SVMs and Kernels

UNIVERSITY of WASHINGTON

Two different approaches to regression/classification

- Assume something about P(x,y)
- Find f which maximizes likelihood of training data | assumption
  - Often reformulated as minimizing loss

Versus

- Pick a loss function
- Pick a set of hypotheses H
- Pick f from H which minimizes loss on training data

Our description of logistic regression was the former

- **Learn**: f:**X —>**Y
  - **X** – features
  - **Y – target classes**

$$Y \in \{-1, 1\}$$

- **Expected loss of f:**

$$\mathbb{E}_{XY}[\mathbf{1}\{f(X) \neq Y\}] = \mathbb{E}_X[\mathbb{E}_{Y|X}[\mathbf{1}\{f(x) \neq Y\}|X = x]]$$

$$\mathbb{E}_{Y|X}[\mathbf{1}\{f(x) \neq Y\}|X = x] = 1 - P(Y = f(x)|X = x)$$

- **Bayes optimal classifier:**

$$f(x) = \arg\max_y \mathbb{P}(Y = y|X = x)$$

- **Model of logistic regression:**

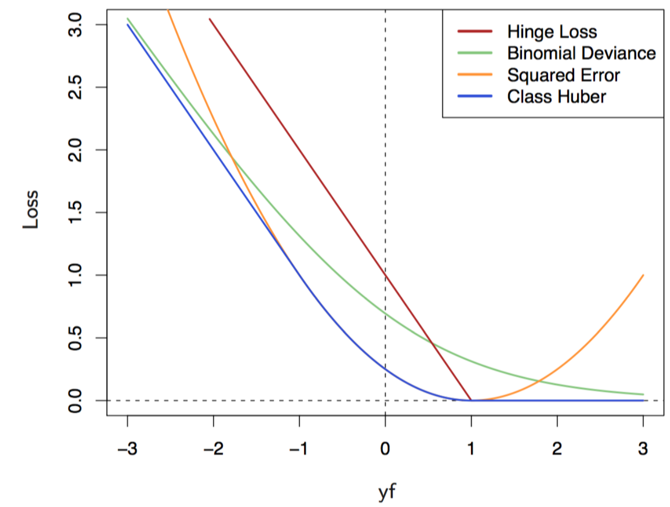
$$P(Y = y|x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

- **Loss function:**

$$\ell(f(x), y) = \mathbf{1}\{f(x) \neq y\}$$

What if the model is wrong? What other ways can we pick linear decision rules?

Loss Functions



$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

- Loss functions: $\sum_{i=1}^n \ell_i(w)$

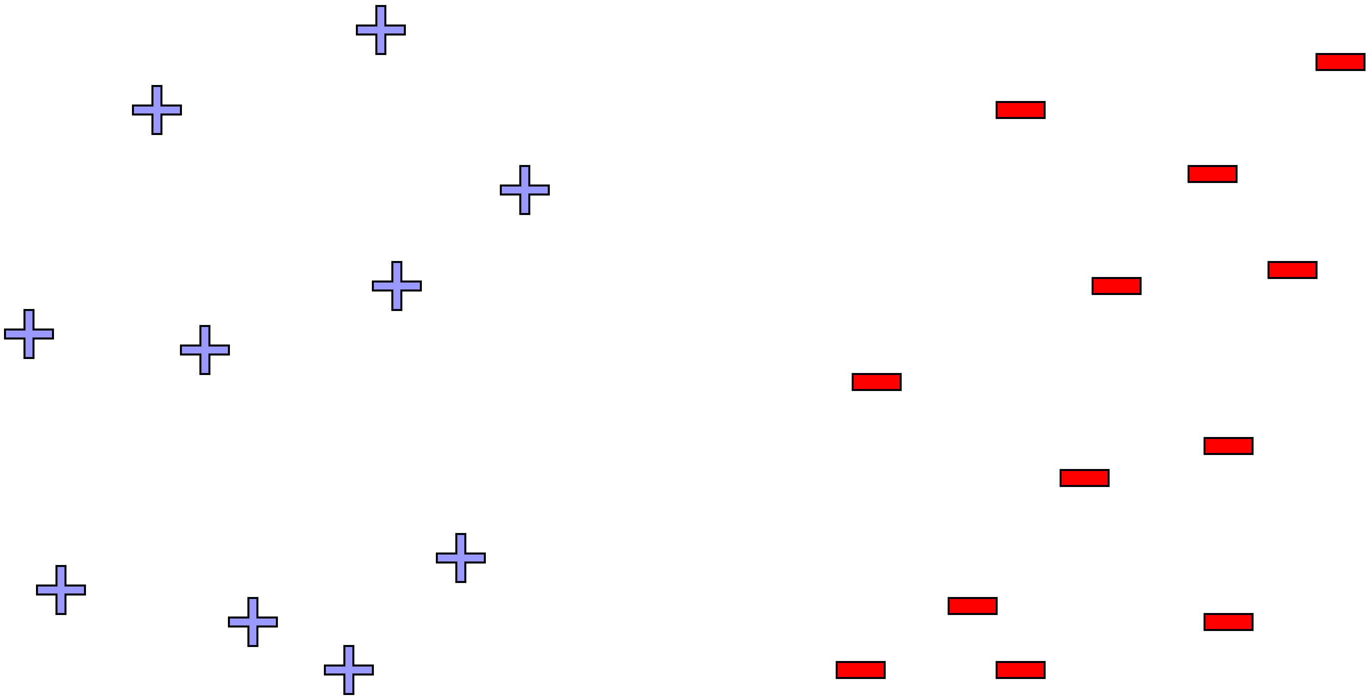
Squared error Loss: $\ell_i(w) = (y_i - x_i^T w)^2$

Logistic Loss: $\ell_i(w) = \log(1 + \exp(-y_i\, x_i^T w))$

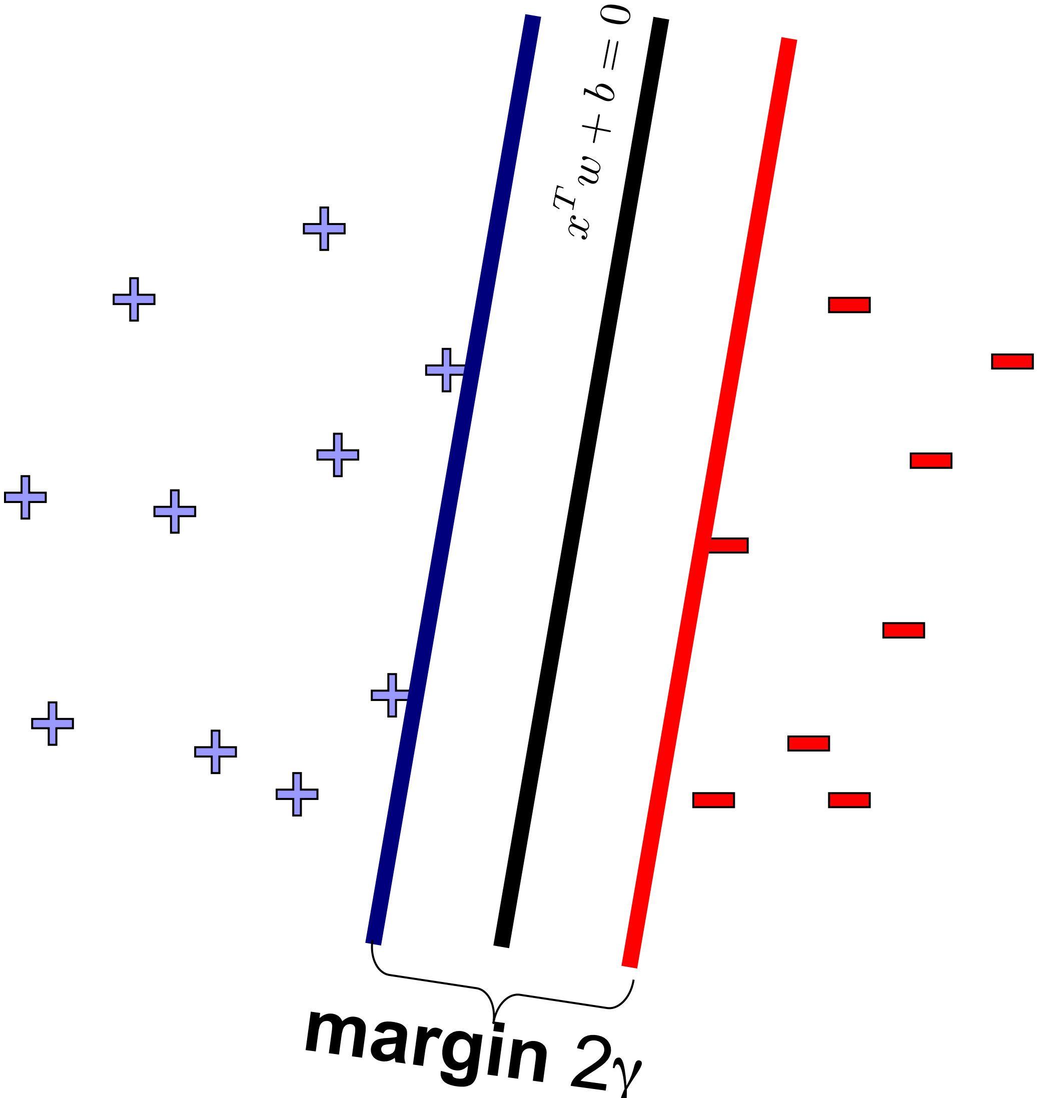
0/1 loss: $\ell_i(w) = \mathbb{I}[\text{sign}(y_i) \neq \text{sign}(x_i^T w)]$

