CPSC 340: Machine Learning and Data Mining

Kernel Methods

Fall 2016

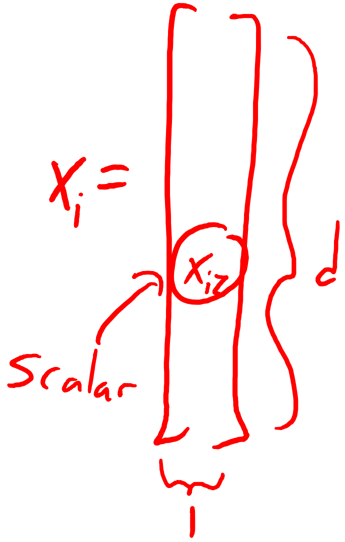
Admin

- Assignment 2:
  – Solution posted.
- Assignment 3:
  – Due Wednesday (before midnight anywhere on Earth).
  – Solutions released next Wednesday after class (last possible late class).
- Midterm on Friday October 28.
  – Midterm from last year and list of topics posted (covers Assignments 1-3)
  – In class, 55 minutes, closed-book, cheat sheet: 2-pages each double-sided.

Part 3 Review

- Focus of Part 3 is linear models:
  - Supervised learning where prediction is linear combination of features:

$$y_i = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id}$$
$$= w^T x_i$$



- Change of basis: replace features x_i with z_i:
  - Add a bias variable (feature that is always one).
  - Polynomial basis.
  - Radial basis functions (non-parametric basis).
- Regression:
  - Target y_i is numerical.
  - Testing whether (yhat == y_i) doesn't make sense.

Good fit that doesn't exactly pass through any point.

Part 3 Review

- Alternate error functions for regression:
  - Squared error: $$\frac{1}{2}\sum_{i=1}^{n}(w^Tx_i - y_i)^2 \quad \text{or} \quad \frac{1}{2}\|Xw - y\|^2$$
    - Can find optimal 'w' by solving linear system.
  - L_1-norm and L_∞-norm errors: $$\|Xw - y\|_1 \qquad \|Xw - y\|_\infty$$
    - More/less robust to outliers.
- L2-regularization:
  - Adding a penalty on the L2-norm of 'w' to decrease overfitting:

$$f(w) = \|Xw - y\|_1 + \frac{\lambda}{2}\|w\|^2$$

Part 3 Review

- Gradient descent:
  - Can we used to find a local minimum of a smooth function.
- L_1-norm and L_∞-norm errors are convex but non-smooth:
  - But we can smooth them using Huber and log-sum-exp functions.
- Convex functions:
  - Special functions where all local minima are global minima.
  - Simple rules for showing that a function is convex.

Last Time: Classification using Regression

- Binary classification using sign of linear models:

  Fit model $y_i = w^T x_i$ and predict using $\text{sign}(w^T x_i)$

  $y_i \to +1, -1$

- Problems with existing errors:
  - If $y_i = +1$ and $w^T x_i = +100$, then squared error $(w^T x_i - y_i)^2$ is huge.
  - Hard to minimize training error ("0-1 loss") in terms of 'w'.
- Motivates convex approximations to 0-1 loss:
  - Logistic loss (logistic regression): $\sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i)) + \frac{\lambda}{2}\|w\|^2$
  - Hinge loss (support vector machine): $\sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{\lambda}{2}\|w\|^2$

