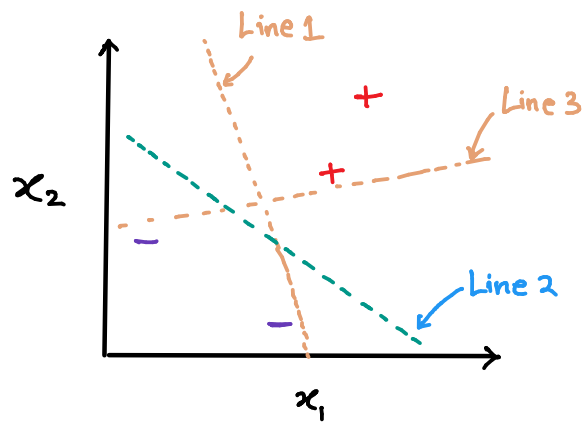# Support Vector Machine (SVM)

- In kernel methods, we have introduced kernel ridge regression

- Kernel logistic regression was introduced for classification but it is rarely used

- Consider a binary classification with $y \in \{-1, 1\}$ and a linearly separable dataset
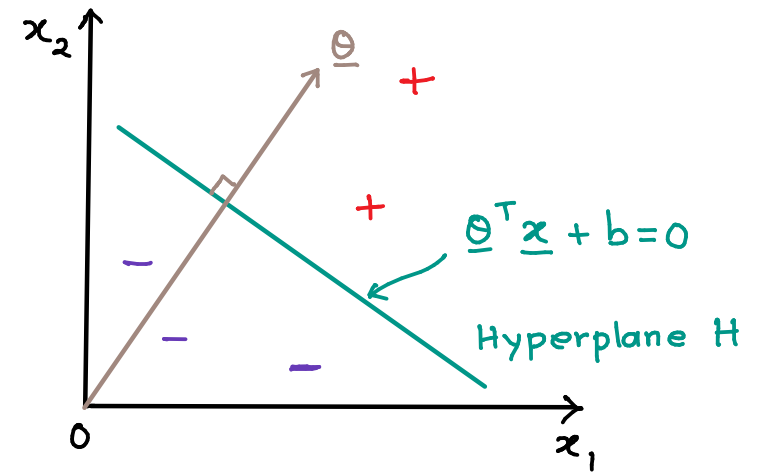

- For this dataset shown, how do you draw a line to separate the positive '+' data points from the negative '−' data points?
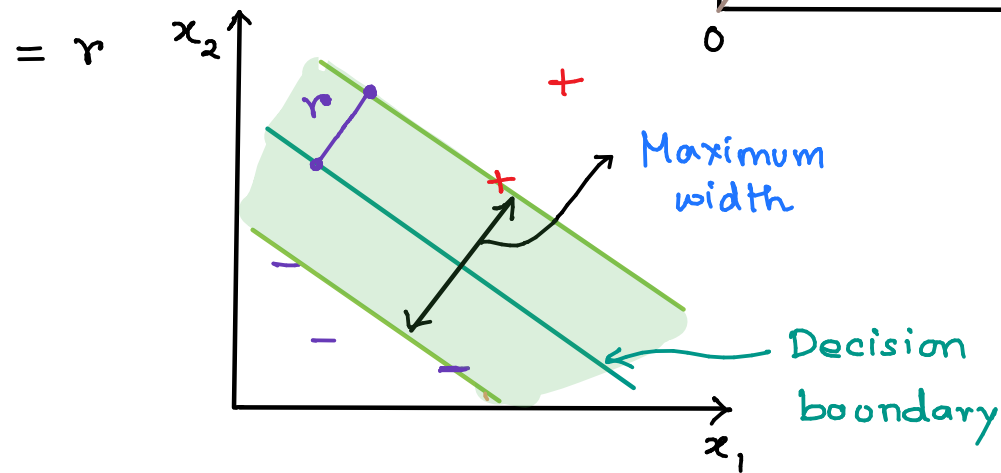
  − Several possible choices
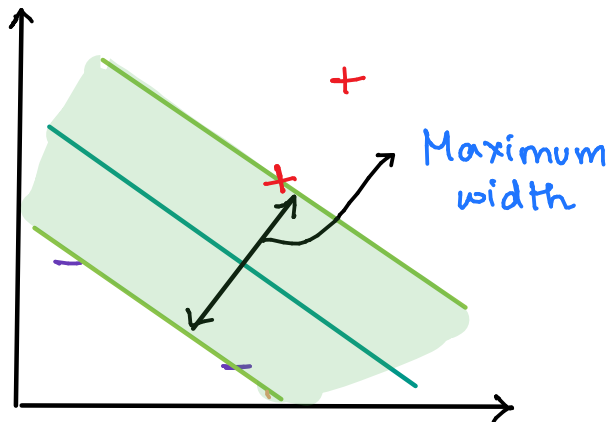
- The best line (in 2D) that separates the two classes lies midway of the widest street that separates the '+' samples from '−' samples

- The goal is to find a hyperplane H that maximizes the width of the street

$$H = \{ \underline{x} : \underline{\Theta}^T \underline{x} + b = 0 \}$$

- width of the street from decision boundary $= r$


- Let's construct an optimization problem:

