Lecture 12:

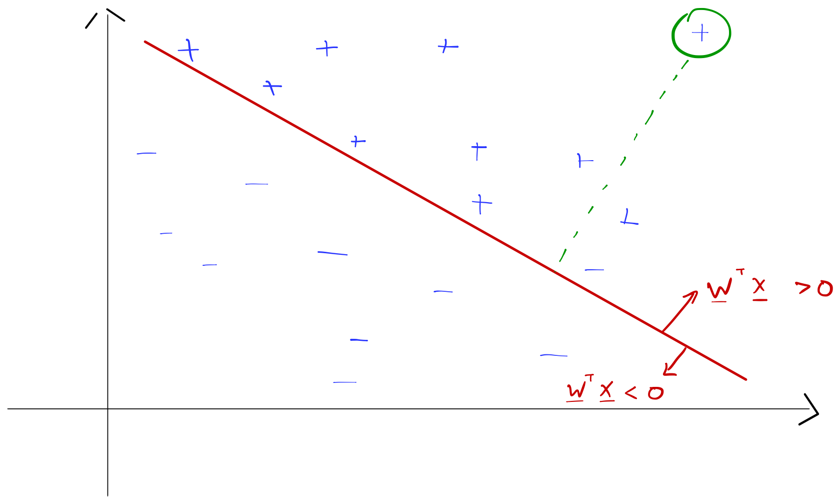
Support Vector Machines

Support Vector Machines

Assume Labels $y_i \in \{+1, -1\}$

Then $\|y - X\underline{w}\|_2^2 = \sum_{i=1}^{n} (y_i - x_i^T \underline{w})^2 = \sum_{i=1}^{n} (1 - \underbrace{y_i \underline{x}_i^T \underline{w}})^2$

if this is $\neq 1$, then we incur loss even if point is correctly classified



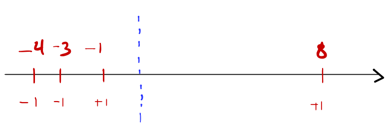
Ex. predict whether someone plays basketball based on height.

$y_i = +1 \longrightarrow$ plays bball

$y_i = -1 \longrightarrow$ doesn't play bball

4 training samples: x_i : $70'$, $71'$, $73'$, $82'$ ← difference from mean $74'$: -4, -3, -1, 8

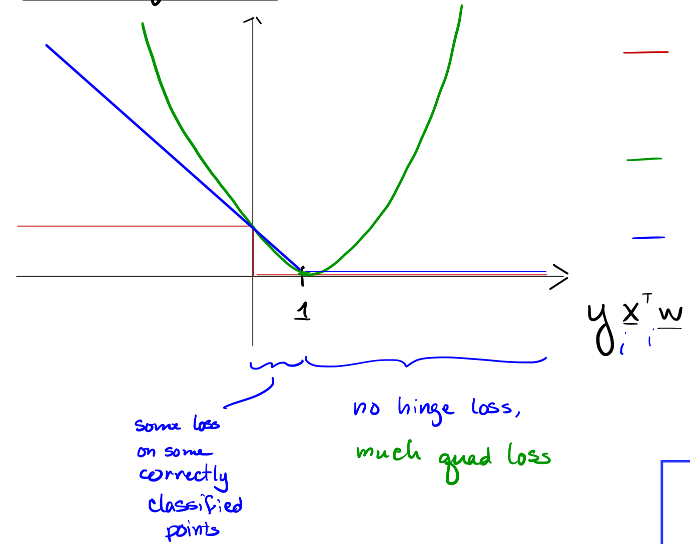
y_i : -1, -1, $+1$, $+1$



$\hat{\underline{w}}_{LS} = (\underline{X}^T\underline{X})^{-1}\underline{X}^T y = 0.15 \Longrightarrow \hat{y} = +1$ if (height − mean) > 0.15

LS decision boundary misclassifies a point even though perfect classifier exists

Loss functions



— $0/1$ loss, $= \mathbb{I}_{\{y \neq \text{sign}(x^T w)\}} = \mathbb{I}_{\{y x^T \underline{w} < 0\}}$ = "ideal loss"