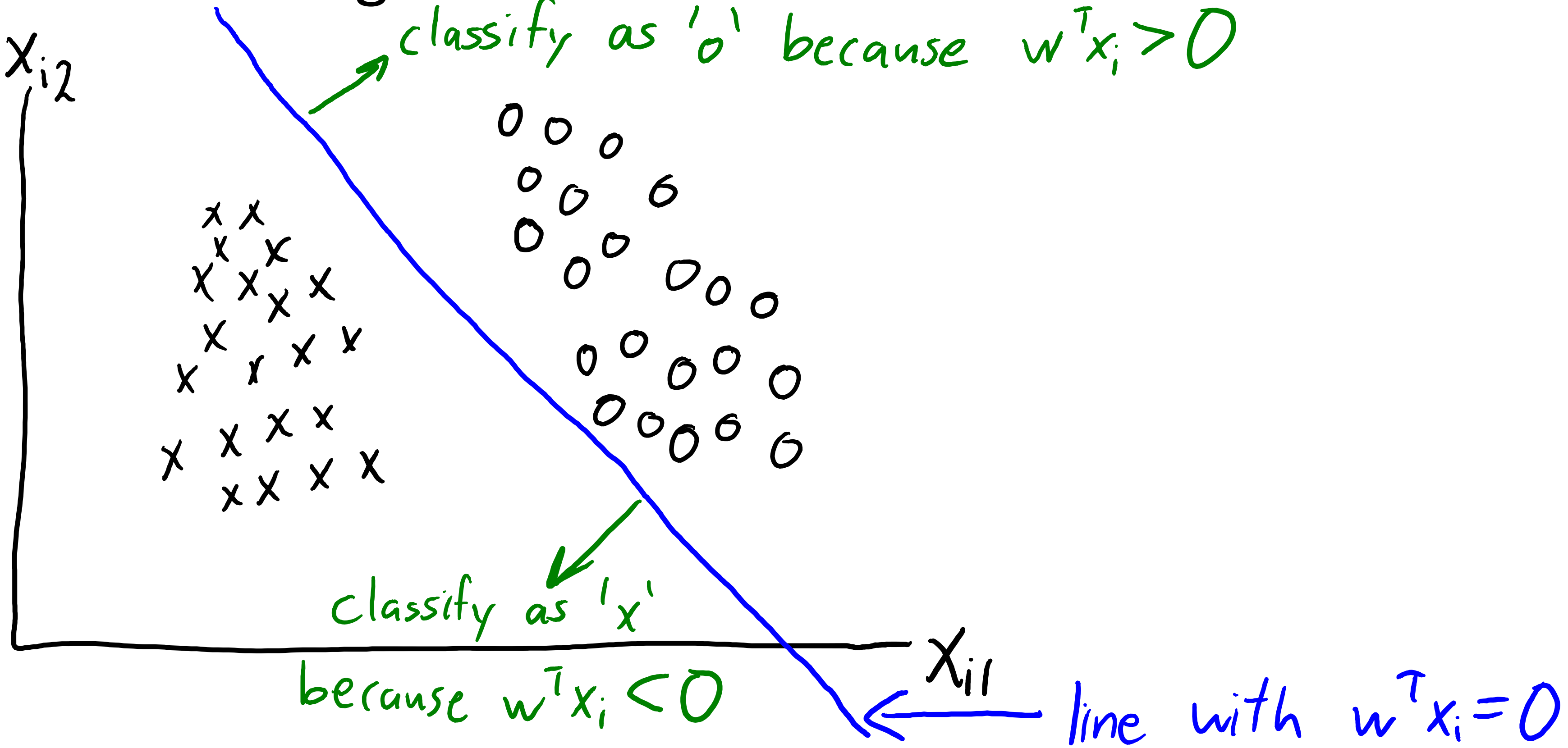
Last Time: Classification using Regression

- Can minimize smooth/convex logistic loss using gradient descent.
  - There are also efficient methods for support vector machines (SVMs).
- Logistic regression and SVMs are used EVERYWHERE!
  - Fast training and testing, weights w_j are easy to understand.
  - With high-dimensional features and regularization, often good test error.
    - Otherwise, often good test error with RBF basis and regularization.

- Some random questions you might be asking:
  - Can we use a polynomial basis with more than 1 feature?
  - Why didn't we do the "textbook" derivation of logistic/SVM?
  - How do we train on all of Gmail?
  - Did we miss feature selection?