  - The variables to optimize are: $\underline{\Theta}$, b, and r

    $\underline{\Theta}$, b determine the placement of hyperplane

    r determines the width of street

- The objective to maximize the width r is subject to the following constraints:

1. All '+' points should lie on the positive side of hyperplane and their distance to the hyperplane H must be $\geqslant r$

2. All '−' points should lie on the negative side of hyperplane and their distance to the hyperplane H must be $\geqslant r$

3. The value of r must be non-negative

x_2

Θ

$+$

r

$+$

$\Theta^T \underline{x} + b = 1$

$-$

$-$

$\Theta^T \underline{x} + b = 0$

$-$

x_1
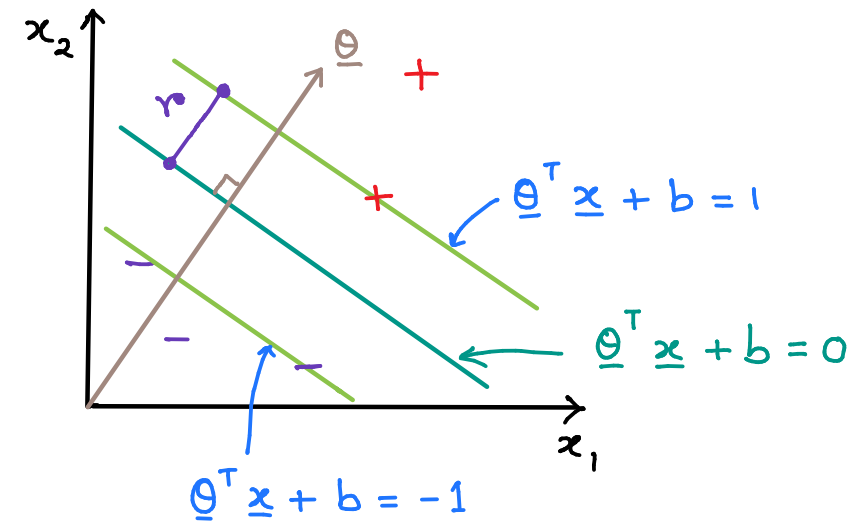
$\Theta^T \underline{x} + b = -1$

- Without any loss of generality, we can assume that the equations of the hyperplanes representing the street boundaries are:

$$\Theta^T \underline{x} + b = 1 \quad \text{(Margin for the positive side)}$$

$$\Theta^T \underline{x} + b = -1 \quad \text{(Margin for the negative side)}$$
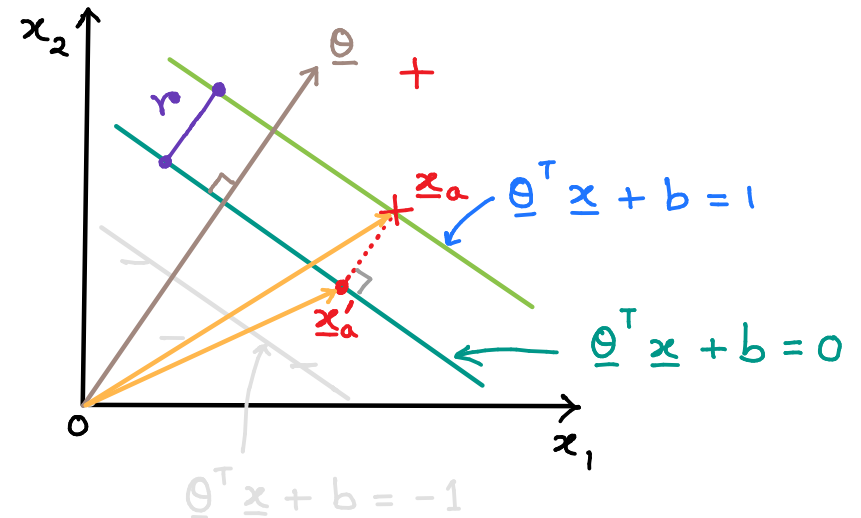
You can imagine that $\underline{x}$ is scaled s.t. $\Theta^T \underline{x} + b = \pm 1$ at the boundaries

- We now want to calculate the width r in terms of $\underline{\theta}$

  - Consider a pt $\underline{x}_a$ on the boundary. The orthogonal projection of $\underline{x}_a$ on H is the point $\underline{x}_a'$

$$\underline{x}_a = \underline{x}_a' + r \underbrace{\frac{\underline{\theta}}{\|\underline{\theta}\|_2}}_{\text{← direction of } \underline{\theta}}$$

$$\Rightarrow \quad \underline{x}_a' = \underline{x}_a - r \frac{\underline{\theta}}{\|\underline{\theta}\|_2}$$

  - Since $\underline{x}_a'$ lies on the hyperplane H, it must be:

$$\underline{\theta}^T \underline{x}_a' + b = 0$$

$$\Rightarrow \quad \underline{\theta}^T \left(\underline{x}_a - r \frac{\underline{\theta}}{\|\underline{\theta}\|_2} \right) + b = 0$$

$$\Rightarrow \quad \underbrace{\underline{\theta}^T \underline{x}_a + b}_{= 1 \ (\text{by defn})} - r \underbrace{\frac{\underline{\theta}^T \underline{\theta}}{\|\underline{\theta}\|_2}}_{\|\underline{\theta}\|_2^2} = 0$$

$$\Rightarrow \quad 1 - r \|\underline{\theta}\|_2 = 0 \qquad \Rightarrow \quad \boxed{r = \frac{1}{\|\underline{\theta}\|_2}}$$