— quadratic loss $= (1 - y\underline{x}^T w)^2$

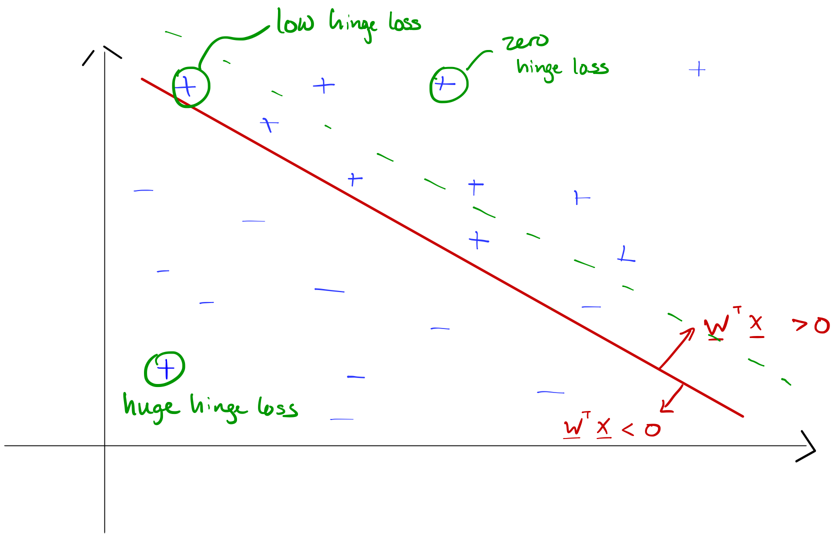
— hinge loss $= (1 - y\underline{x}^T \underline{w})_+$

if $y = +1$, $x^T w > 0$
then we accurately predicted label
if $y = -1$, $x^T w < 0$, acc label

$$(a)_+ = \begin{cases} a & \text{if } a > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbb{I}_{\{a\}} = \begin{cases} 1 & a \text{ true} \\ 0 & a \text{ false} \end{cases}$$

Hinge loss mimics the ideal loss but is convex and easy to minimize



$$\ell(\underline{w}) = \sum_{i=1}^{n} (1 - y_i \underline{x}_i^T \underline{w})_+$$

$$\nabla_{\underline{w}} \ell = \sum_{i=1}^{n} \mathbb{I}_{\{y_i \underline{x}_i^T \underline{w} < 1\}} (-y_i \underline{x}_i)$$

technically called "subgradient" because $\ell(\underline{w})$ not differentiable

if $yx^T w > 0$, then correct label prediction
$yx^T w < 0$, then wrong label prediction

If we minimize $\sum_{i=1}^{n} (1 - y_i \underline{x}_i^T \underline{w})_+ + \lambda \|w\|_2^2$

or the kernel version $\sum_{i=1}^{n} (1 - y_i\, \phi(x_i)^T \underline{w})_+ + \lambda \|w\|_2^2$

this is called a support vector machine

Let $\hat{w}_{svm} = \underset{w}{\text{argmin}} \sum_{i=1}^{n} (1 - y_i \underline{x}_i^\top w)_+ + \lambda \|w\|_2^2$ or $\hat{w}_{svm} = \underset{w}{\text{argmin}} \sum_{i=1}^{n} (1 - y_i \phi(\underline{x}_i)^\top w)_+ + \lambda \|w\|_2^2$