2D View of Linear Classifiers

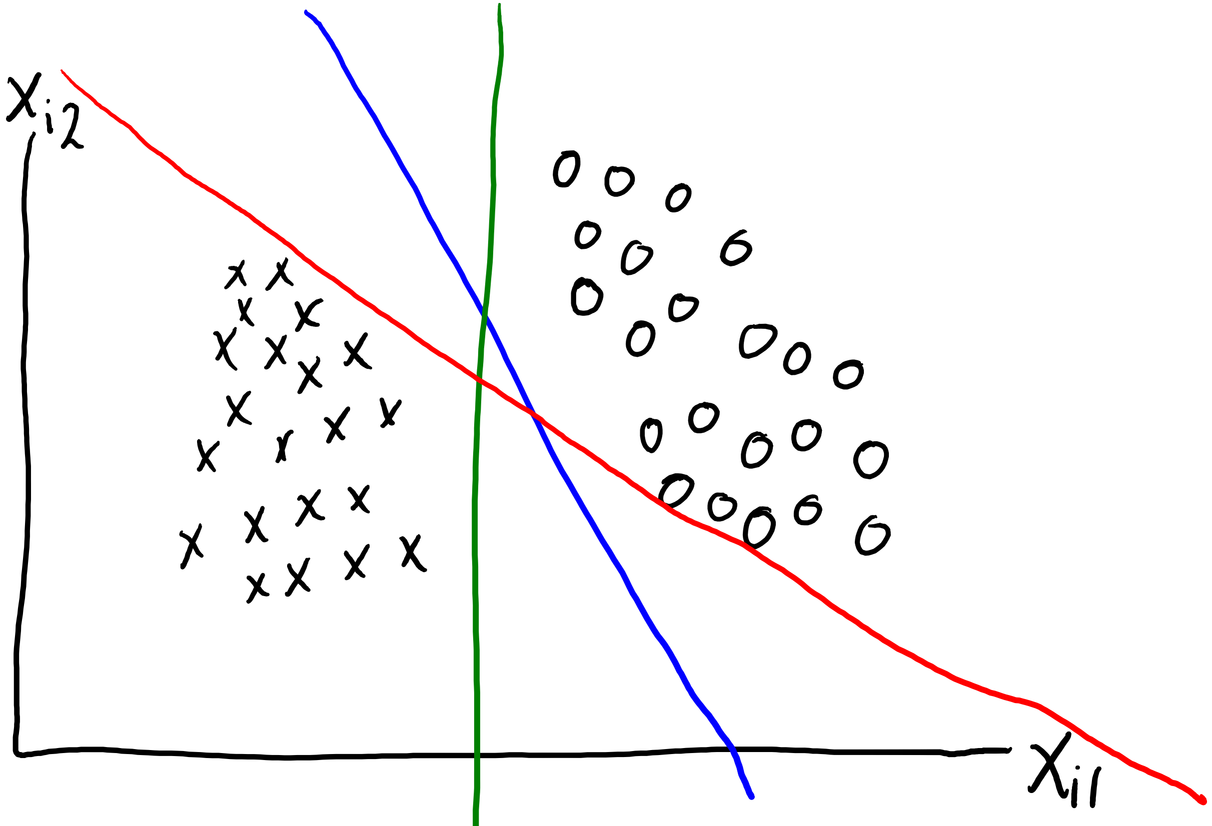
- 2D Visualization of linear regression for classification:



- “Linearly separable”: a perfect linear classifier exists.