- Now lets look at the constraints:

1. All '+' points should lie on the positive side of hyperplane and their distance to the hyperplane H must be $\geq r$ $\Rightarrow$ rewrite since $y_i = +1$ $\curvearrowright$ $\underline{\theta}^T \underline{x}_i + b \geq 1$ $y_i(\underline{\theta}^T \underline{x}_i + b) \geq 1$

2. All '−' points should lie on the negative side of hyperplane and their distance to the hyperplane H must be $\geq r$ $\Rightarrow$ rewrite since $y_i = -1$ $\curvearrowright$ $\underline{\theta}^T \underline{x}_i + b \leq -1$ $y_i(\underline{\theta}^T \underline{x}_i + b) \geq 1$

3. The value of r must be non-negative $\Rightarrow$ $r \geq 0$

- Putting everything together, we have the following optimization problem:

$$\underset{r, \underline{\theta}, b}{\text{maximize}} \; r \quad \text{subject to constraints} \quad y_i(\underline{\theta}^T \underline{x}_i + b) \geq 1 \quad \forall \, i$$
$$r \geq 0$$

$\Rightarrow$

$$\boxed{\underset{\underline{\theta}, b}{\text{maximize}} \; \frac{1}{\|\underline{\theta}\|}} \quad \text{subject to constraints} \quad \boxed{\begin{array}{l} y_i(\underline{\theta}^T \underline{x}_i + b) \geq 1 \\ \text{for} \quad i = 1, \ldots, N \end{array}}$$

- Instead of maximizing $\frac{1}{\|\underline{\Theta}\|_2}$, we can instead minimize $\|\underline{\Theta}\|_2$ or even the squared norm $\|\underline{\Theta}\|_2^2$

  - The objective then becomes

  HARD MARGIN $\left\{\begin{array}{l} \text{minimize} \quad \frac{1}{2}\|\underline{\Theta}\|_2^2 \\ \underline{\Theta}, b \\ \text{s.t.} \quad y_i\left(\underline{\Theta}^T\underline{x}_i + b\right) \geq 1 \quad \text{for } i=1,\ldots,N \end{array}\right.$

  SVM

- The above optimization is the standard formulation of hard-margin SVMs

  - "Hard" because the formulation does not allow for any violations of the street margins (i.e. constraints)

  - Only applicable to a linearly separable data

  Hard margin SVMs can be applied
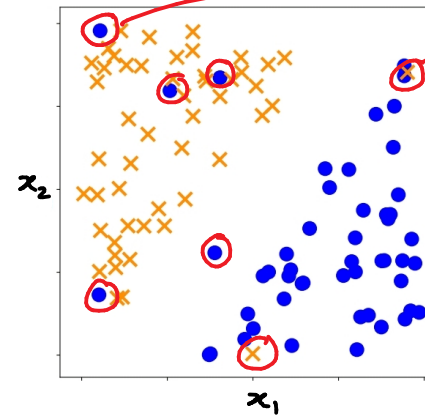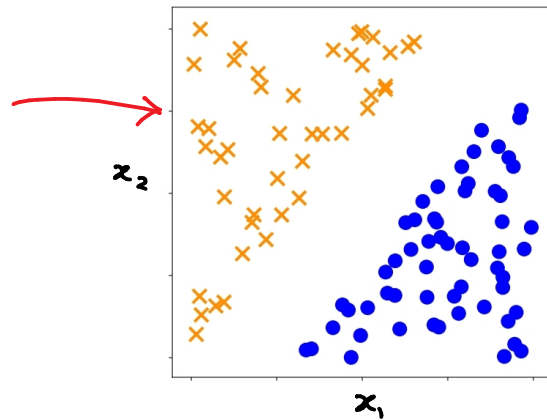
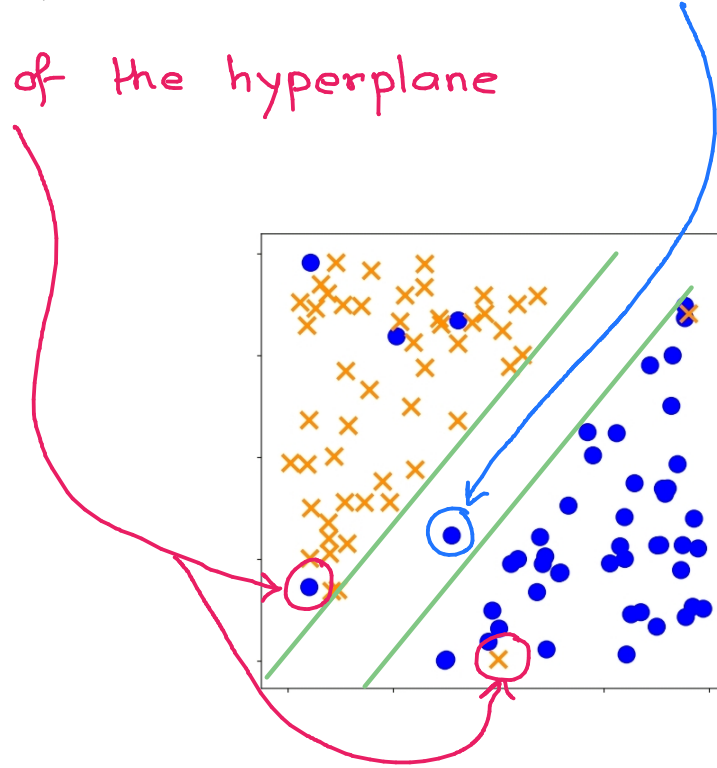 violation of constraints.