Hinge Loss: $\ell_i(w) = \max\{0, 1 - y_i x_i^T w\}$

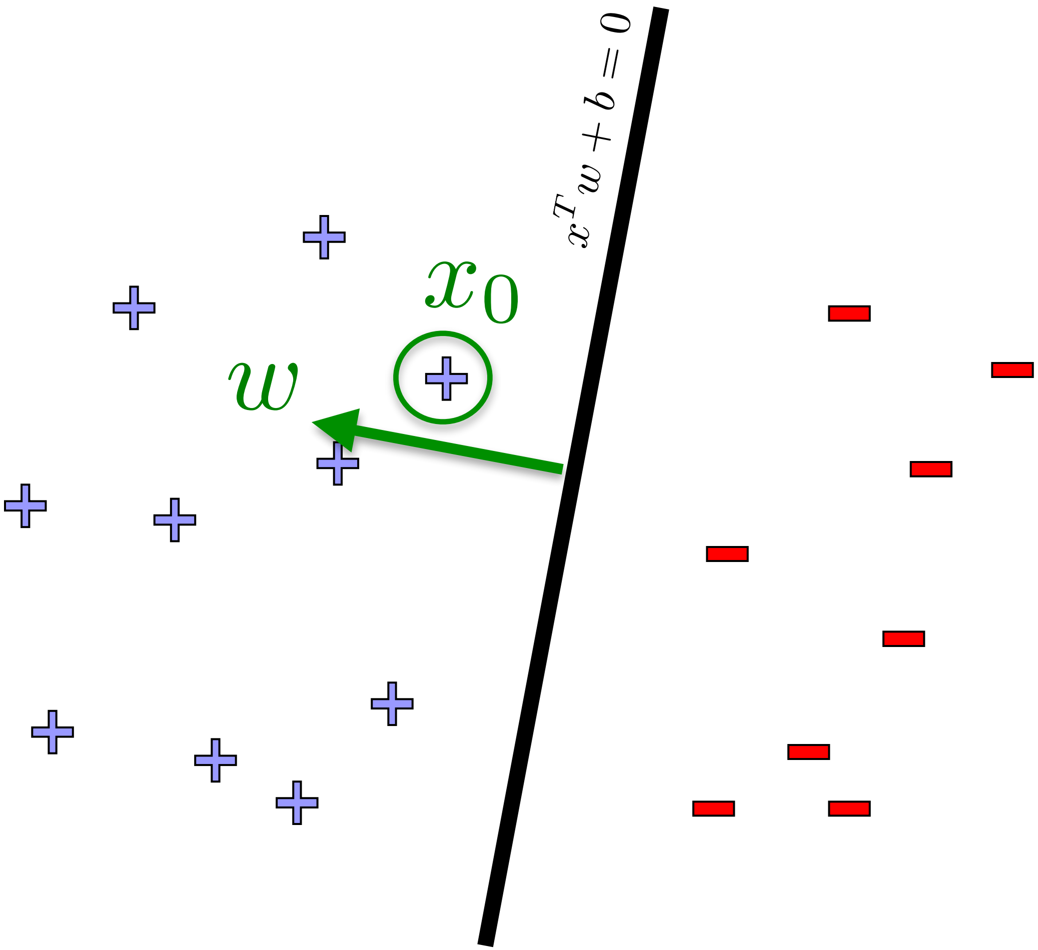
Linear classifiers – Which line is better?



Pick the one with the largest margin!

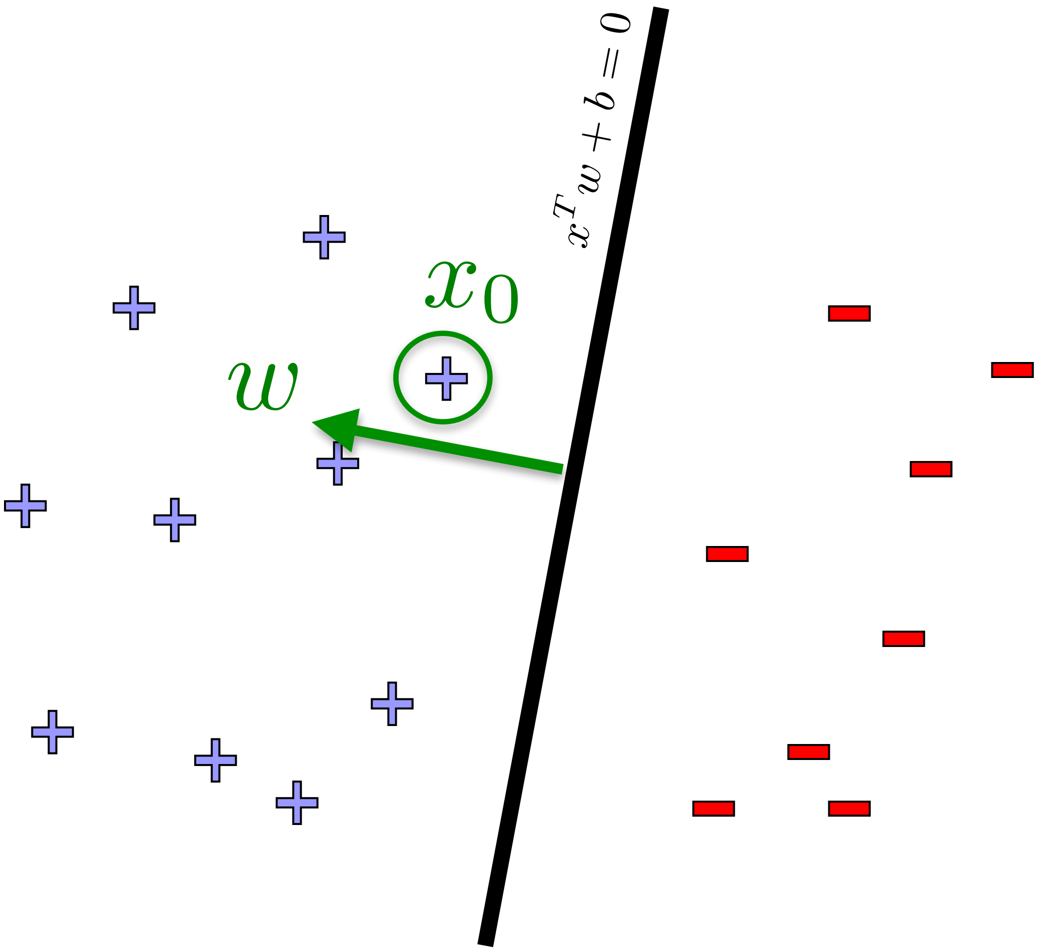


Pick the one with the largest margin!



