimize $\Omega(f)$, it suffices to focus on subspace $\mathcal{H}_1$. (Does the proof sound familiar? Yes, it follows the same argument as the one in the proof for smoothing splines.)

Summary: SVMs

Primal

$$\min_{\boldsymbol{\beta},\beta_0,\xi_{1:n}} \quad \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \gamma \sum \xi_i$$

$$\text{subj to } y_i(\mathbf{x}_i \cdot \boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i,$$
$$\xi_i \geq 0$$

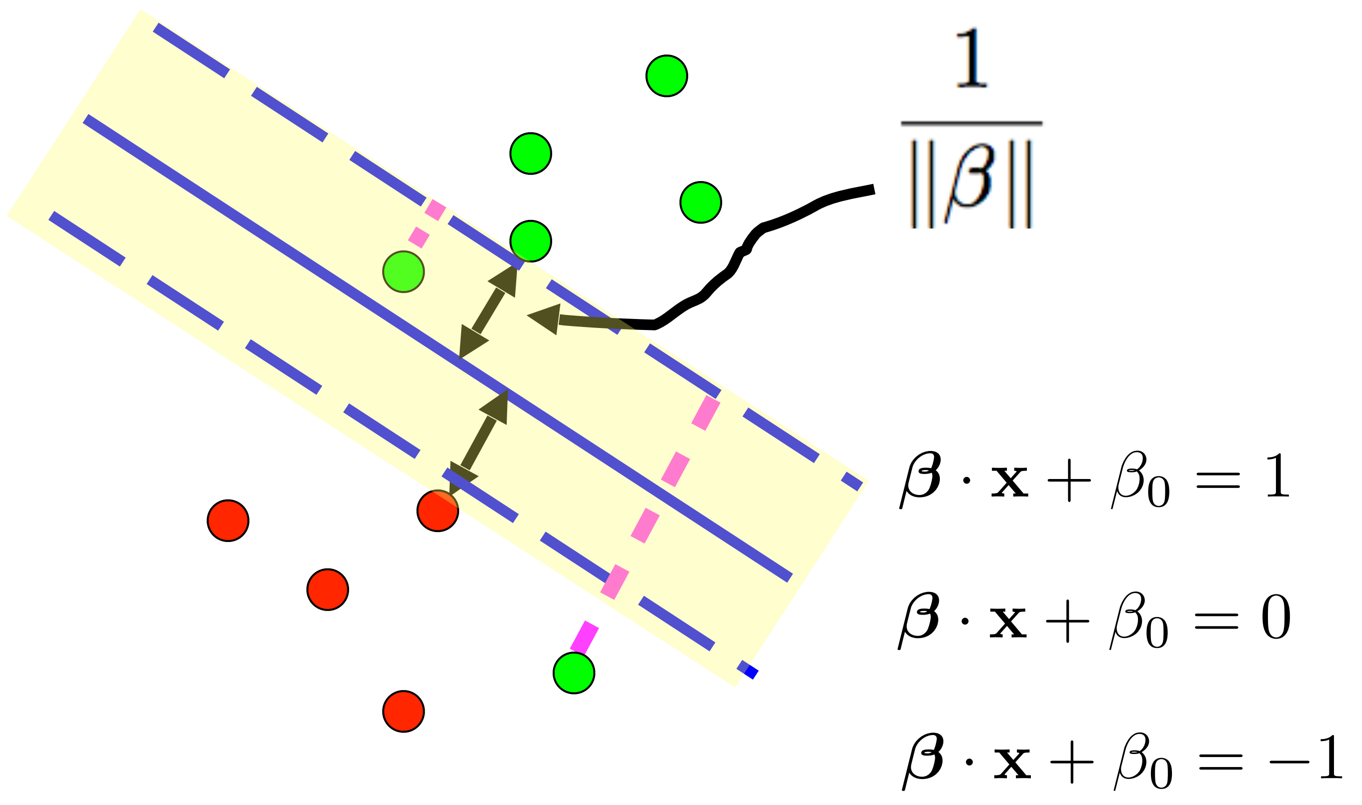
Dual

$$\max_{\lambda_{1:n}} \quad \sum \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i \lambda_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\text{subj to } \sum \lambda_i y_i = 0, \;\; \gamma \geq \lambda_i \geq 0$$

Prediction

$$\text{sign}\Big(\sum_{i \in N_s} \lambda_i y_i (\mathbf{x}_i \cdot \mathbf{x}_*) + \hat{\beta}_0\Big)$$



Primal

$$\min_{\boldsymbol{\beta},\beta_0} \sum_{i=1}^{n} [1 - y_i f(\mathbf{x}_i)]_+ + \nu\|\boldsymbol{\beta}\|^2$$

$$f(\mathbf{x}) = \boldsymbol{\beta} \cdot \mathbf{x} + \beta_0$$

1. Formulate the **Primal** Problem (dim = p+1)
2. Solve the **Dual** Problem (dim = n)
3. **KKT Conditions** link the two sets of solutions
4. **SV**: data points on the dashed lines or on the wrong side of the datelines

Popular kernel functions

- *d*-th degree polynomial

$$K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x} \cdot \mathbf{z})^d$$

- Gaussian kernel

$$K(\mathbf{x}, \mathbf{z}) = \exp(-\sigma\|\mathbf{x} - \mathbf{z}\|^2)$$

Summary: The Kernel Machine

Here K(x, z) is any symmetric function reflecting the similarity between x and z, which doesn't need to satisfy the **Mercer's condition.**

Kernel Model

$$f(\mathbf{x}) = \alpha_1 K(\mathbf{x}_1, \mathbf{x}) + \cdots + \alpha_n K(\mathbf{x}_n, \mathbf{x}) + \alpha_0$$

Matrix Representation

$$\begin{pmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_n) \end{pmatrix} = \begin{pmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \cdots & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \cdots & K(\mathbf{x}_2, \mathbf{x}_n) \\ \cdots & \cdots & \cdots & \cdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & \cdots & K(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} + \alpha_0$$

$$= \mathbf{K}\boldsymbol{\alpha} + \alpha_0$$

Parameter Estimation via Regularization

Here we can employ any loss function for regression/classification, and any penalty function on alpha.

$$\min_{\boldsymbol{\alpha}, \alpha_0} \sum_{i=1}^{n} \frac{1}{n} L(y_i, f(\mathbf{x}_i)) + \nu \boldsymbol{\alpha}^t \mathbf{K} \boldsymbol{\alpha}$$