  Hard-margin SVM cannot be applied!

Soft-margin SVMs

- In the case where data is not linearly separable, we would want to allow some examples to fall with the street margins, or even to be on the wrong side of the hyperplane

- We now want our SVM to be not so sensitive to outliers and it should work even in the presence of data that is not linearly separable

Soft-margin SVMs

- A soft-margin SVM modifies the constraints from the hard-margin SVM by allowing some points to violate the margin boundaries.

- It introduces **slack variables** ξ_i, one for each training point, into the constraints:

$$y_i(\underline{\theta}^T\underline{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

- The slack variable ξ_i measures the distance of a positive example $\underline{x}_i^+$ to the positive margin hyperplane $\underline{\theta}^T\underline{x} + b = 1$ when x_i^+ is on the wrong side or between margin boundaries

- The constraints are now less-strict, because each point $\underline{x}_i$ need only be a "distance" of $1 - \xi_i$ from the separating hyperplane instead of a hard "distance" r
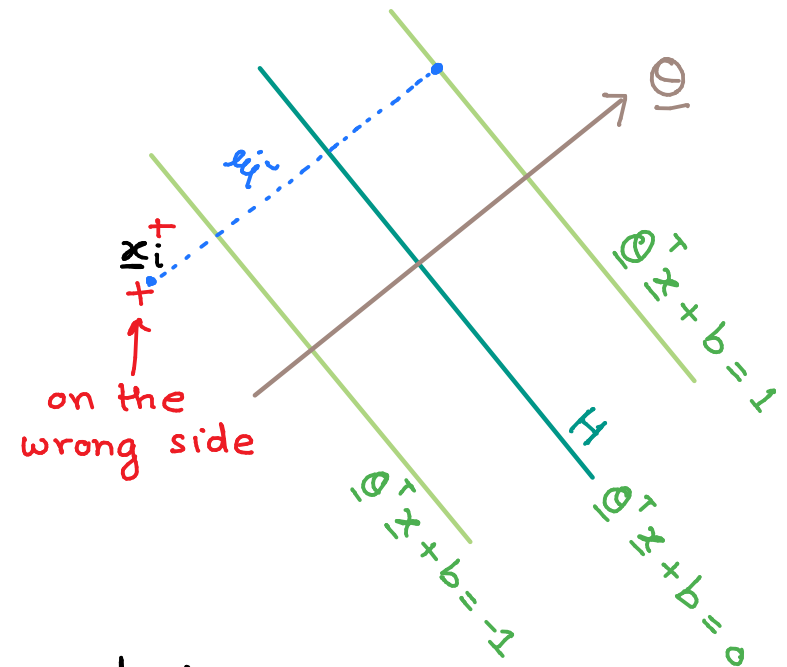
- However, we must restrain the values of ξ_i's, else any point can violate the margin by arbitrarily large distance

- Therefore, we modify our objective function to penalize the slack variables

$$\underset{\underline{\Theta}, b, \xi_1, \ldots, \xi_N}{\text{minimize}} \quad \frac{1}{2} \|\underline{\Theta}\|_2^2 + C \sum_{i=1}^{N} \xi_i$$

hyperparameter (tuned through cross-validation)

- The full optimization problem for soft-margin SVM becomes:

$$\underset{\underline{\Theta}, b, \xi_1, \ldots, \xi_N}{\text{minimize}} \quad \frac{1}{2} \|\underline{\Theta}\|_2^2 + C \sum_{i=1}^{N} \xi_i$$

$$\text{s.t.} \quad \left. \begin{array}{c} y_i (\underline{\Theta}^T x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{array} \right\} \text{ for } i = 1, \ldots, N$$

- $C \to \infty \Rightarrow \xi_i$'s $= 0 \Rightarrow$ Hard-margin SVM

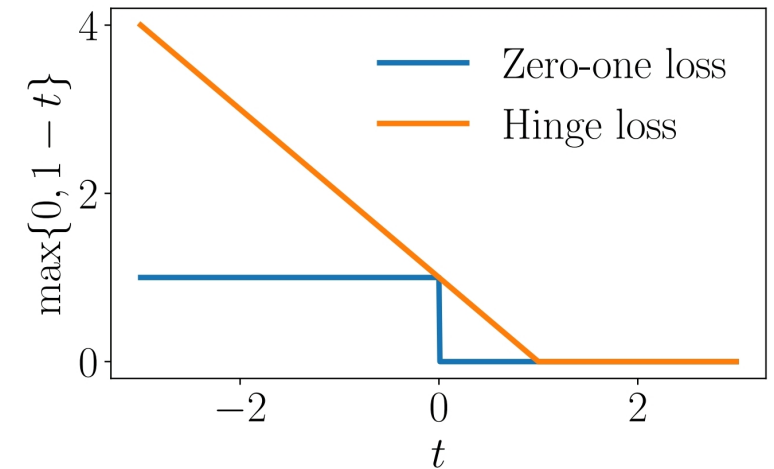 $C \to$ small $\Rightarrow$ less sensitive to outliers

A loss function view of soft-margin SVMs

- One might ask what is the loss function in SVM?

