ECE 5424: Introduction to Machine Learning

Topics:

- SVM
  - soft & hard margin
  - comparison to Logistic Regression

Readings: Barber 17.5

Stefan Lee
Virginia Tech

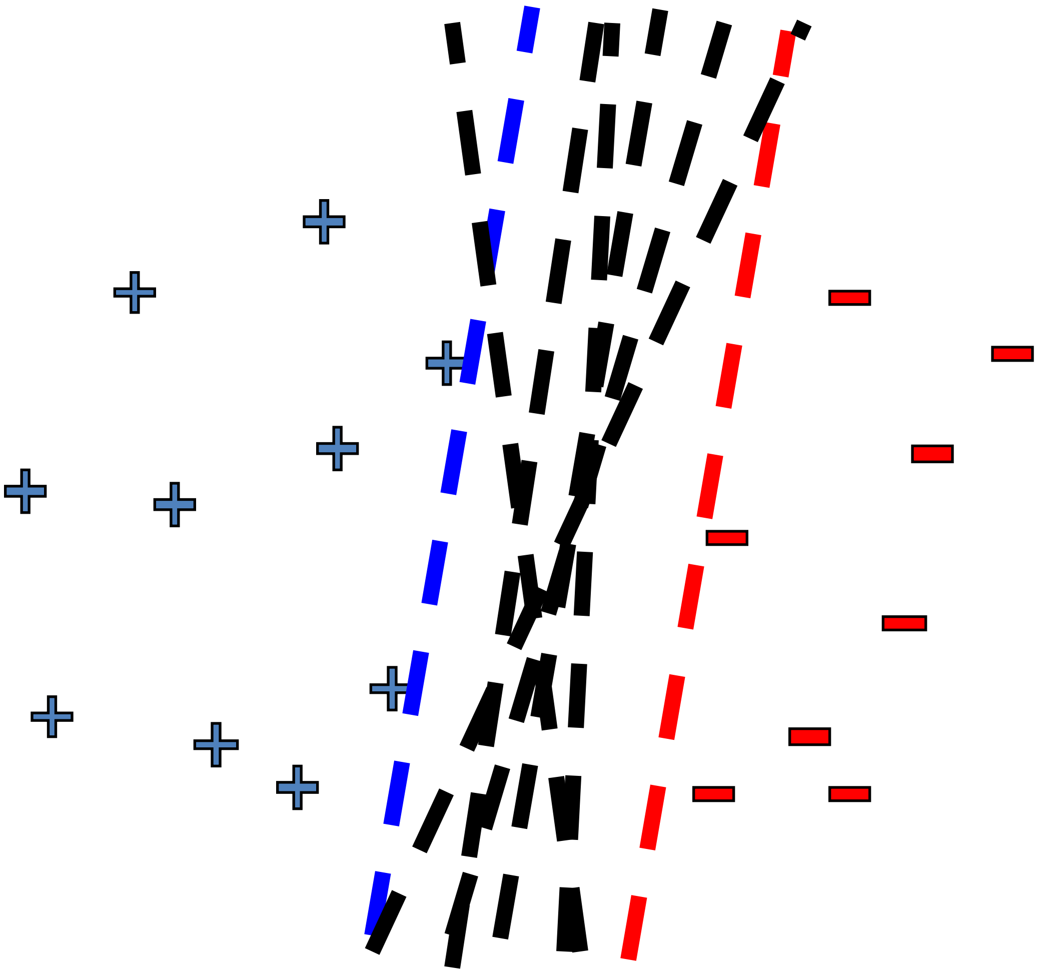
New Topic

Generative vs. Discriminative

- Generative Approach (Naïve Bayes)
  - Estimate p(x|y) and p(y)
  - Use Bayes Rule to predict y