Distance from x_0 to hyperplane defined by $x^T w + b = 0$?

Pick the one with the largest margin!



Distance from x_0 to hyperplane defined by $x^T w + b = 0$?

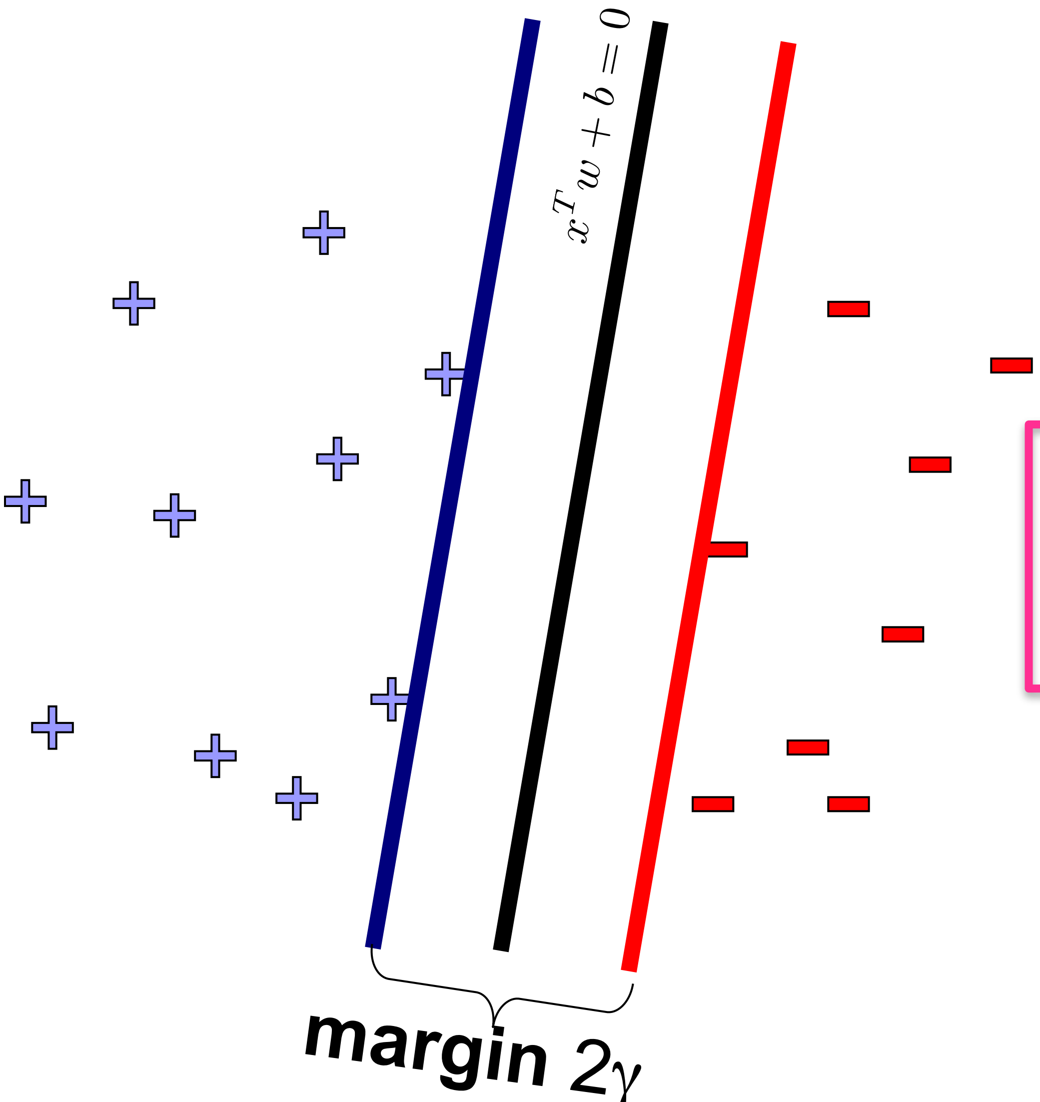
If $\widetilde{x}_0$ is the projection of x_0 onto the hyperplane then

$$||x_0 - \widetilde{x}_0||_2 = |(x_0^T - \widetilde{x}_0)^T \frac{w}{||w||_2}|$$

$$= \frac{1}{||w||_2} |x_0^T w - \widetilde{x}_0^T w|$$

$$= \frac{1}{||w||_2} |x_0^T w + b|$$

Pick the one with the largest margin!



Distance of x_0 from hyperplane $x^T w + b$:

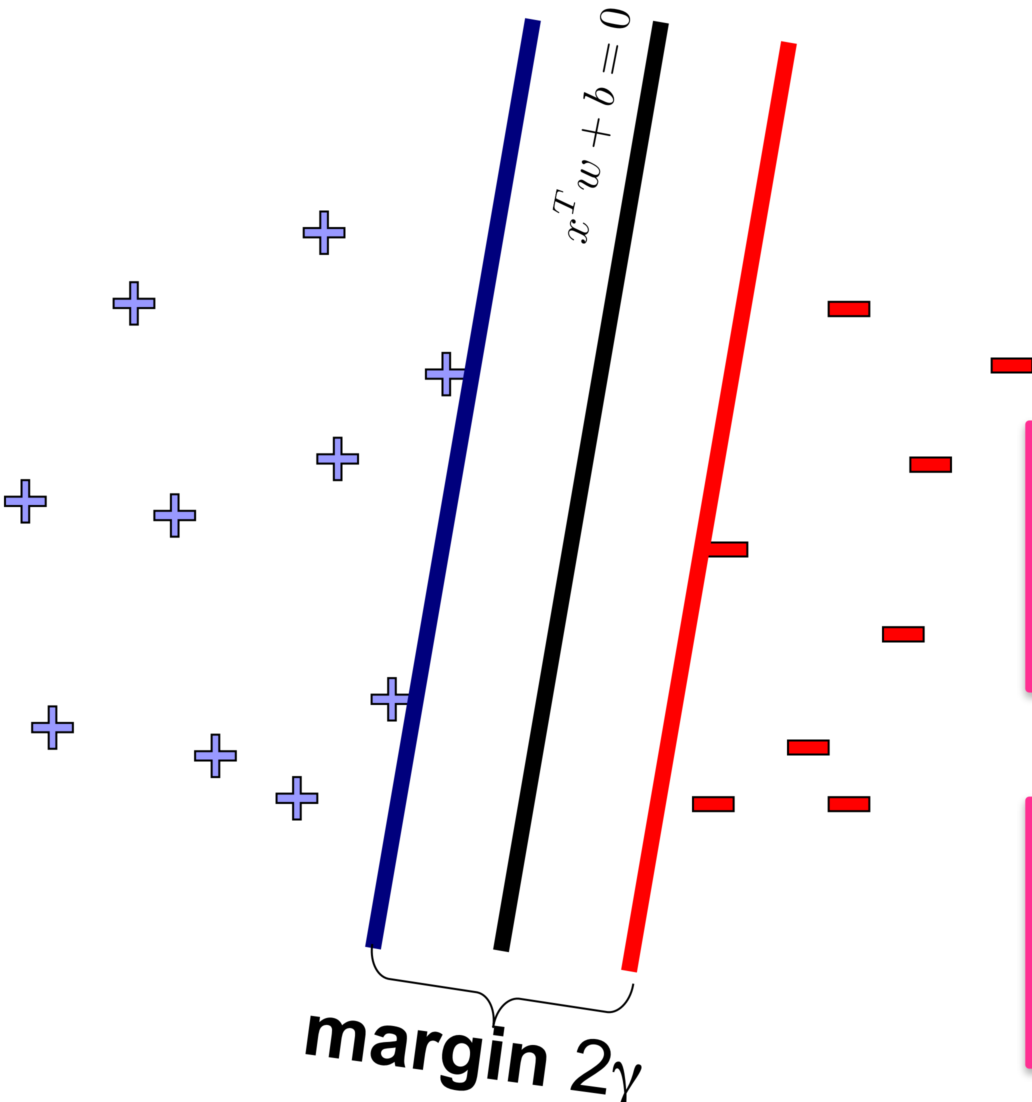
$$\frac{1}{||w||_2}(x_0^T w + b)$$

Optimal Hyperplane

$$\max_{w,b} \gamma$$

$$\text{subject to } \frac{1}{||w||_2} y_i(x_i^T w + b) \geq \gamma \quad \forall i$$

Pick the one with the largest margin!