- An equivalent way of deriving soft-margin SVMs is using HINGE loss

$$\mathcal{L}(t) = \max\{0, 1-t\}$$

Replace $t \rightarrow y(\underline{\theta}^T\underline{x} + b)$

$$\mathcal{L}(\underline{\theta}) = \max\{0, 1 - y(\underline{\theta}^T\underline{x} + b)\}$$

- If $\underline{\theta}^T\underline{x}$ is on the correct side (based on the corresponding label y) of the hyperplane H, and farther than distance 1, then hinge loss is zero

- For a given training set $\{(\underline{x}_i, y_i)\}_{i=1}^{N}$, we can seek to minimize hinge loss while regularizing the objective with ℓ_2-regularization.

$$\min_{\underline{\theta}, b} \quad \underbrace{\frac{1}{2}\|\underline{\theta}\|_2^2}_{\text{regularizer}} + C\underbrace{\sum_{i=1}^{N} \max\{0, 1 - y_i(\underline{\theta}^T\underline{x}_i + b)\}}_{\text{loss term}} \quad \Longleftarrow \quad \text{Unconstrained optimization}$$

Dual SVM

- The SVM discussed till now, in terms of $\underline{\Theta}$ and b, is called primal SVM

- Here $\underline{\Theta}$ is of the same dimension as input features $\underline{x}$, which means that the number of parameters of the optimization grows with # of features

- In the dual view, we consider an equivalent optimization that is independent of the number of features. Instead, the number of parameters increases with the number of training examples

- For the dual view, we write out the Lagrangian of the soft-margin SVM
  (we do not use the hinge loss as the max function is discontinuous)

$$L\left(\underline{\Theta}, b, \underline{\xi}, \underline{\alpha}, \underline{\gamma}\right) = \frac{1}{2}\underline{\Theta}^T\underline{\Theta} + c\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left(y_i\left(\underline{\Theta}^T\underline{x}_i + b\right) - 1 + \xi_i\right)$$
$$- \sum_{i=1}^{N}\gamma_i\,\xi_i$$

$$\text{Dual}: \quad \max L\left(\underline{\Theta}, b, \underline{\xi}, \underline{\alpha}, \underline{\gamma}\right)$$
$$\underline{\alpha} \geqslant 0, \; \underline{\gamma} \geqslant 0$$

By differentiating the Lagrangian L w.r.t three primal variables $\underline{\Theta}, b,$ and ξ_i respectively, and setting the derivatives to zero, we obtain:

$$\frac{\partial L}{\partial \underline{\Theta}} = 0 \Rightarrow \underline{\Theta}^T - \sum_{i=1}^{N} \alpha_i y_i \underline{x}_i^T = 0 \Rightarrow \underline{\Theta} = \underbrace{\sum_{i=1}^{N} \alpha_i y_i \underline{x}_i}$$

$\underline{\Theta}$ is going to be a weighted linear combination of positive-class $\underline{x}_i$'s and negative-class $\underline{x}_i$'s

$$\frac{\partial L}{\partial b} = 0 \Rightarrow -\sum_{i=1}^{N} \alpha_i y_i = 0 \Rightarrow$$

The weights α_i's will be equally distributed among positive- and negative-class training points

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow C - \alpha_i - \gamma_i = 0 \Rightarrow 0 \leq \alpha_i \leq C \Rightarrow$$

$\uparrow$ +ve $\uparrow$ +ve $\uparrow$ +ve

Weights α_i are restricted to being less than the hyperparameter C

By substituting the expression of $\underline{\theta} = \sum_{j=1}^{N} \alpha_j y_j \underline{x}_j$ in the Lagrangian L,

$$D(\underline{\xi}, \underline{\alpha}, \underline{\gamma}) = \frac{1}{2} \underbrace{\sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \underline{x}_i^T \underline{x}_j}_{\text{same}} - \underbrace{\sum_{i=1}^{N} y_i \alpha_i \sum_{j=1}^{N} y_j \alpha_j \underline{x}_j^T \underline{x}_i}_{}$$

$$+ C \sum_{i=1}^{N} \xi_i - b \underbrace{\sum_{i=1}^{N} y_i \alpha_i}_{\to 0} + \sum_{i=1}^{N} \alpha_i - \sum_{i=1}^{N} \alpha_i \xi_i - \sum_{i=1}^{N} \gamma_i \xi_i$$

$$= -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \underline{x}_i^T \underline{x}_j + \sum_{i=1}^{N} \alpha_i + \underbrace{\sum_{i=1}^{N} (C - \alpha_i - \gamma_i) \xi_i}_{\to 0}$$

$$= -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \underline{x}_i^T \underline{x}_j + \sum_{i=1}^{N} \alpha_i$$