Just like with kernel ridge regression, it is possible to show $\hat{w}_{svm} = X^\top \underline{\alpha}$ for some $\underline{\alpha}$ (different from least squares $\underline{\alpha}$)

or $\hat{w}_{svm} = \underline{\Phi}^\top \underline{\alpha}$

To see this:

$$(a)_+ = \begin{cases} a & \text{if } a \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Imagine $\underline{w} = X^\top \underline{\alpha} + \underline{w}^\perp = \sum_{j=1}^{n} \alpha_j \underline{x}_j + \underline{w}^\perp$ ($\underline{w}^\perp$ some vector orthogonal to the $\underline{x}_i$'s)

then
$$\min_{\alpha, w^\perp} \sum_{i=1}^{n} \left(1 - y_i \underline{x}_i^\top \left(\sum_{j=1}^{n} \alpha_j x_j + w^\perp\right)\right)_+ + \lambda \left\|\left(\sum_{j=1}^{n} \alpha_j x_j + w^\perp\right)\right\|_2^2$$

$$= \min_{\underline{\alpha}, \underline{w}^\perp} \sum_i \left(1 - y_i \left[\sum_{j=1}^{n} \alpha_j \langle \underline{x}_i, \underline{x}_j \rangle + \underbrace{\underline{x}_i^\top \underline{w}^\perp}_{\text{always zero!}}\right]\right)_+ + \lambda \left[\left\|\sum_{j=1}^{n} \alpha_j \underline{x}_j\right\|_2^2 + \|w^\perp\|_2^2\right]$$

$$= \min_{\underline{\alpha}, \underline{w}^\perp} \sum_i \left(1 - y_i \left[\sum_{j=1}^{n} \alpha_j \langle \underline{x}_i, \underline{x}_j \rangle\right]\right)_+ + \lambda \left[\left\|\sum_{j=1}^{n} \alpha_j \underline{x}_j\right\|_2^2 + \underbrace{\|w^\perp\|_2^2}_{\text{this must be 0 at optimum!}}\right]$$

$$\underline{\hat{\alpha}} = \underset{\underline{\alpha}}{\text{argmin}} \sum_i \left(1 - y_i \left[\sum_{j=1}^{n} \alpha_j \langle \underline{x}_i, \underline{x}_j \rangle\right]\right)_+ + \lambda \underbrace{\left\|\sum_{j=1}^{n} \alpha_j \underline{x}_j\right\|_2^2}_{= \sum_i \sum_j \alpha_i \alpha_j \langle \underline{x}_i, \underline{x}_j \rangle}$$

Alternatively, we can apply the "kernel trick" and replace inner products with kernels:

$$\underline{\hat{\alpha}} = \underset{\underline{\alpha}}{\text{argmin}} \sum_{i=1}^{n} \left(1 - y_i \sum_j \alpha_j K(\underline{x}_i, \underline{x}_j)\right)_+ + \lambda \sum_i \sum_j \alpha_i \alpha_j K(\underline{x}_i, \underline{x}_j)$$

There is no closed form solution to this optimization problem.

$\Rightarrow$ need to use gradient descent or other numerical optimization methods.