Distance of x_0 from hyperplane $x^T w + b$:

$$\frac{1}{||w||_2}(x_0^T w + b)$$

Optimal Hyperplane

$$\max_{w,b} \gamma$$

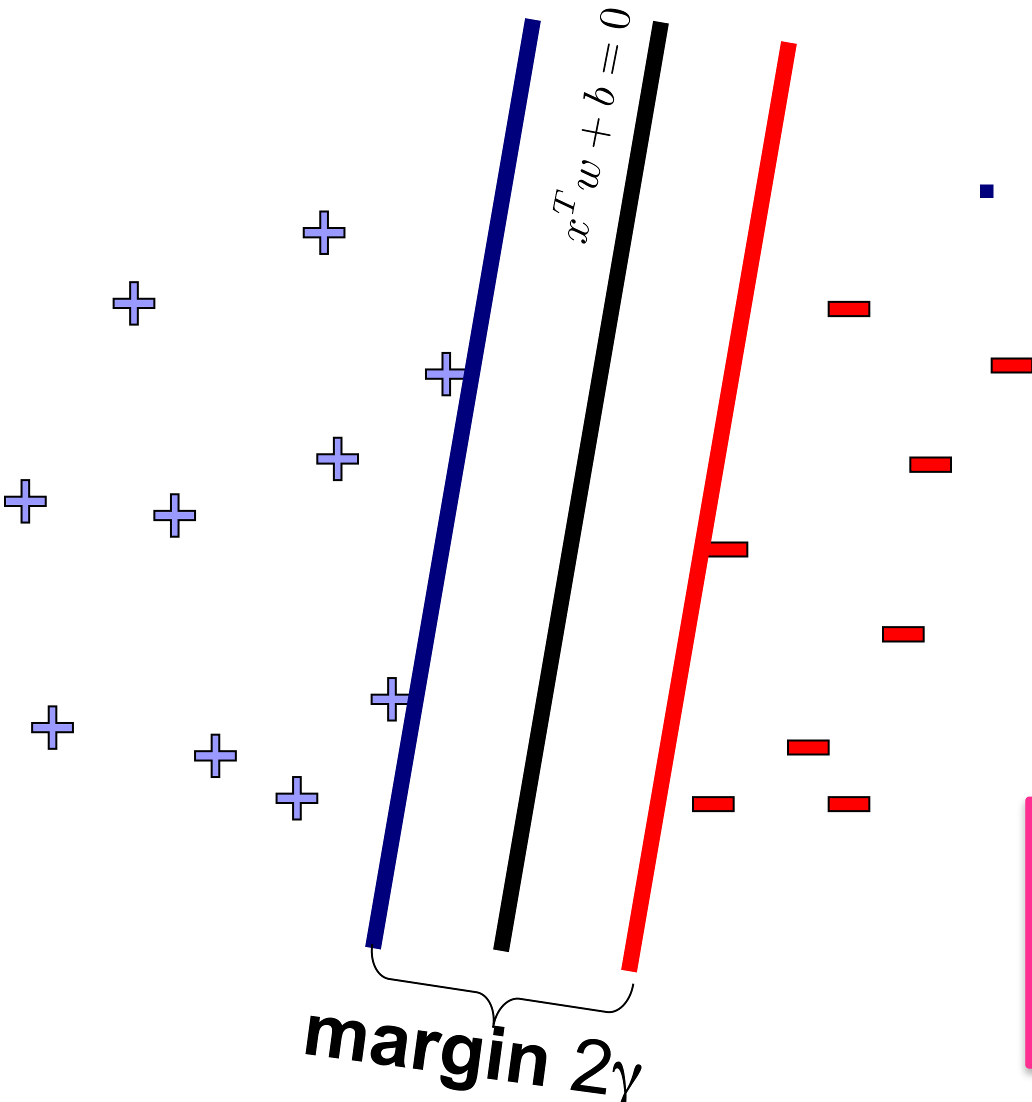
$$\text{subject to } \frac{1}{||w||_2} y_i(x_i^T w + b) \geq \gamma \quad \forall i$$

Optimal Hyperplane (reparameterized)

$$\min_{w,b} ||w||_2^2$$

$$\text{subject to } y_i(x_i^T w + b) \geq 1 \quad \forall i$$

Pick the one with the largest margin!



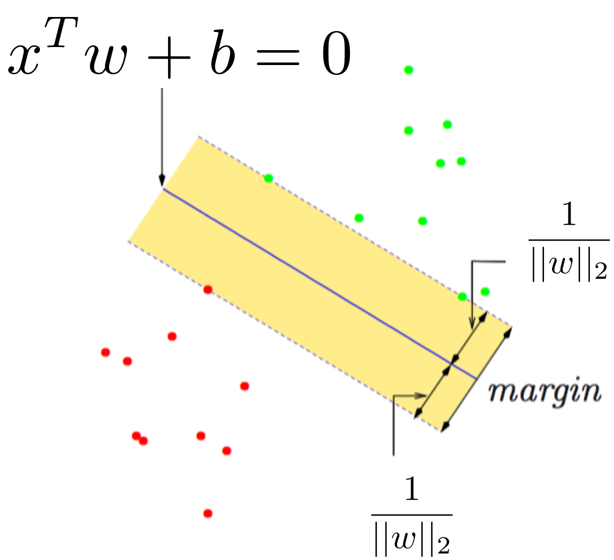
- Solve efficiently by many methods, e.g.,
  - quadratic programming (QP)
    - Well-studied solution algorithms
  - Stochastic gradient descent
  - Coordinate descent (in the dual)

Optimal Hyperplane (reparameterized)

$$\min_{w,b} ||w||_2^2$$

$$\text{subject to } y_i(x_i^T w + b) \geq 1 \quad \forall i$$

What if the data is not linearly separable?



If data is linearly separable

$$\min_{w,b} ||w||_2^2$$

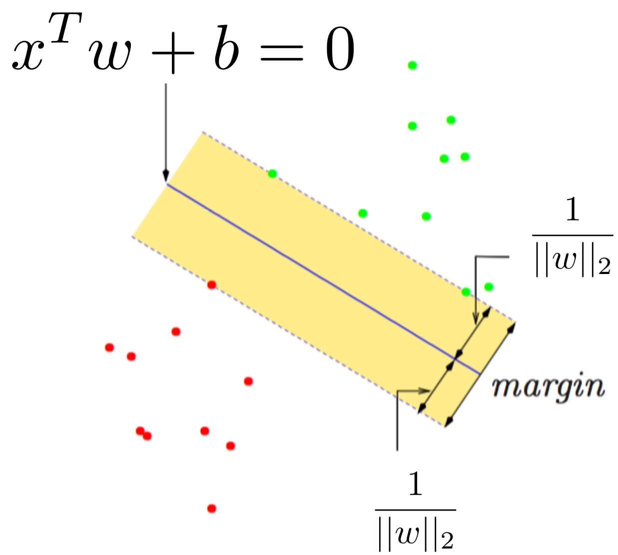
$$y_i(x_i^T w + b) \geq 1 \quad \forall i$$

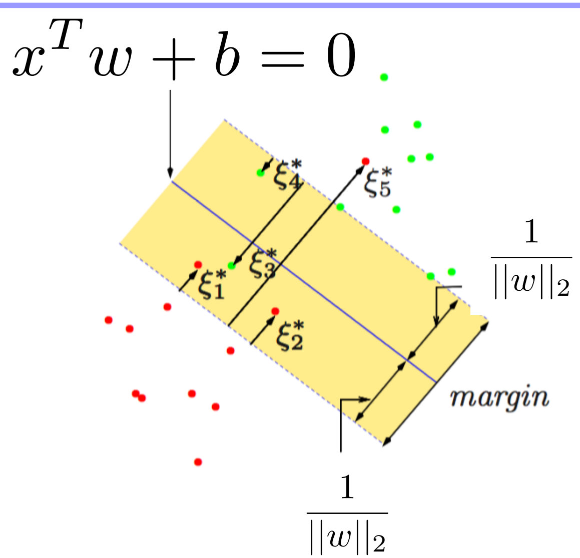
If data is not linearly separable, some points don't satisfy margin constraint:

Two options:

1. Introduce slack to this optimization problem
2. Lift to higher dimensional space

What if the data is not linearly separable?