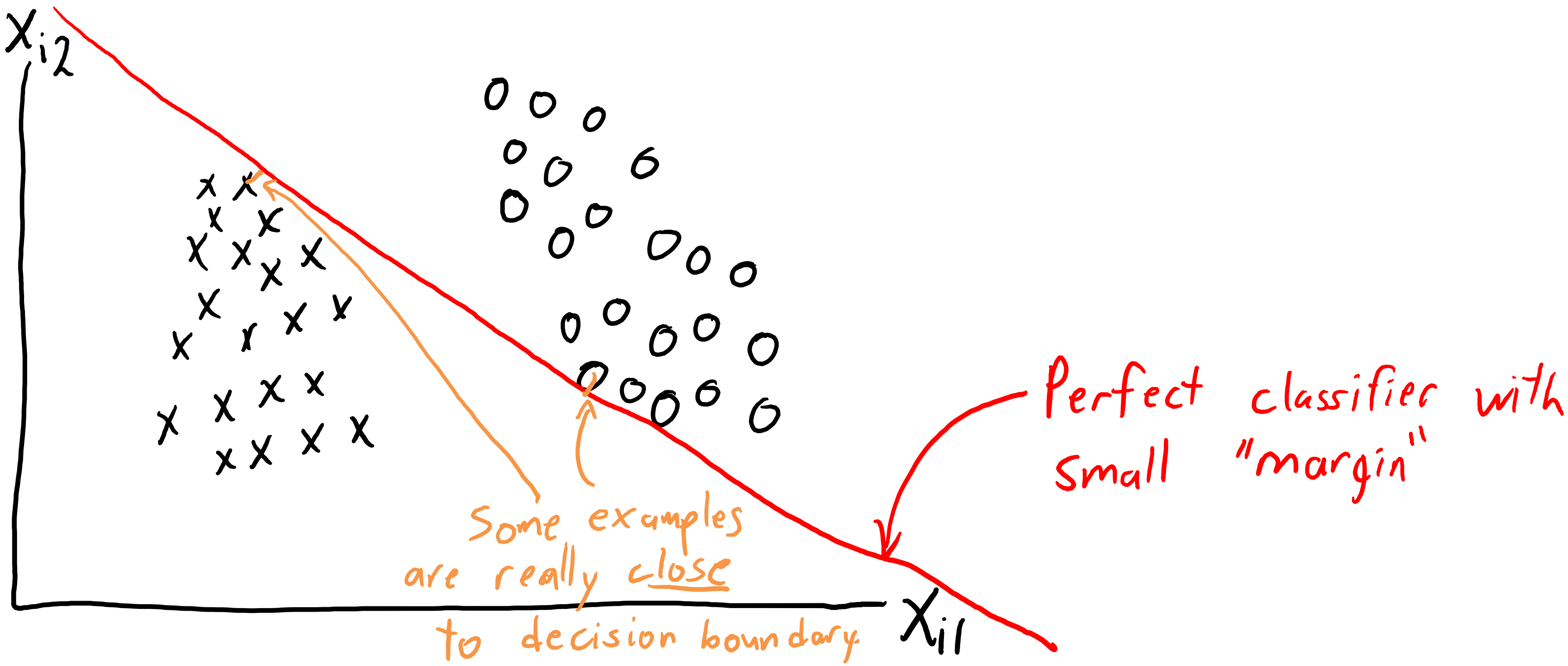
Maximum-Margin Classifier

- Consider a linearly-separable dataset.
  - "Perceptron" algorithm finds *some* classifier with zero error.
  - But are all zero-error classifiers equally good?