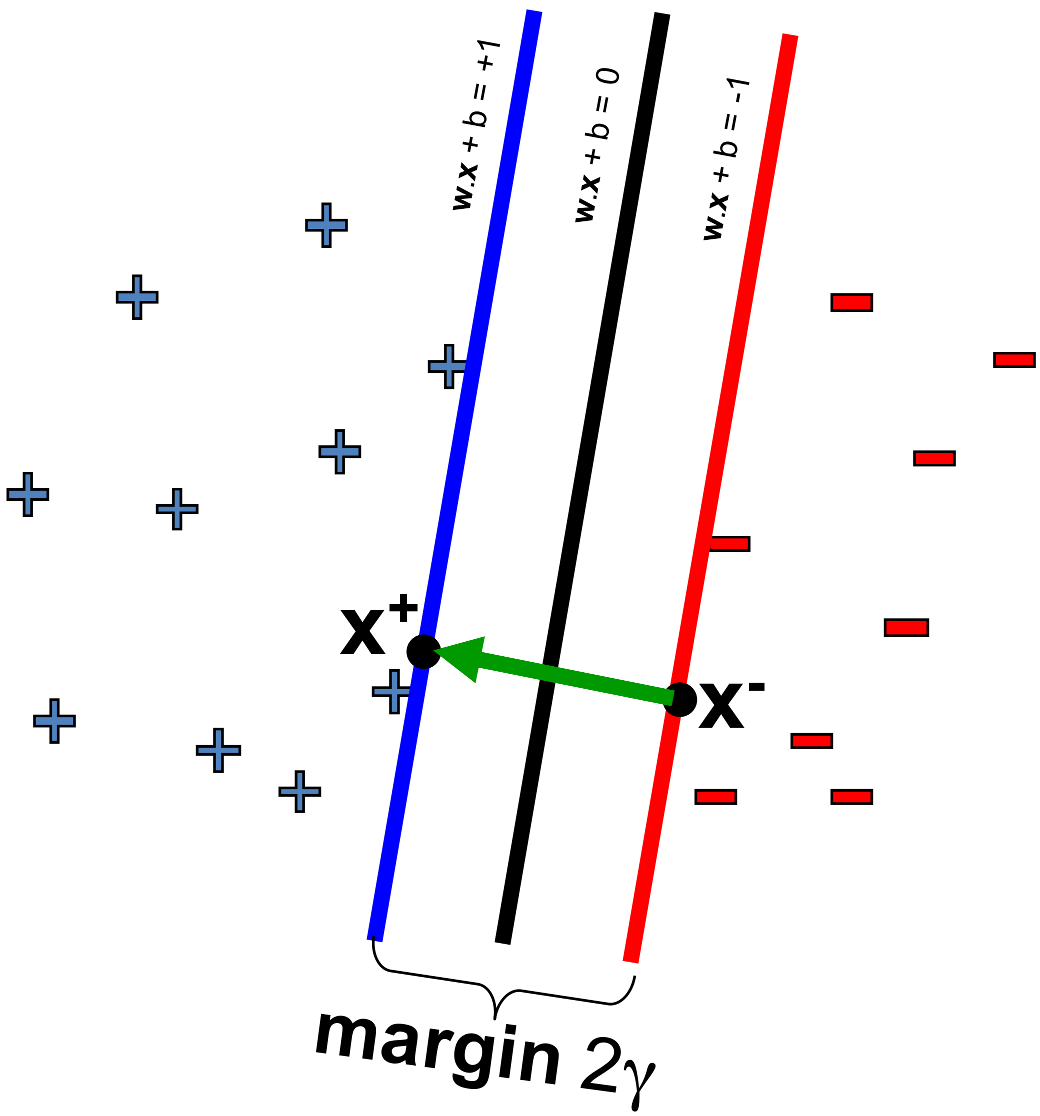
- Discriminative Approach
  - Estimate p(y|x) directly (Logistic Regression)
  - Learn "discriminant" function f(x) (Support Vector Machine)

Linear classifiers – Which line is better?