If data is linearly separable:

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 \quad \forall i$$

If data is not linearly separable,

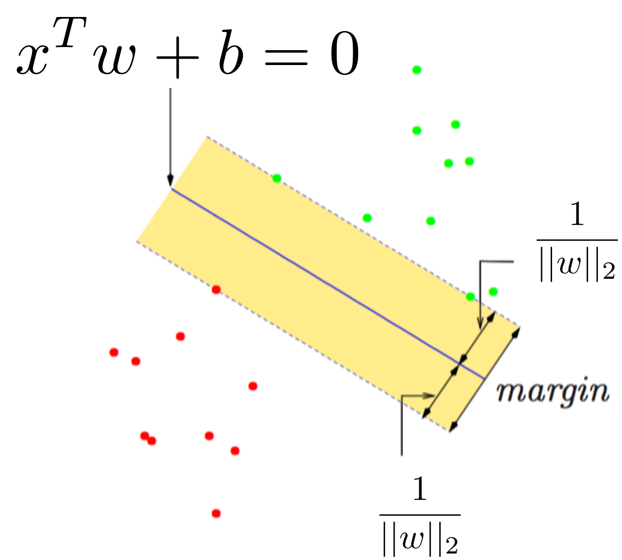
some points don't satisfy margin constraint:

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0, \sum_{j=1}^{n} \xi_j \leq \nu$$

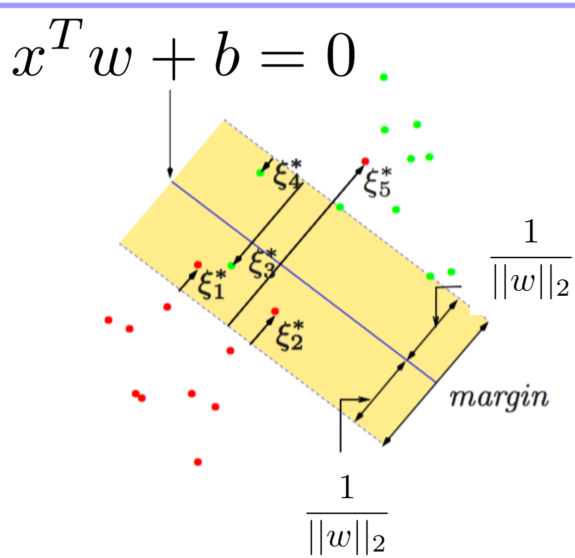
What if the data is not linearly separable?



If data is linearly separable:

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 \quad \forall i$$



If data is not linearly separable,

some points don't satisfy margin constraint:

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0, \sum_{j=1}^{n} \xi_j \leq \nu$$

- What are "support vectors?"

SVM as penalization method

- Original quadratic program with linear constraints:

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0, \sum_{j=1}^n \xi_j \leq \nu$$

SVM as penalization method

- Original quadratic program with linear constraints:

$$\min_{w,b} ||w||_2^2$$
$$y_i(x_i^T w + b) \geq 1 - \xi_i \quad \forall i$$
$$\xi_i \geq 0, \sum_{j=1}^n \xi_j \leq \nu$$

- Using same constrained convex optimization trick as for lasso:

For any $\nu \geq 0$ there exists a $\lambda \geq 0$ such that the solution the following solution is equivalent:

$$\sum_{i=1}^n \max\{0, 1 - y_i(b + x_i^T w)\} + \lambda ||w||_2^2$$

SVMs: optimizing what?

SVM objective:

$$\sum_{i=1}^{n} \max\{0, 1 - y_i(b + x_i^T w)\} + \lambda ||w||_2^2 \quad = \sum_{i=1}^{n} \ell_i(w, b)$$

$$\nabla_w \ell_i(w, b) = \begin{cases} -x_i y_i + \frac{2\lambda}{n} w & \text{if } y_i(b + x_i^T w) < 1 \\ \frac{2\lambda}{n} & \text{otherwise} \end{cases}$$

$$\nabla_b \ell_i(w, b) = \begin{cases} -y_i & \text{if } y_i(b + x_i^T w) < 1 \\ 0 & \text{otherwise} \end{cases}$$

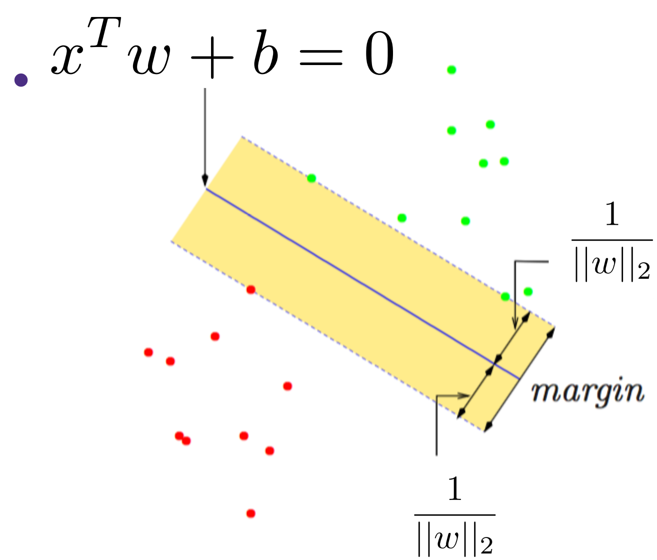
SVMs: optimizing what?

SVM objective:

$$\sum_{i=1}^{n} \max\{0, 1 - y_i(b + x_i^T w)\} + \lambda ||w||_2^2 \quad = \sum_{i=1}^{n} \ell_i(w, b)$$

Note: the minimizer of this can be written in terms of very few of the training points. These points are known as support vectors.

What if the data is not linearly separable?



Ie, some points don't satisfy margin

$$\min_{w,b} ||w||_2^2$$

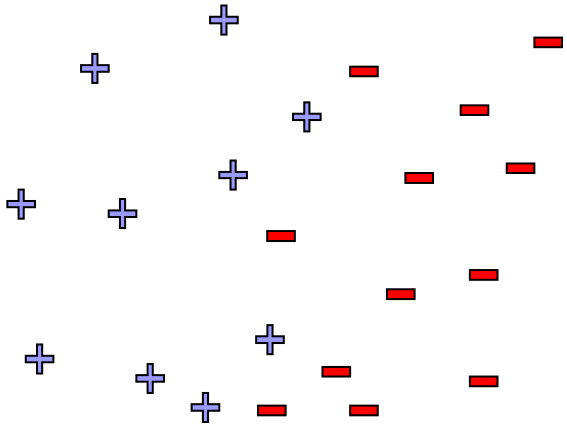
$$y_i(x_i^T w + b) \geq 1 \quad \forall i$$

Two options:

1. Introduce slack to this optimization problem
2. **Lift to higher dimensional space**

What if the data is not linearly separable?

Use features of features of features…



$$\phi(x) : \mathbb{R}^d \to \mathbb{R}^p$$

Feature space can get really large really quickly!