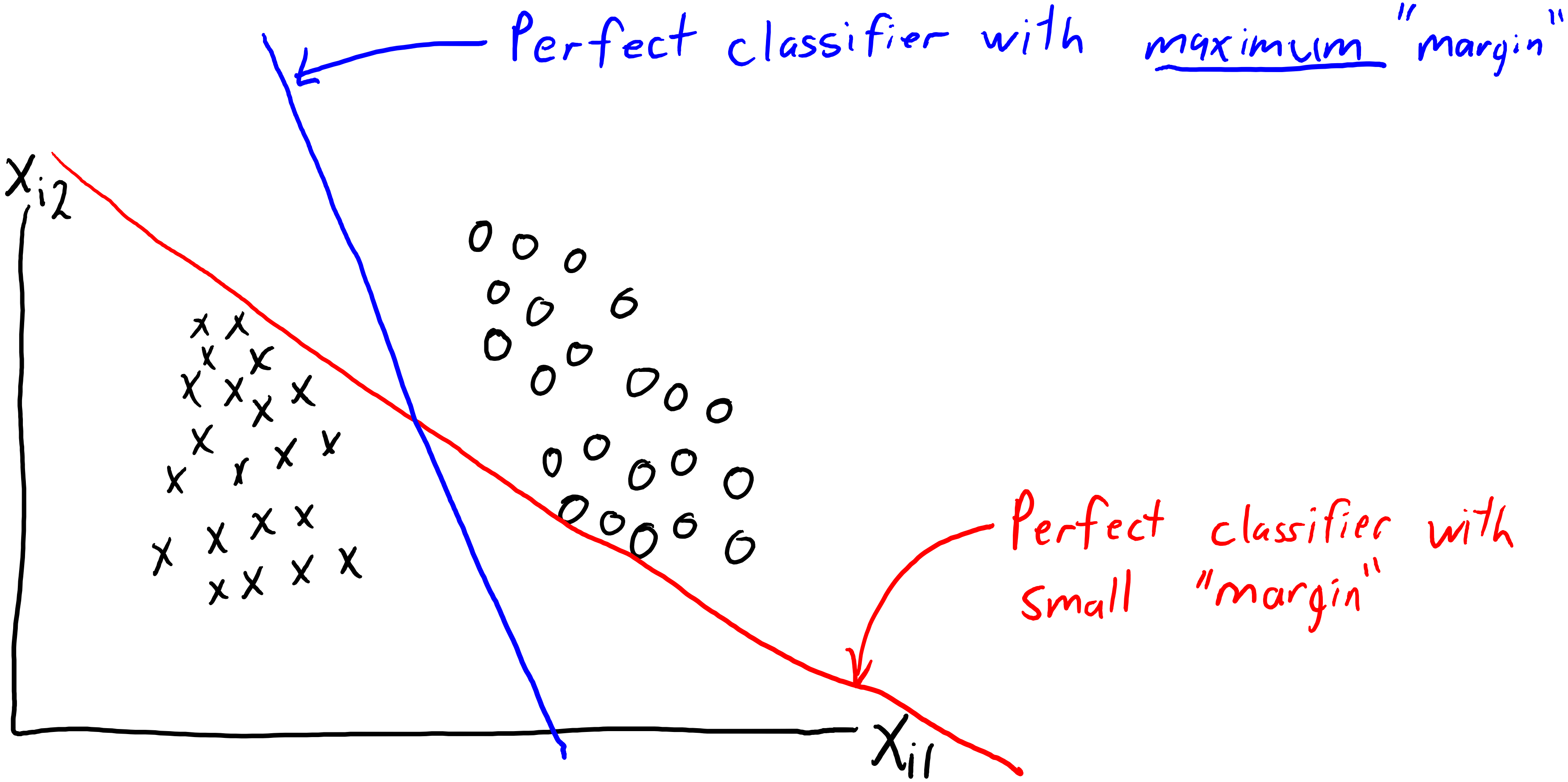
Maximum-Margin Classifier

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.



Maximum-Margin Classifier

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.