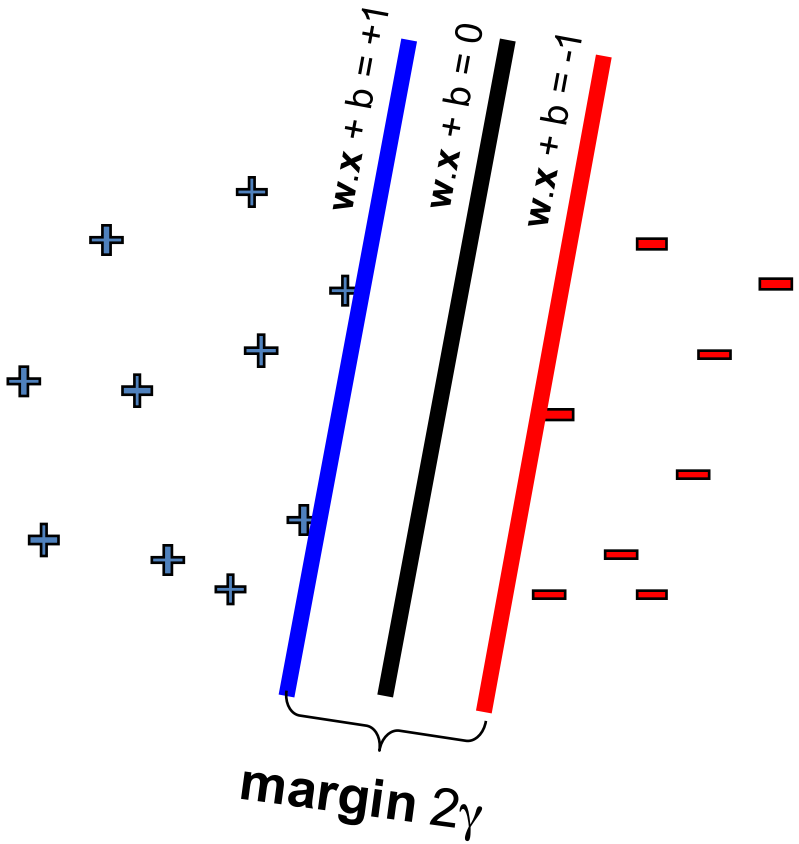


$\mathbf{w}.\mathbf{x} = \ {}_{j}\, w^{(j)}\, x^{(j)}$

Margin



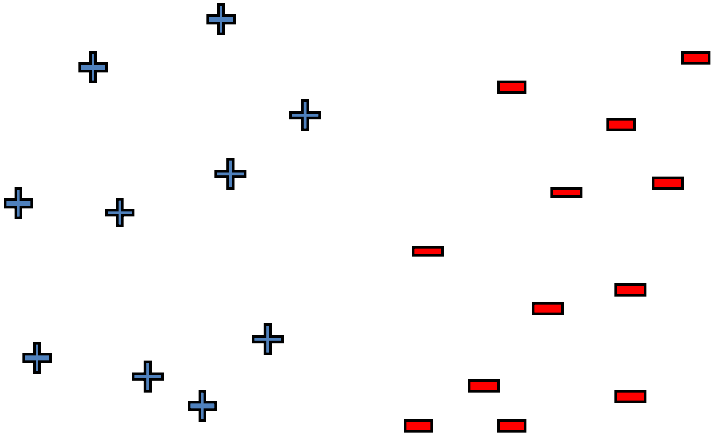
Support vector machines (SVMs)



$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w}$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \ \forall j$$

- Solve efficiently by quadratic programming (QP)
  - Well-studied solution algorithms
- Hyperplane defined by support vectors

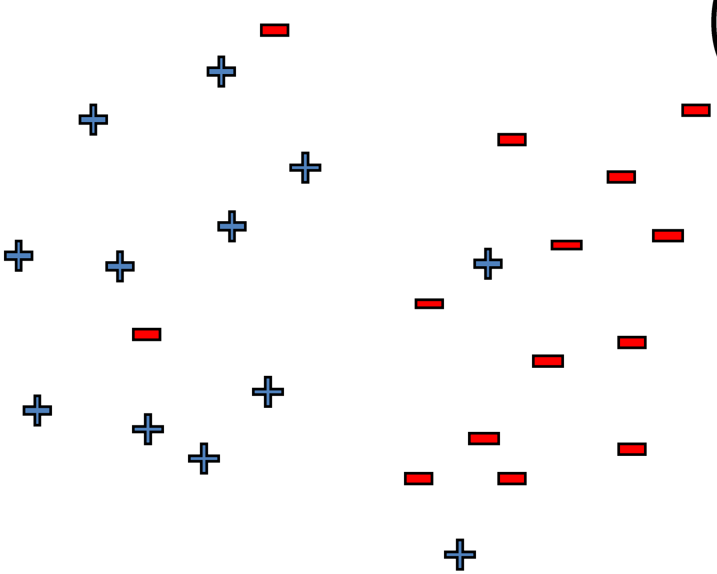
What if the data is not linearly separable?