Dot-product of polynomials

$$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = \text{polynomials of degree exactly d}$$

$$d = 1 : \phi(u) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad \langle \phi(u), \phi(v) \rangle = u_1 v_1 + u_2 v_2$$

Dot-product of polynomials

$$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = \text{polynomials of degree exactly d}$$

$$d = 1 : \phi(u) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad \langle \phi(u), \phi(v) \rangle = u_1 v_1 + u_2 v_2$$

$$d = 2 : \phi(u) = \begin{bmatrix} u_1^2 \\ u_2^2 \\ u_1 u_2 \\ u_2 u_1 \end{bmatrix} \qquad \langle \phi(u), \phi(v) \rangle = u_1^2 v_1^2 + u_2^2 v_2^2 + 2 u_1 u_2 v_1 v_2$$

Dot-product of polynomials

$$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = \text{polynomials of degree exactly d}$$

$$d = 1 : \phi(u) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad \langle \phi(u), \phi(v) \rangle = u_1 v_1 + u_2 v_2$$

$$d = 2 : \phi(u) = \begin{bmatrix} u_1^2 \\ u_2^2 \\ u_1 u_2 \\ u_2 u_1 \end{bmatrix} \qquad \langle \phi(u), \phi(v) \rangle = u_1^2 v_1^2 + u_2^2 v_2^2 + 2 u_1 u_2 v_1 v_2$$

Feature space can get really large really quickly!

General d :

Dimension of $\phi(u)$ is roughly p^d if $u \in \mathbb{R}^p$

How do we deal with high-dimensional lifts/data?

A fundamental trick in ML: use kernels

A function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a *kernel* for a map ϕ if $K(x, x') = \phi(x) \cdot \phi(x')$ for all x, x'.

So, if we can represent our algorithms/decision rules as dot products
and we can find a kernel for our feature map
then we can avoid explicitly dealing with $\phi(x)$.

Examples of Kernels

- **Polynomials of degree exactly d**

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^p$$

- **Polynomials of degree up to d**

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^p$$

- **Gaussian (squared exponential) kernel**

$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||^2}{2\sigma^2}\right)$$

- **Sigmoid**

$$K(u, v) = \tanh(\gamma \cdot u^T v + r)$$

The Kernel Trick

Pick a kernel K

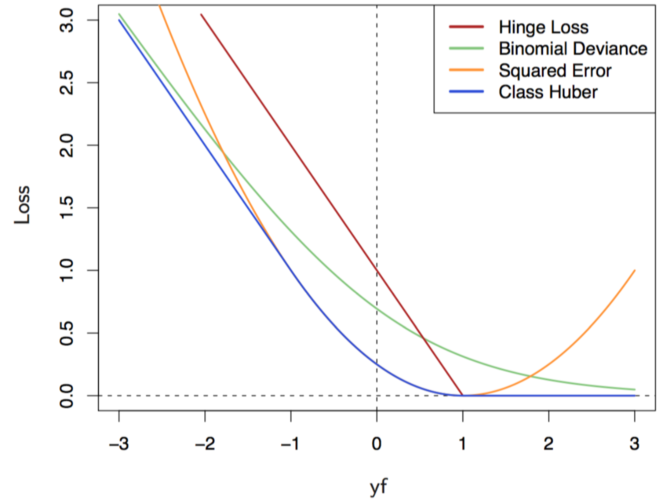
Prove $w = \sum_i \alpha_i x_i$

Change loss function/decision rule to only access data through dot products

Decision rule is easy: why?

Substitute $K(x_i, x_j)$ for $x_i^T x_j$

Loss Functions



$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

- Loss functions: $$\sum_{i=1}^n \ell_i(w)$$

Squared error Loss: $\ell_i(w) = (y_i - x_i^T w)^2$

Logistic Loss: $\ell_i(w) = \log(1 + \exp(-y_i\, x_i^T w))$

0/1 loss: $\ell_i(w) = \mathbb{I}[\text{sign}(y_i) \neq \text{sign}(x_i^T w)]$

Hinge Loss: $\ell_i(w) = \max\{0, 1 - y_i x_i^T w\}$

The Kernel Trick for SVMs

Pick a kernel K

Prove $w = \sum_i \alpha_i x_i$

Change loss function/decision rule to only access data through dot products

Substitute $K(x_i, x_j)$ for $x_i^T x_j$

The Kernel Trick for regularized least squares

$$\widehat{w} = \arg\min_w \sum_{i=1}^n (y_i - x_i^T w)^2 + \lambda ||w||_w^2$$

There exists an $\alpha \in \mathbb{R}^n$: $\widehat{w} = \sum_{i=1}^n \alpha_i x_i$ Why?

The Kernel Trick for regularized least squares

$$\widehat{w} = \arg\min_w \sum_{i=1}^n (y_i - x_i^T w)^2 + \lambda ||w||_w^2$$

There exists an $\alpha \in \mathbb{R}^n$: $\widehat{w} = \sum_{i=1}^n \alpha_i x_i$

$$\widehat{\alpha} = \arg\min_\alpha \sum_{i=1}^n (y_i - \sum_{j=1}^n \alpha_j \langle x_j, x_i \rangle)^2 + \lambda \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \langle x_i, x_j \rangle$$

The Kernel Trick for regularized least squares

$$\widehat{w} = \arg\min_w \sum_{i=1}^n (y_i - x_i^T w)^2 + \lambda ||w||_w^2$$

There exists an $\alpha \in \mathbb{R}^n$: $\widehat{w} = \sum_{i=1}^n \alpha_i x_i$

$$\widehat{\alpha} = \arg\min_\alpha \sum_{i=1}^n (y_i - \sum_{j=1}^n \alpha_j \langle x_j, x_i \rangle)^2 + \lambda \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \langle x_i, x_j \rangle$$

$$= \arg\min_\alpha \sum_{i=1}^n (y_i - \sum_{j=1}^n \alpha_j K(x_i, x_j))^2 + \lambda \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j K(x_i, x_j)$$

$$= \arg\min_\alpha ||\mathbf{y} - \mathbf{K}\alpha||_2^2 + \lambda \alpha^T \mathbf{K} \alpha$$

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

Why regularization?

Typically, $\mathbf{K} \succ 0$. What if $\lambda = 0$?

$$\widehat{\alpha} = \arg\min_{\alpha} ||\mathbf{y} - \mathbf{K}\alpha||_2^2 + \lambda \alpha^T \mathbf{K} \alpha$$

Why regularization?

Typically, $\mathbf{K} \succ 0$. What if $\lambda = 0$?

$$\widehat{\alpha} = \arg\min_{\alpha} ||\mathbf{y} - \mathbf{K}\alpha||_2^2 + \lambda \alpha^T \mathbf{K} \alpha$$

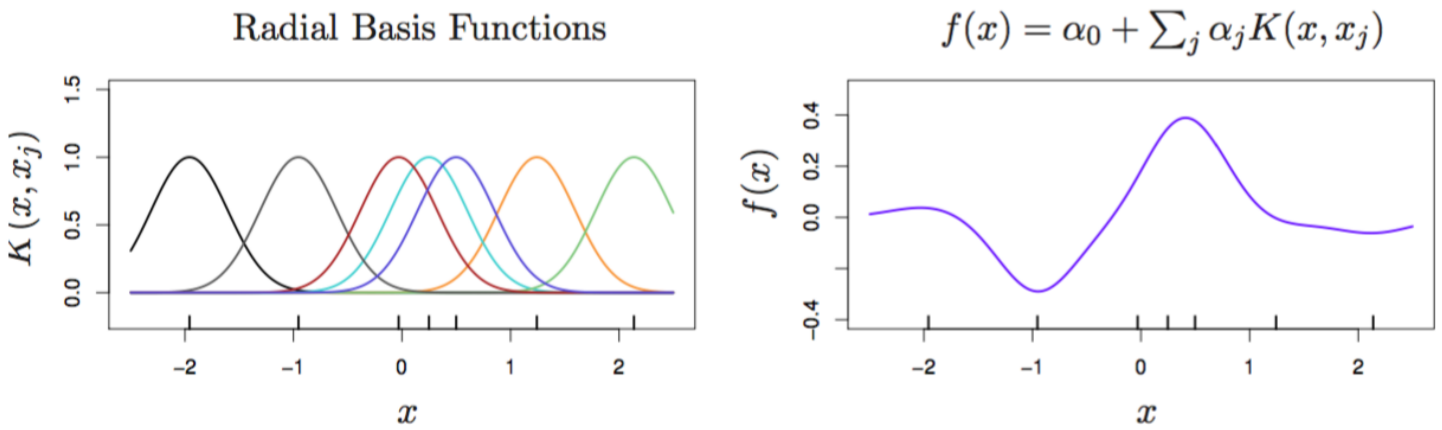
Unregularized kernel least squares can (over) fit **any data**!