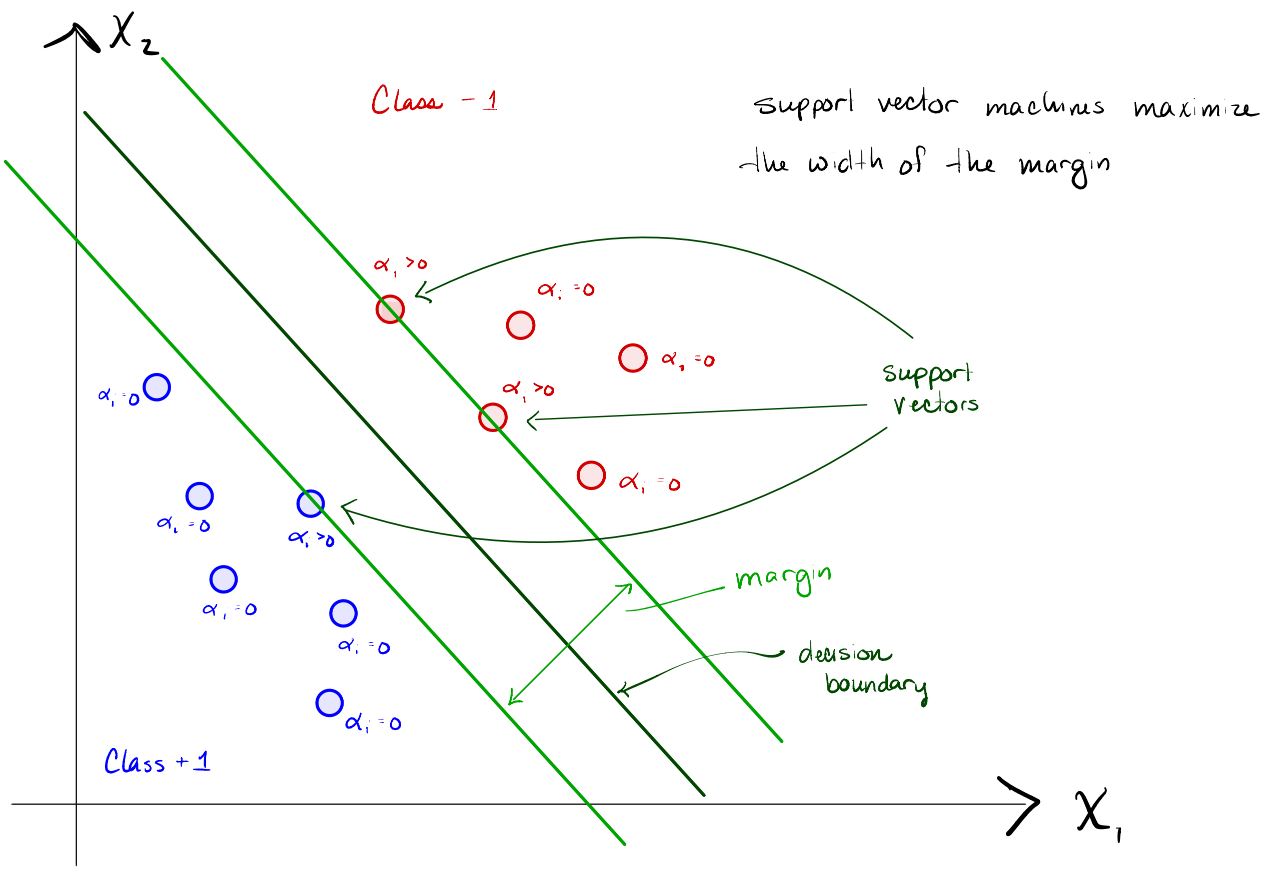
Why is this called a "support vector machine"?

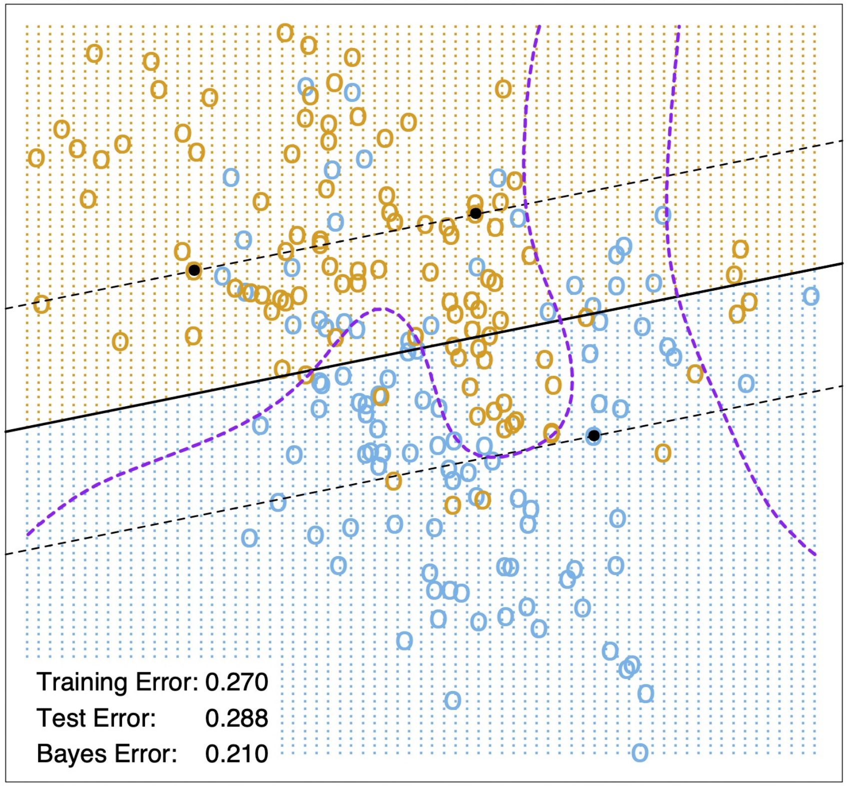
Typically, $\underline{\hat{\alpha}}$ is sparse — most $\hat{\alpha}_j = 0$