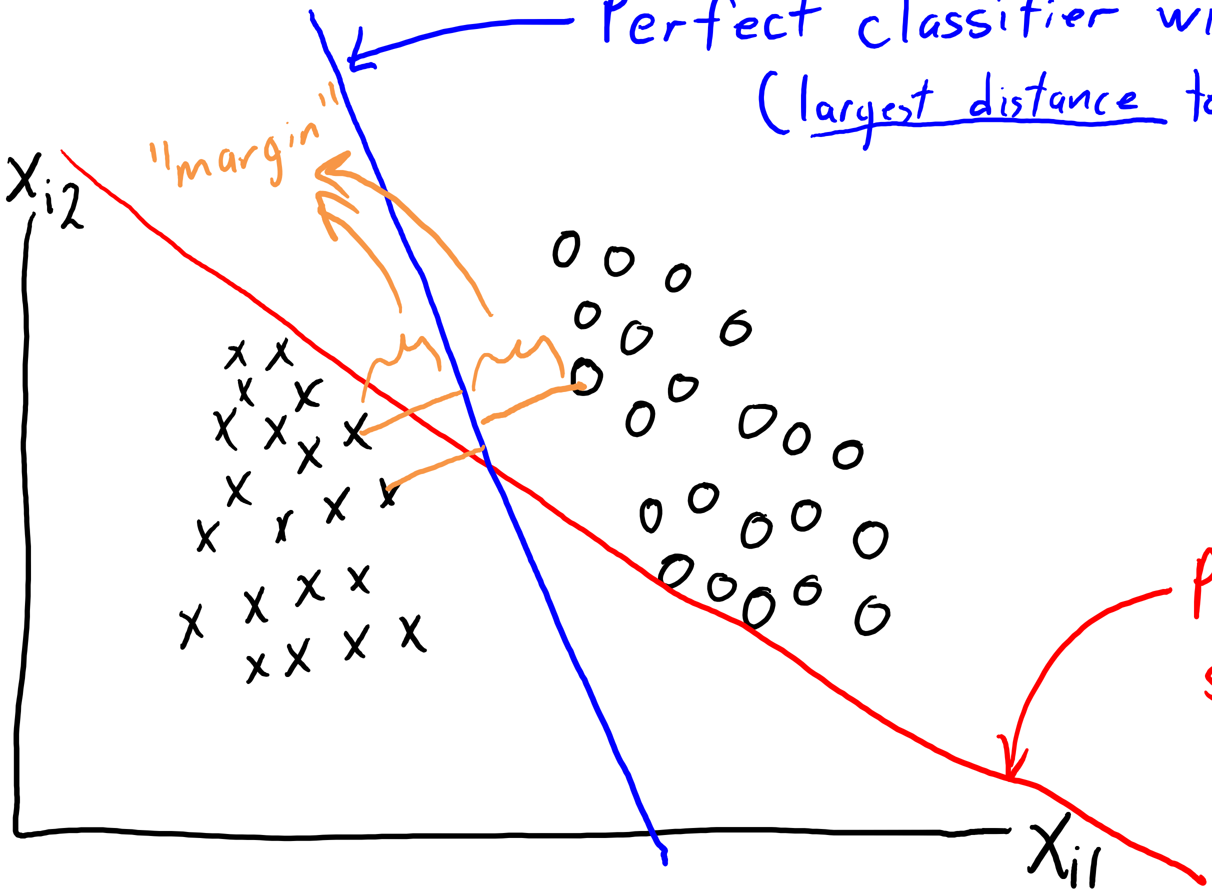


Maximum-Margin Classifier

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.

Why maximize margin?

If test is close to training data, then max margin leaves more "room" before we make an error.

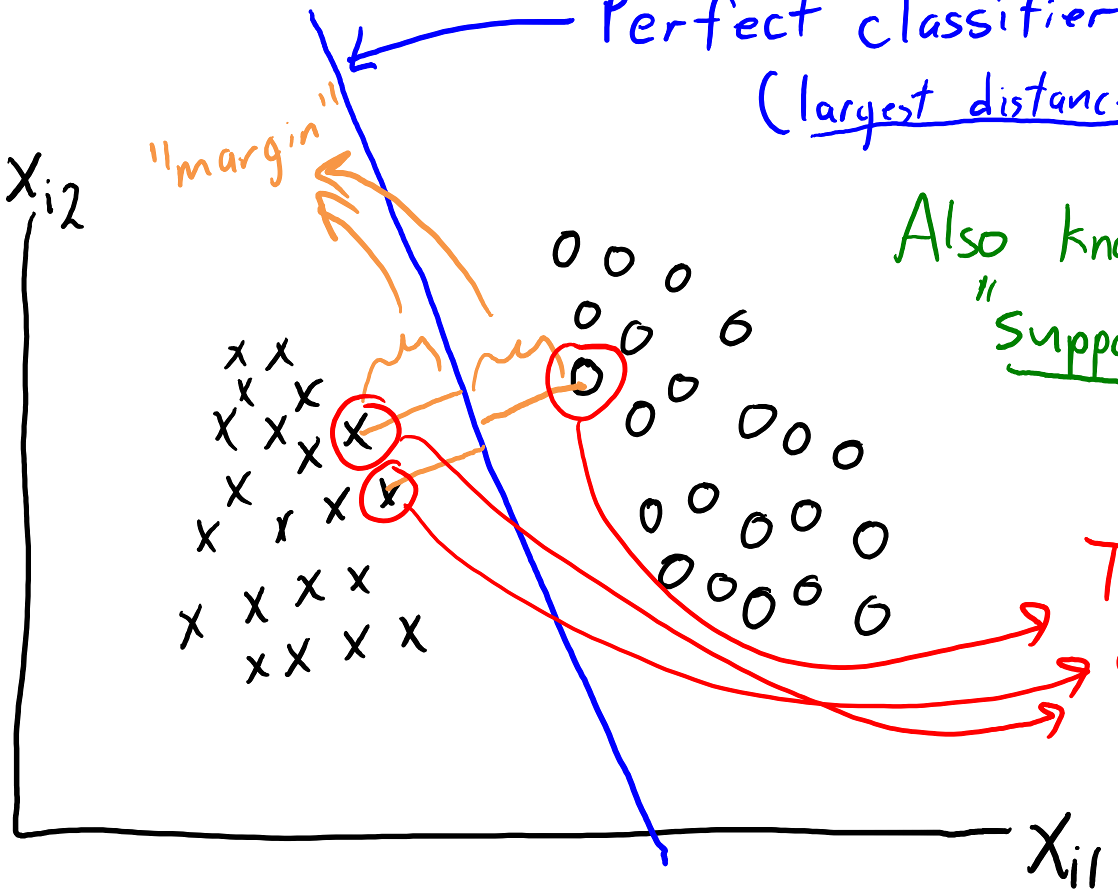


Maximum-Margin Classifier

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.