Slide Credit: Carlos Guestrin

What if the data is not linearly separable?

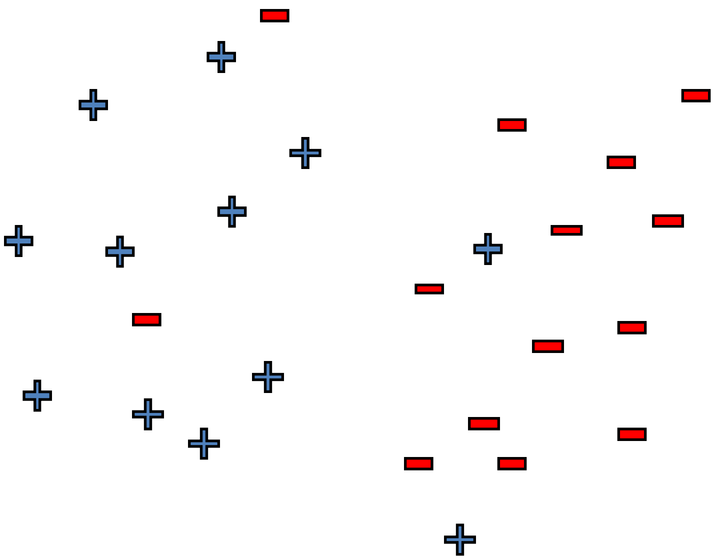
$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w}$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 \qquad , \forall j$$



- Minimize **w.w** and number of training mistakes
  - 0/1 loss
  - Slack penalty *C*
  - Not QP anymore
  - Also doesn’t distinguish near misses and really bad mistakes

Slack variables – Hinge loss

$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w} + C \sum_j \xi_j$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j, \ \forall j$$
$$\xi_j \geq 0, \ \forall j$$



- If margin >= 1, don't care
- If margin < 1, pay linear penalty

Soft Margin SVM

- Matlab Demo

Side note: What's the difference between SVMs and logistic regression?

SVM:

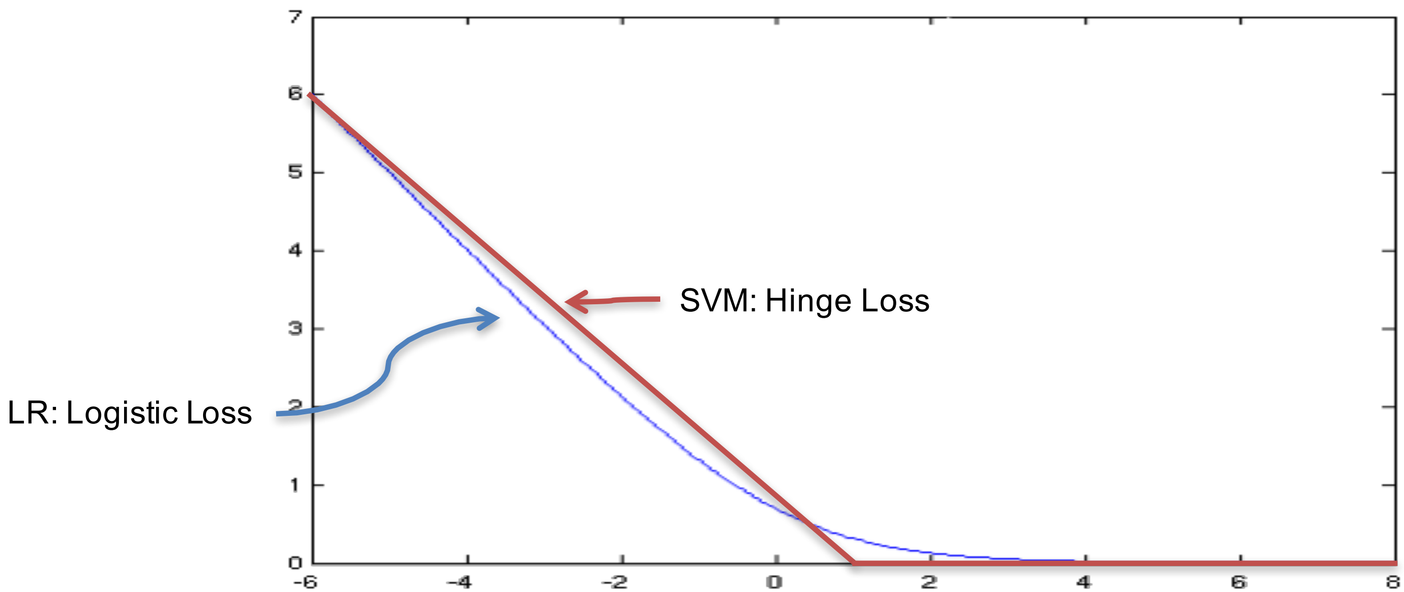
$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w} + C \sum_j \xi_j$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j, \ \ \forall j$$
$$\xi_j \geq 0, \ \ \forall j$$