The dual SVM (maximization = − minimization)

$$\min_{\underline{\alpha}} \quad \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \underline{x}_i^T \underline{x}_j - \sum_{i=1}^{N} \alpha_i$$

$$\text{s.t} \quad \sum_{i=1}^{N} y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C \quad \text{for } i = 1, \ldots, N$$

- We can rewrite the dual SVM optimization in vector form:

$$\min_{\underline{\alpha}} \quad \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \underline{x}_i^T \underline{x}_j - \sum_{i=1}^{N} \alpha_i$$

$$\text{s.t} \quad \sum_{i=1}^{N} y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C \quad \text{for } i = 1, \ldots, N$$

$$\min_{\underline{\alpha}} \quad \frac{1}{2} \underline{\alpha}^T \underline{\underline{K}} \, \underline{\alpha} - \underline{\alpha}^T \underline{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C \quad \text{for } \forall i$$

Here $K_{ij} = y_i \underline{x}_i^T \underline{x}_j y_j$ $\left(\text{and } \underline{\underline{K}} = \text{diag}(\underline{y}) \, \underline{\underline{X}} \, \underline{\underline{X}}^T \text{diag}(\underline{y})\right)$

- The dual optimization problem is a quadratic programming optimization with linear constraints. There is no closed-form solution

- Why formulate and solve the dual problem instead of primal?

Quadratic program
$$\min_{\underline{\alpha}} \quad \frac{1}{2} \underline{\alpha}^T \underline{\underline{K}} \, \underline{\alpha} - \underline{\alpha}^T \underline{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} \alpha_i y_i = 0, \quad \underbrace{0 \leq \alpha_i \leq C}_{\text{linear constraints}} \quad \text{for } \forall \, i$$

— The quadratic program with linear constraints is more easy to solve using any standard optimization software

— The optimization is over the number of training points N than the number of features d, making it attractive when $N \ll d$

— The dual optimization incorporates the term $\underline{\underline{X}} \underline{\underline{X}}^T$ which is simply the Gram matrix $\underline{\underline{K}}$ of kernel evaluations of all pairs of training pts One can apply the kernel trick to form this Gram matrix

Geometric Intuition of SVM

What do the dual values α_i mean?

- Stationarity condition: $\frac{\partial L}{\partial \xi_i} = 0 \implies C - \alpha_i - \gamma_i = 0 \quad$ — (a)

- Complementary slackness: $\alpha_i \cdot \left[y_i \left(\underline{\theta}^T \underline{x}_i + b \right) - 1 + \xi_i \right] = 0 \quad$ — (b)

$$\gamma_i \cdot \xi_i = 0 \quad \text{— (c)}$$

1) $\alpha_i = 0 \overset{(a)}{\implies} \gamma_i = C \overset{(c)}{\implies} \xi_i = 0 \rightarrow$ data point $\underline{x}_i$ is not given any slackness

$\implies$ data point $\underline{x}_i$ lies on or outside the margin

2) $\alpha_i \neq 0$

i) $\alpha_i = C \overset{(a)}{\implies} \gamma_i = 0 \overset{(c)}{\implies} \xi_i \nearrow \begin{array}{l} 0 \longrightarrow \underline{x}_i \text{ lies on margin} \\ \text{or} \\ \searrow \text{non-zero} \rightarrow \underline{x}_i \text{ violates the margin} \end{array}$

ii) $0 < \alpha_i < C \overset{(a)}{\implies} \gamma_i \neq 0 \overset{(c)}{\implies} \xi_i = 0 \overset{(b)}{\implies} y_i \left(\underline{\theta}^T \underline{x}_i + b \right) - 1 = 0$

$\implies y_i \left(\underline{\theta}^T \underline{x}_i + b \right) = 1 \implies \underline{x}_i$ lies on margin
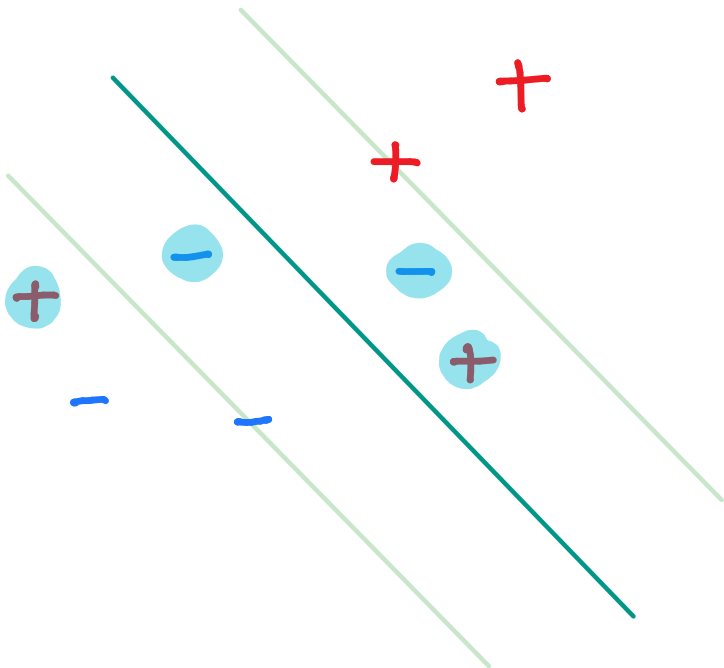
Geometric Intuition of SVM