Recall $\underline{\hat{w}} = \sum_{j=1}^{n} \hat{\alpha}_j \, \phi(\underline{x}_j)$

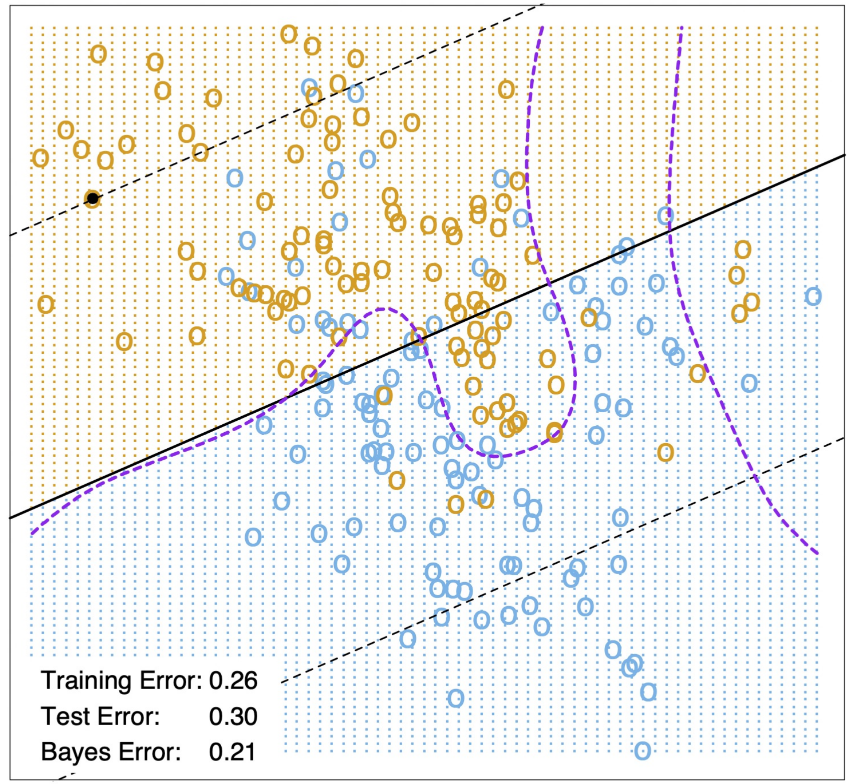
$\Rightarrow$ $\underline{\hat{w}}$ is a linear combination of only a few training samples (in feature space)

those $\underline{x}_i$'s are called support vectors.