Logistic regression:

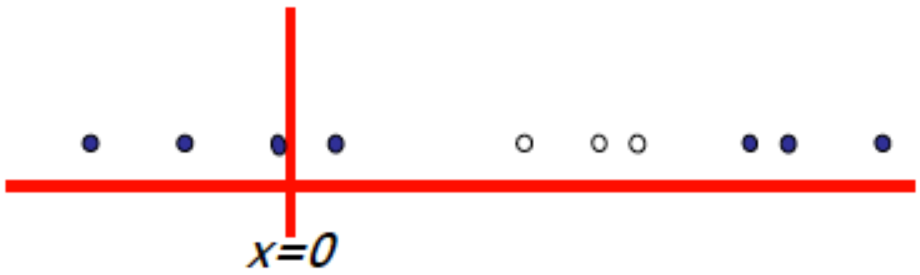
$$P(Y = 1 \mid x, \mathbf{w}) \ = \ \frac{1}{1 + e^{-(\mathbf{w}.\mathbf{x}+b)}}$$

Log loss:

$$-\ln P(Y = 1 \mid x, \mathbf{w}) \ = \ \ln \left(1 + e^{-(\mathbf{w}.\mathbf{x}+b)}\right)$$



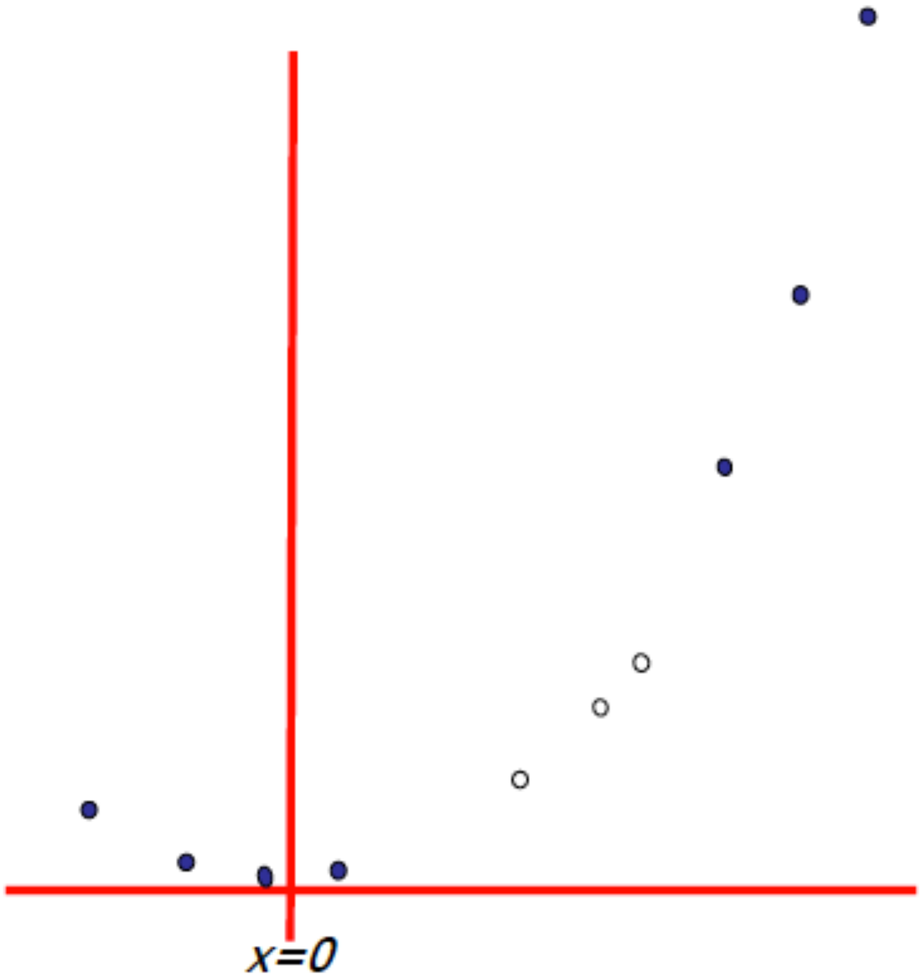
Harder 1-dimensional dataset



That's wiped the smirk off SVM's face.

What can be done about this?