Perfect classifier with maximum "margin"
(largest distance to closest examples)

Also known as a "support vector machine"



The examples that are closest are called "support vectors"

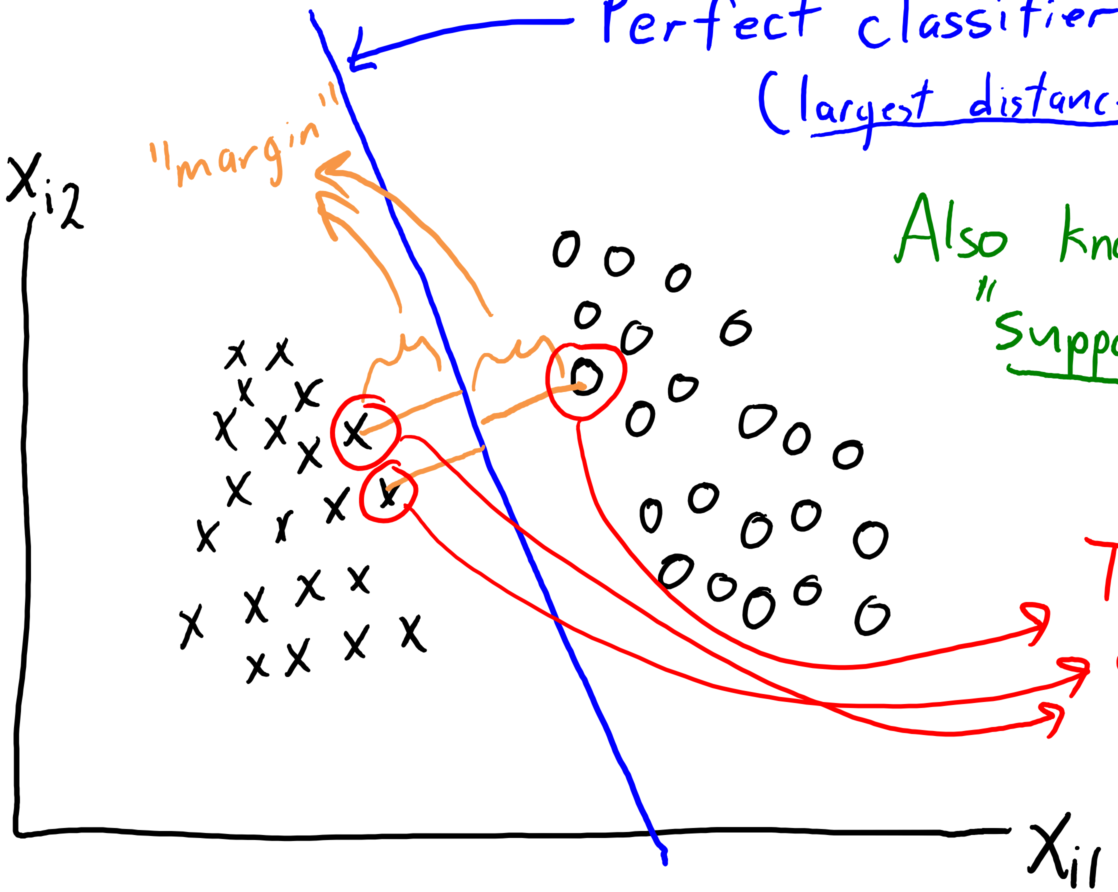
Maximum-Margin Classifier

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.