$$\widehat{\alpha} = \mathbf{K}^{-1}\mathbf{y}$$

RBF Kernel

$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u}-\mathbf{v}||_2^2}{2\sigma^2}\right)$$

Note that this is like weighting "bumps" on each point like kernel smoothing but now we learn the weights

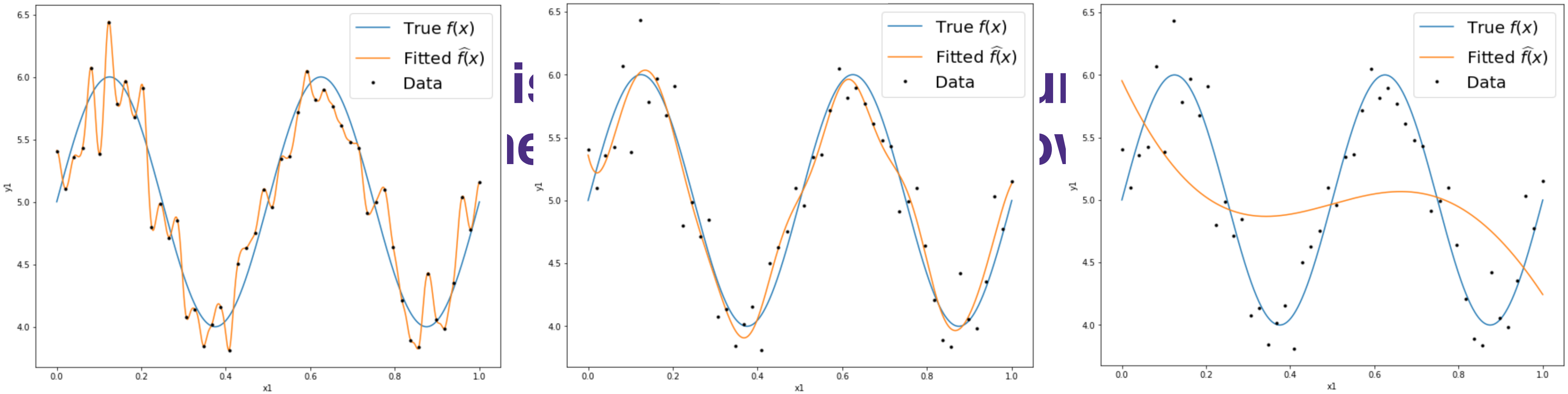


RBF Kernel

$$K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{||\mathbf{u}-\mathbf{v}||_2^2}{2\sigma^2}\right)$$

The bandwidth sigma has an enormous effect on fit:

$\sigma = 10^{-2} \quad \lambda = 10^{-4}$ $\qquad$ $\sigma = 10^{-1} \quad \lambda = 10^{-4}$ $\qquad$ $\sigma = 10^{-0} \quad \lambda = 10^{-4}$



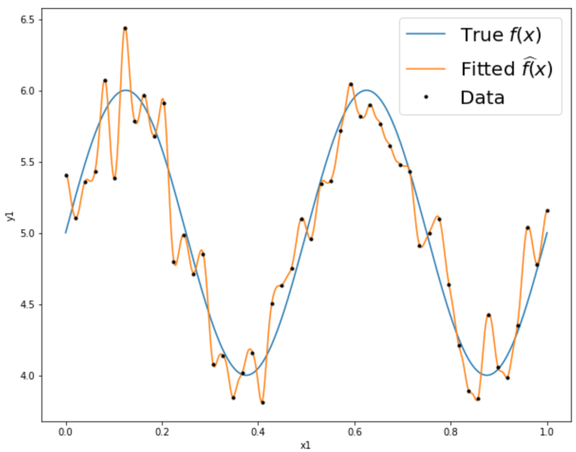
$$\widehat{f}(x) = \sum_{i=1}^{n} \widehat{\alpha}_i K(x_i, x)$$

RBF Kernel

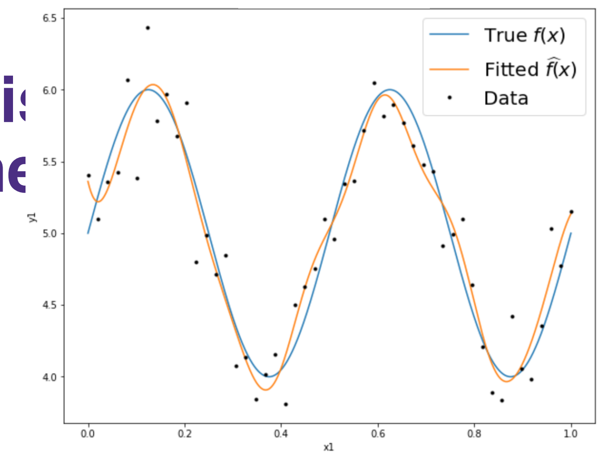
$$K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{||\mathbf{u}-\mathbf{v}||_2^2}{2\sigma^2}\right)$$

The bandwidth sigma has an enormous effect on fit:

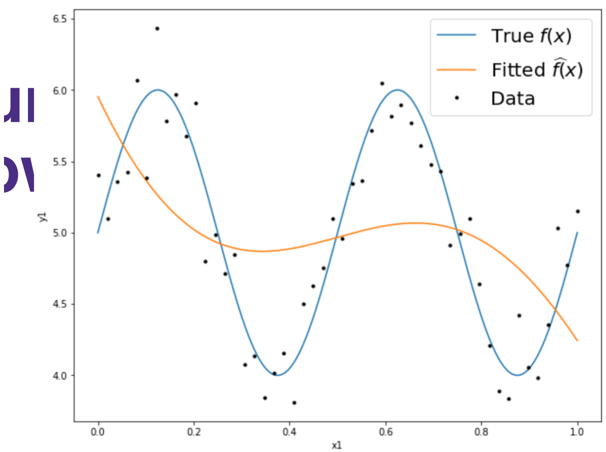
$\sigma = 10^{-2} \quad \lambda = 10^{-4}$



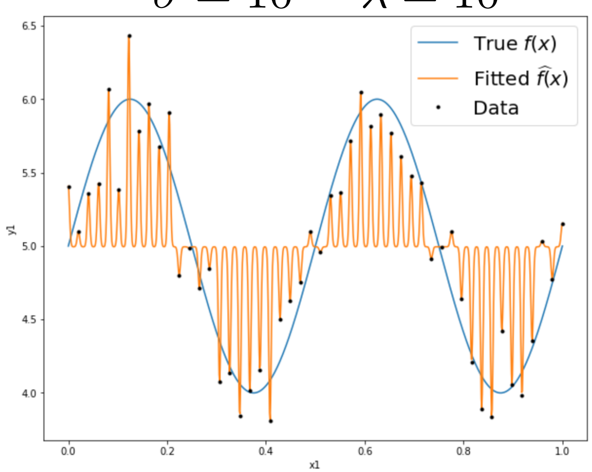
$\sigma = 10^{-1} \quad \lambda = 10^{-4}$



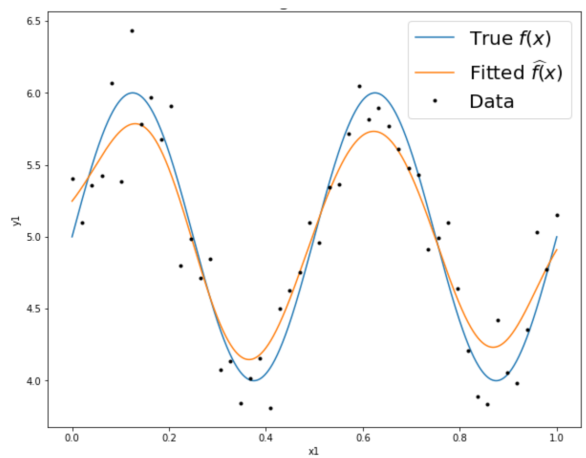
$\sigma = 10^{-0} \quad \lambda = 10^{-4}$



$\sigma = 10^{-3} \quad \lambda = 10^{-4}$



$\sigma = 10^{-1} \quad \lambda = 10^{-0}$



$$\widehat{f}(x) = \sum_{i=1}^{n} \widehat{\alpha}_i K(x_i, x)$$

RBF Kernel

$$K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{||\mathbf{u}-\mathbf{v}||_2^2}{2\sigma^2}\right)$$

Basis representation in 1d?

$$[\phi(x)]_i = \frac{1}{\sqrt{i!}} e^{-\frac{x^2}{2}} x^i \quad \text{for } i = 0, 1, \ldots$$

> **Note that this is like weighting "bumps" on each point like kernel smoothing but now we learn the weights**

$$\phi(x)^T\phi(x') = \sum_{i=0}^{\infty} \left(\frac{1}{\sqrt{i!}} e^{-\frac{x^2}{2}} x^i\right)\left(\frac{1}{\sqrt{i!}} e^{-\frac{(x')^2}{2}} (x')^i\right)$$

$$= e^{-\frac{x^2+(x')^2}{2}} \sum_{i=0}^{\infty} \frac{1}{i!}(xx')^i$$

$$= e^{-|x-x'|^2/2}$$

If n is very large, allocating an n-by-n matrix is tough. Can we truncate the above sum to approximate the kernel?