Harder 1-dimensional dataset

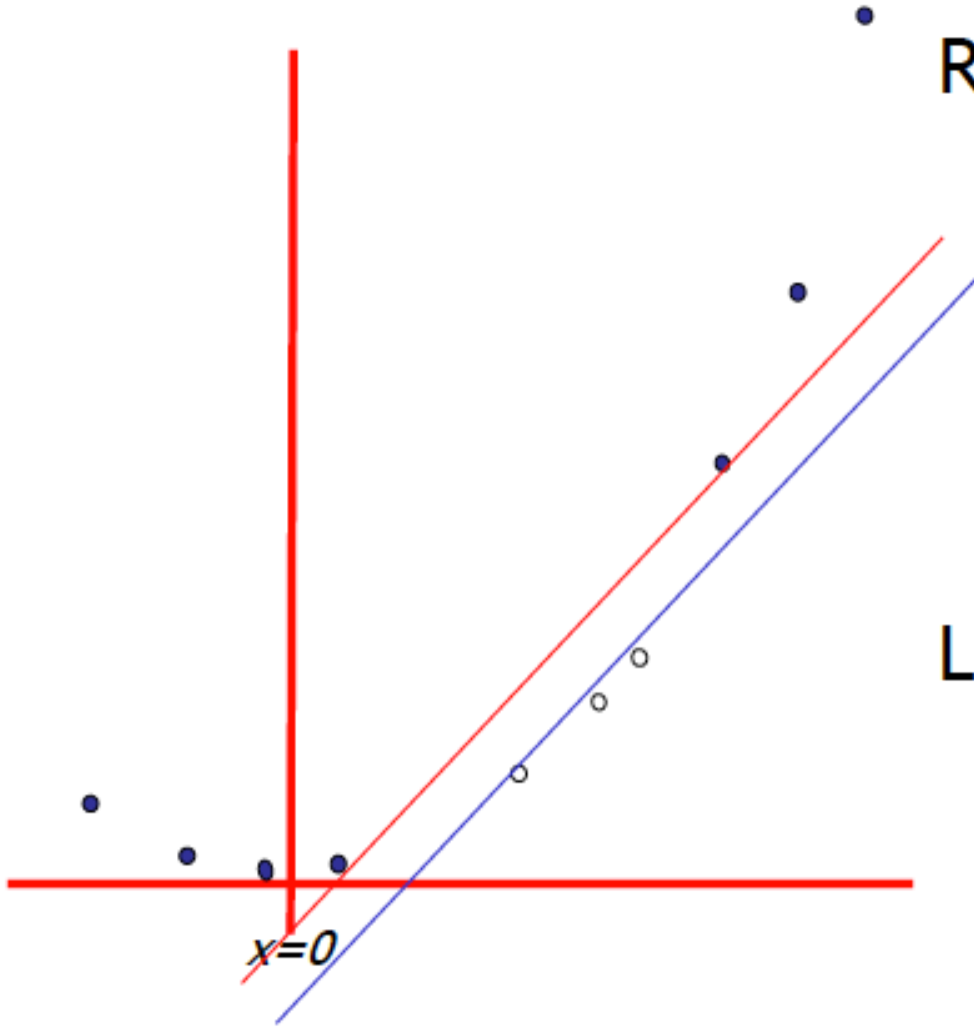


Remember how permitting non-linear basis functions made linear regression so much nicer?

Let's permit them here too

$$\mathbf{z}_k = (x_k, x_k^2)$$

Harder 1-dimensional dataset



Remember how permitting non-linear basis functions made linear regression so much nicer?

Let's permit them here too

$$\mathbf{z}_k = (x_k, x_k^2)$$

Does this always work?

- In a way, yes

Lemma

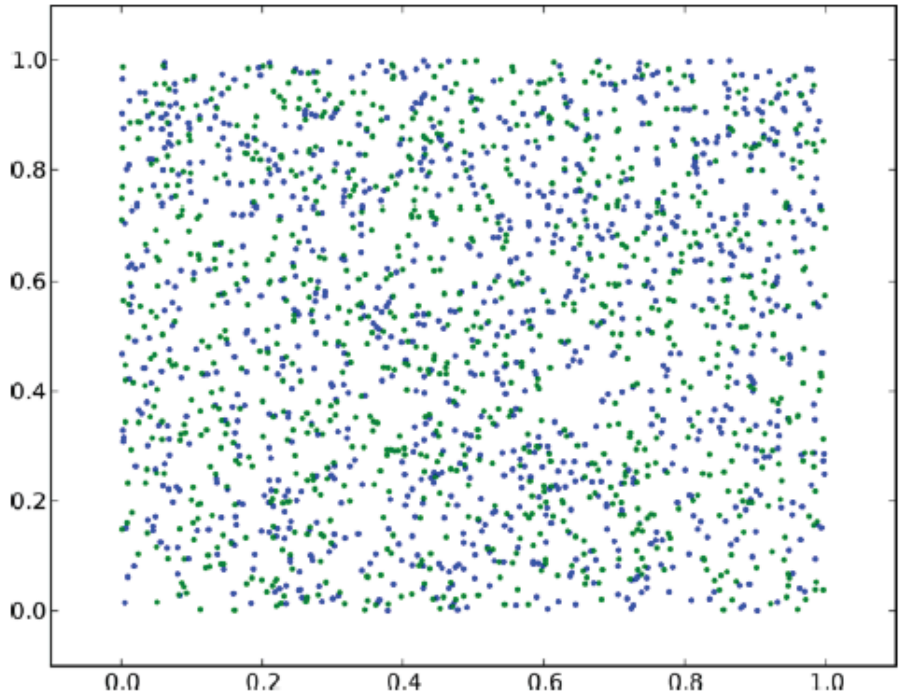
Let $(x_i)_{i=1,\ldots,n}$ *with* $x_i \neq x_j$ *for* $i \neq j$. *Let* $\varphi : \mathbb{R}^k \to \mathbb{R}^m$ *be a feature map. If the set* $\varphi(x_i)_{i=1,\ldots,n}$ *is linearly independent, then the points* $\varphi(x_i)_{i=1,\ldots,n}$ *are linearly separable.*

Lemma

If we choose $m > n$ *large enough, we can always find a map* φ.

Caveat

Caveat: We can separate *any* set, not just one with "reasonable" y_i:



There is a fixed feature map $\varphi : \mathbb{R}^2 \rightarrow \mathbb{R}^{20001}$ such that – no matter how we label them – there is always a hyperplane classifier that has 0 training error.

Kernel Trick

- One of the most interesting and exciting advancement in the last 2 decades of machine learning
  - The "kernel trick"
  - High dimensional feature spaces at no extra cost!

- But first, a detour
  - Constrained optimization!