$\alpha_i = 0 \Rightarrow y_i f(x_i) \geqslant 1 :$ on or outside the margin

$0 < \alpha_i < C \Rightarrow y_i f(x_i) = 1 :$ on the margin

$\alpha_i = C \Rightarrow y_i f(x_i) \leq 1 :$ on or <u>inside</u> the margin

$\alpha_i = 0 \Leftarrow y_i f(x_i) > 1 :$ outside the margin (correct side)

$\alpha_i = C \Leftarrow y_i f(x_i) < 1 :$ <u>inside</u> the margin (<u>wrong side</u>)

The training data-points for which the corresponding weights $\alpha_i \neq 0$. Such data-pts are called SUPPORT VECTORS.

Geometric Intuition of SVM

The training data-points for which the corresponding weights $\alpha_i \neq 0$. Such data-pts are called SUPPORT VECTORS.

- All training points that violate the boundary
  $\Rightarrow \alpha_i > 0$ and are support vectors

- All training points that strictly do not violate the boundary (meaning they do not lie on the boundary) have $\alpha_i = 0$ and are NOT support vectors

- For training points which lie exactly on the boundary, some may have $\alpha_i > 0$ and some may have $\alpha_i = 0$; only the points that are critical to determining the boundary have $\alpha_i > 0$ and are thus support vectors

- There are very few support vectors compared to the total number of training pts.
  $\Rightarrow$ dual vector $\underline{\alpha}$ is sparse
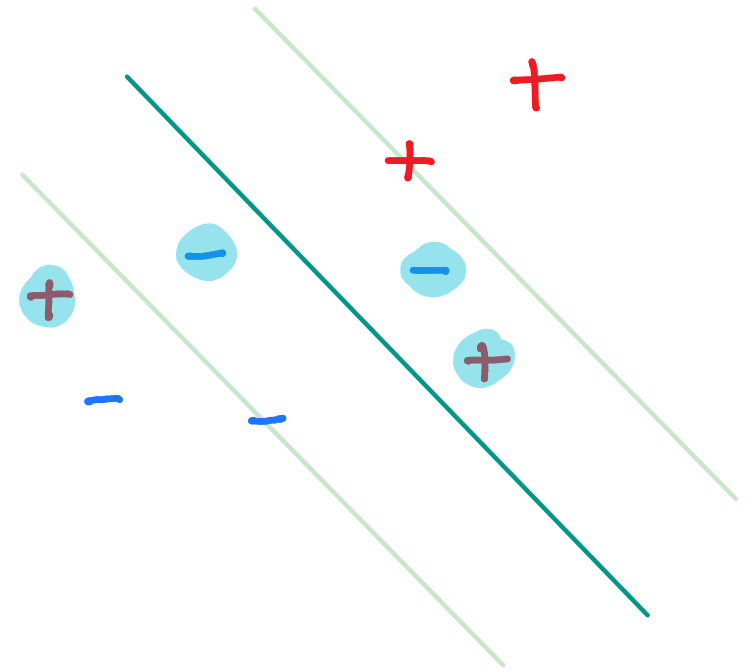
- Instead of working in the original input space $\underline{x}$, one may choose a feature space $\emptyset(\underline{x})$

- Using transformed features $\emptyset(\underline{x})$ instead of $\underline{x}$, the decision rule becomes

$$H := \left\{ \underline{x} : \underline{\theta}^T \emptyset(\underline{x}) + b = 0 \right\} \quad \left[\text{Decision boundary} \right]$$

- The dual kernel SVM optimization problem:

$$\min_{\underline{\alpha}} \quad \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i \, y_j \, \alpha_i \, \alpha_j \, \underbrace{\left(\emptyset(\underline{x}_i) \right)^T \emptyset(\underline{x}_j)}_{\text{kernel} \longrightarrow K(\underline{x}_i, \underline{x}_j)} - \sum_{i=1}^{N} \alpha_i$$

$$\text{s.t} \quad \sum_{i=1}^{N} y_i \, \alpha_i = 0, \quad 0 \leq \alpha_i \leq C \quad \text{for} \quad i = 1, \ldots, N$$

- In vectorized form:

$$\text{minimize} \quad \frac{1}{2} \underline{\alpha}^T \underline{\underline{K}}(\underline{\underline{X}}, \underline{\underline{X}}) \underline{\alpha} - \underline{\alpha}^T \underline{1}$$

Dual parameter $\longrightarrow \underline{\alpha}$

$$\text{s.t.} \quad \sum_{i=1}^{N} y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C \quad \text{for } \forall i$$

No closed-form solution; you need an optimizer to find solution numerically

- Prediction: $\hat{y}(\underline{x}_*) = \text{sign}\left(\underline{\hat{\alpha}}^T \underline{\underline{K}}(\underline{\underline{X}}, \underline{x}_*) \right)$

- The interesting point in SVM is that the dual parameter $\underline{\alpha}$ turns out to be sparse