$\lambda = 10^{-4}$



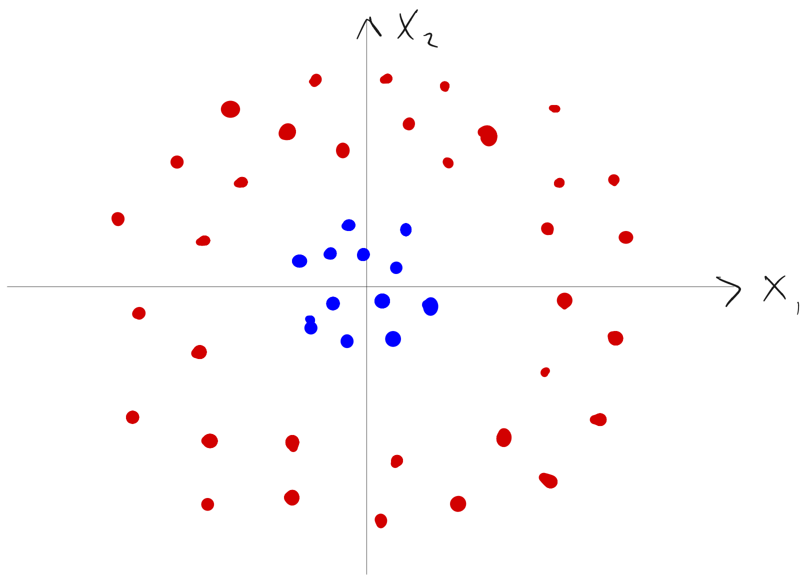
$\lambda = 100$

- - - - - - - ideal ("Bayes") decision boundary — gives smallest test errors on average

- - - - - - - points $\underline{x}$ where $\hat{y} = \underline{x}^T \hat{\underline{w}}_{svm} = \pm 1$ $\longleftarrow$

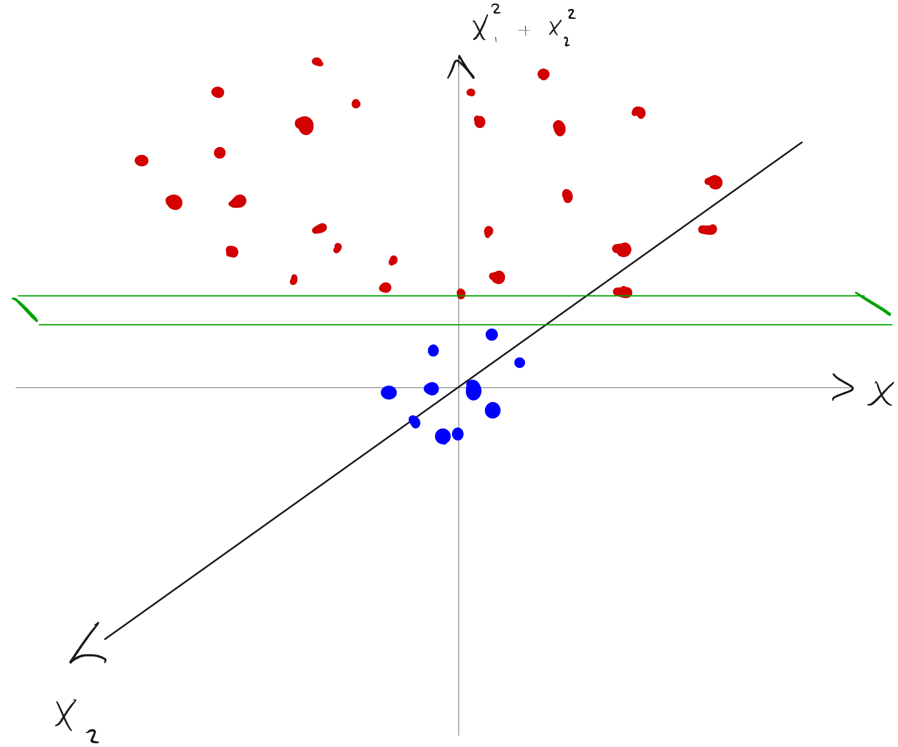
——— SVM decision boundary

How do Kernels help?