RBF kernel and random features

$$2\cos(\alpha)\cos(\beta) = \cos(\alpha+\beta) + \cos(\alpha-\beta)$$

$$e^{jz} = \cos(z) + j\sin(z)$$

Recall HW1 where we used the feature map:

$$\phi(x) = \begin{bmatrix} \sqrt{2}\cos(w_1^T x + b_1) \\ \vdots \\ \sqrt{2}\cos(w_p^T x + b_p) \end{bmatrix} \qquad \begin{array}{l} w_k \sim \mathcal{N}(0, 2\gamma I) \\ b_k \sim \text{uniform}(0,\pi) \end{array}$$

> **Isn't everything separable there? How are we not overfitting?**

$$\mathbb{E}[\frac{1}{p}\phi(x)^T\phi(y)] = \frac{1}{p}\sum_{k=1}^{p}\mathbb{E}[2\cos(w_k^T x + b_k)\cos(w_k^T y + b_k)]$$

$$= \mathbb{E}_{w,b}[2\cos(w^T x + b)\cos(w^T y + b)]$$

> **Regularization! Fat shattering (R/margin)^2**

> **What about sparsity?**

RBF kernel and random features

$$2\cos(\alpha)\cos(\beta) = \cos(\alpha+\beta) + \cos(\alpha-\beta)$$

$$e^{jz} = \cos(z) + j\sin(z)$$

Recall HW1 where we used the feature map:

$$\phi(x) = \begin{bmatrix} \sqrt{2}\cos(w_1^T x + b_1) \\ \vdots \\ \sqrt{2}\cos(w_p^T x + b_p) \end{bmatrix} \qquad \begin{array}{l} w_k \sim \mathcal{N}(0, 2\gamma I) \\ b_k \sim \text{uniform}(0, \pi) \end{array}$$

> **Isn't everything separable there? How are we not overfitting?**

$$\mathbb{E}[\frac{1}{p}\phi(x)^T\phi(y)] = \frac{1}{p}\sum_{k=1}^{p}\mathbb{E}[2\cos(w_k^T x + b_k)\cos(w_k^T y + b_k)]$$

> **Regularization! Fat shattering (R/margin)^2**

$$= \mathbb{E}_{w,b}[2\cos(w^T x + b)\cos(w^T y + b)]$$

$$= e^{-\gamma||x-y||_2^2}$$

[Rahimi, Recht NIPS 2007]

"NIPS Test of Time Award, 2018"

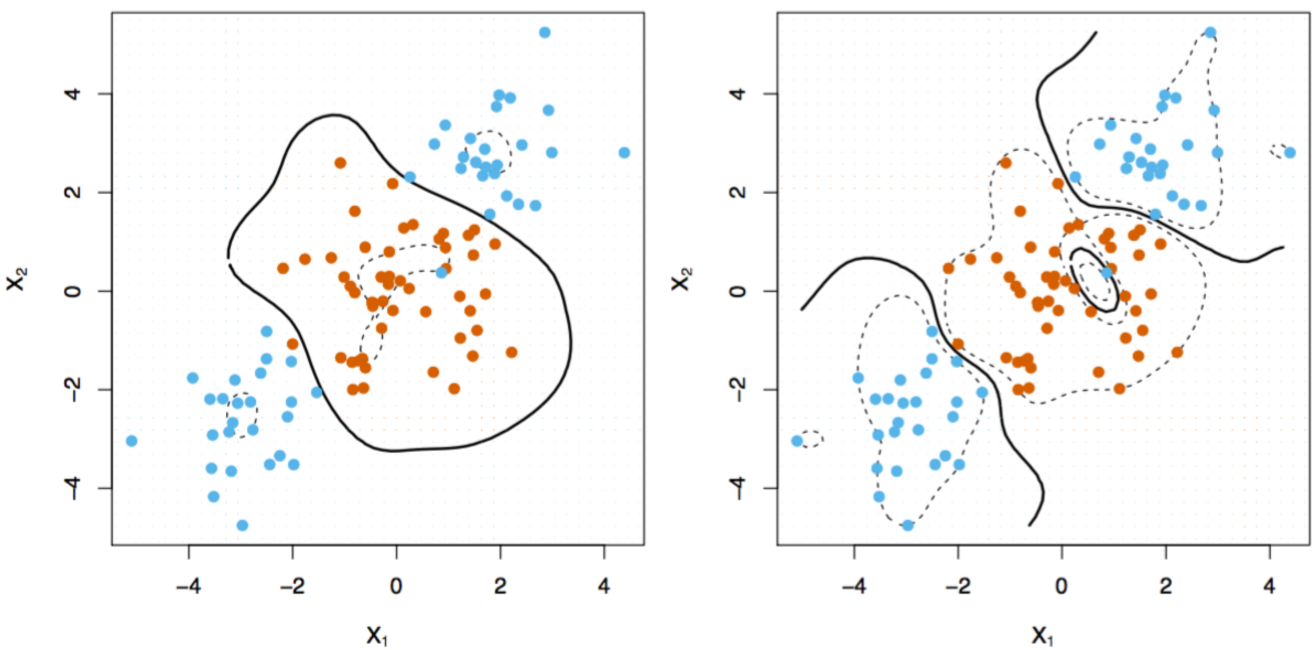
> **What about sparsity?**

RBF Classification

$$\widehat{w} = \sum_{i=1}^{n} \max\{0, 1 - y_i(b + x_i^T w)\} + \lambda ||w||_2^2$$

$$\min_{\alpha,b} \sum_{i=1}^{n} \max\{0, 1 - y_i(b + \sum_{j=1}^{n} \alpha_j \langle x_i, x_j \rangle)\} + \lambda \sum_{i,j=1}^{n} \alpha_i \alpha_j \langle x_i, x_j \rangle$$

- **Polynomials of degree exactly d**



Wait, infinite dimensions?

- Isn’t everything separable there? How are we not overfitting?

- Regularization! Fat shattering (R/margin)^2

String Kernels

Example from Efron and Hastie, 2016

Amino acid sequences of different lengths:

> Isn' ... re we not ove

x1

IPTSALVKETLALLSTHRTLLIANETLRIPVPVHKNHQLCTEEIFQGIGTLESQTVQGGTV
ERLFKNLSLIKKYIDGQKKKCGEERRRVNQFLDY**LQE**FLGVMNTEWI

x2

PHRRDLCSRSIWLARKIRSDLTALTESYVKHQGLWSELTEAER**LQE**NLQAYRTFHVLLA
RLLEDQQVHFTPTEGDFHQAIHTLLLQVAAFAYQIEELMILLEYKIPRNEADGMLFEKK
LWGLKV**LQE**LSQWTVRSIHDLRFISSHQTGIP

> Regularization: Fat shattering (R/margin)^2

All subsequences of length 3 (of possible 20 amino acids) $20^3 = 8{,}000$

$$h^3_{\text{LQE}}(x_1) = 1 \text{ and } h^3_{\text{LQE}}(x_2) = 2.$$

> **What about sparsity?**