- Prediction $\hat{y}(\underline{x}_*)$ depends only on a subset of training points. Note, however, all training points are needed during training

Support Vector Classification

Training

Data: Training data $\mathcal{T} = \{\underline{x}_i, y_i\}_{i=1}^{N}$, choice of kernel

Result: Learned dual parameters $\hat{\underline{\alpha}}$

Procedure: Compute $\hat{\underline{\alpha}}$ by numerically minimizing

$$\underset{\underline{\alpha}}{\text{minimize}} \quad \frac{1}{2} \underline{\alpha}^T \underline{\underline{K}}(\underline{\underline{x}}, \underline{\underline{x}}) \underline{\alpha} - \underline{\alpha}^T \underline{1}$$

$$\text{subject to} \quad \sum_{i=1}^{N} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C$$

Prediction

Data: Learned parameters $\hat{\underline{\alpha}}$ and test input $\underline{x}_*$

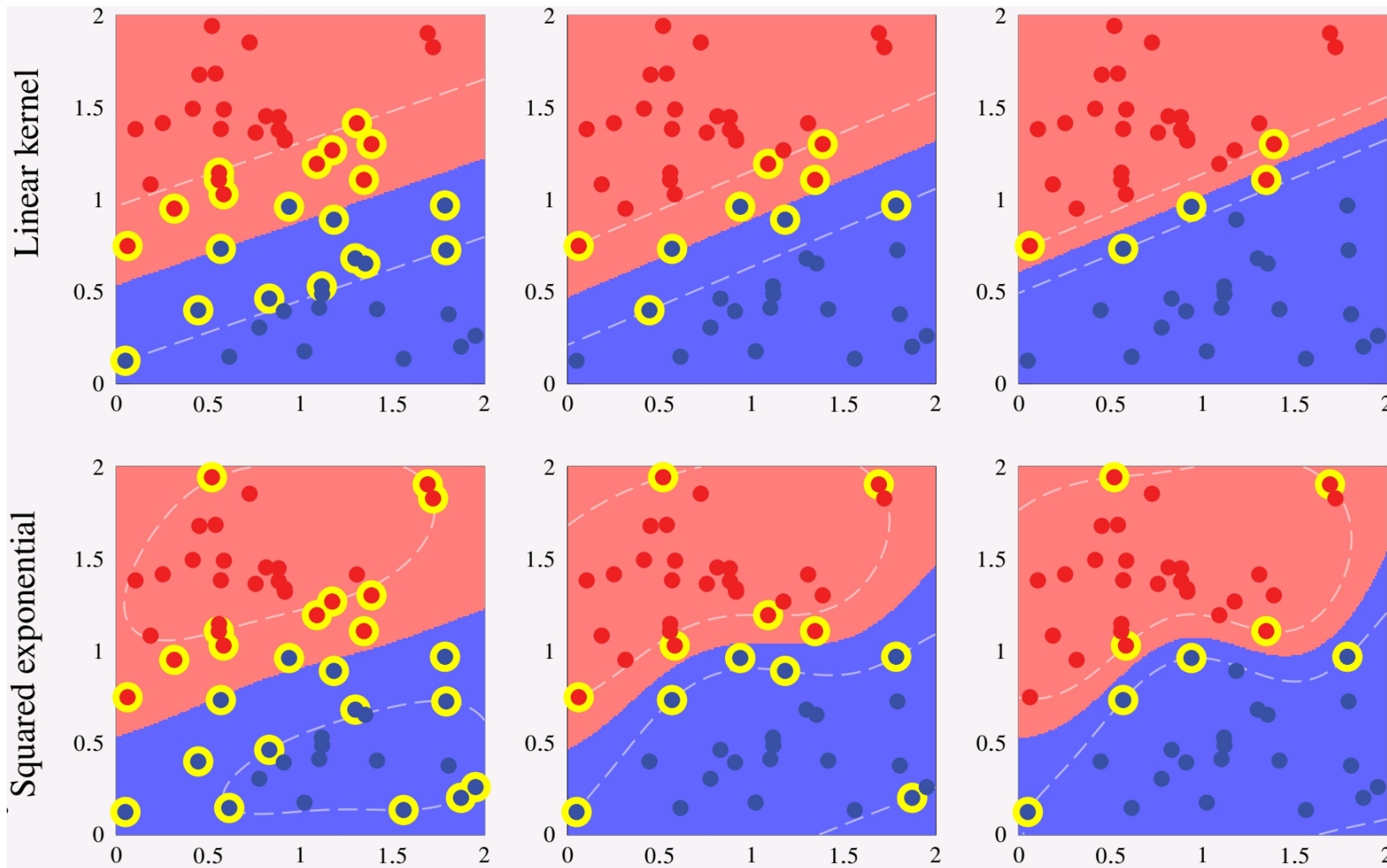Result: Prediction $y(\underline{x}_*) = \text{sign}\left(\hat{\underline{\alpha}}^T \underline{\underline{K}}(\underline{x}, \underline{x}_*)\right)$

Example of binary classification with SVC

- Linear kernel
- Squared exponential kernel } Used kernels

C = 0.01   C = 0.1   C = 1

Linear kernel

Squared exponential

As you increase C, we allow for lesser ε_i, which means a narrower street, and fewer support vectors