no good linear classifier exists

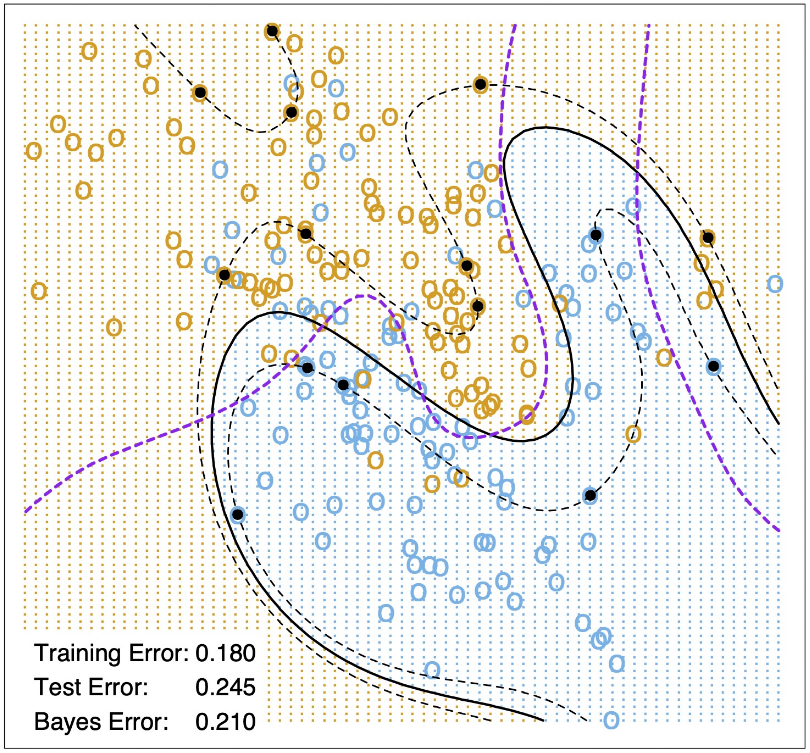
$$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix}$$



in high-dimensional feature space, a good linear seperating hyperplane exists.

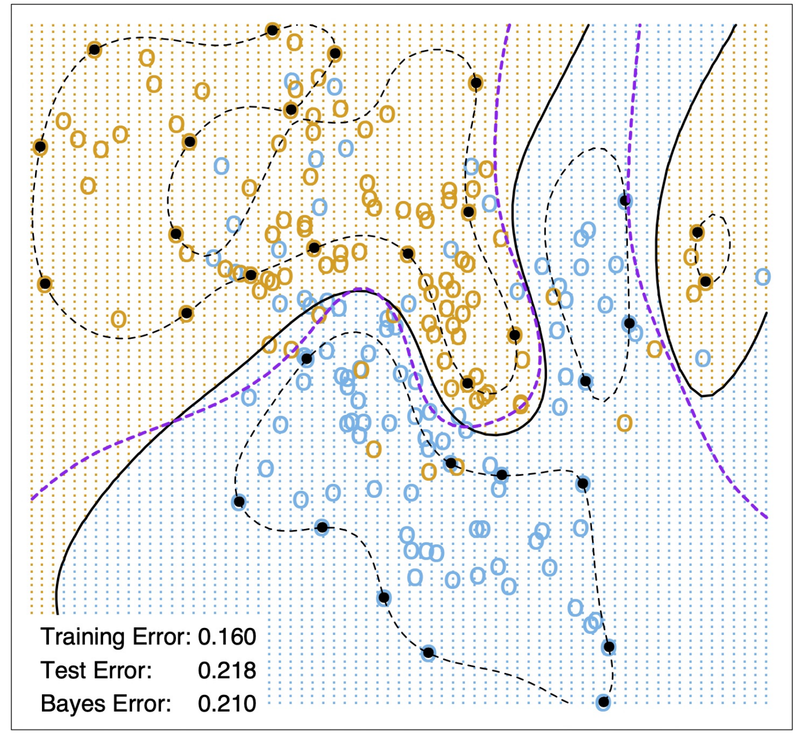
$$\phi(x_i) = \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i1}^2 + x_{i2}^2 \end{bmatrix}$$

SVM - Degree-4 Polynomial in Feature Space



(Gaussian)

SVM - Radial Kernel in Feature Space



$$k(\underline{x}_i, \underline{x}_j) = \exp\left\{ \frac{-\|x_i - x_j\|_2^2}{2\sigma^2} \right\}$$

- - - - - - ideal ("Bayes") decision boundary — gives smallest test errors on average

- - - - - - points $\underline{x}$ where $\hat{y} = \underline{x}^T \hat{\underline{w}}_{svm} = \pm 1$

——— SVM decision boundary