Perfect classifier with maximum "margin" (largest distance to closest examples)

Also known as a "support vector machine"

Final classifier only depends on support vectors



The examples that are closest are called "support vectors"

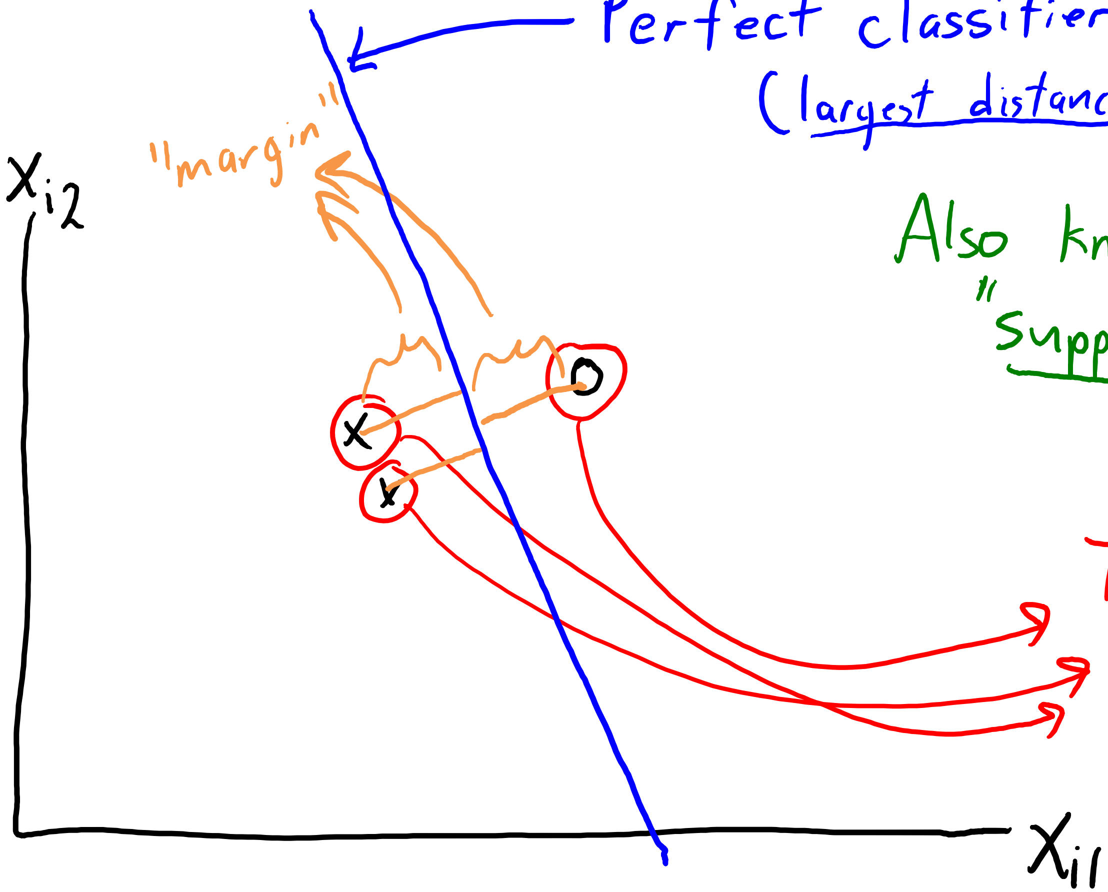
Maximum-Margin Classifier

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.

Perfect classifier with maximum "margin" (largest distance to closest examples)

Also known as a "support vector machine"

Final classifier only depends on support vectors



You could throw away the other examples and get the same classifier.

The examples that are closest are called "support vectors"

Support Vector Machines

- For linearly-separable data, support vector machine (SVM) minimizes:

$$f(w) = \frac{1}{2}\|w\|^2$$

  – Subject to the constraints that: (see Wikipedia or ML textbooks)

$$\left.\begin{aligned} w^T x_i &\geq 1 \quad \text{for } y_i = 1 \\ w^T x_i &\leq -1 \quad \text{for } y_i = -1 \end{aligned}\right\}$$

Or we can write both cases as

$$y_i w^T x_i \geq 1$$

- For non-separable data, try to minimize violation of constraints:

We want $y_i w^T x_i \geq 1$

or equivalently $0 \geq 1 - y_i w^T x_i$

Since we can't satisfy this for all 'i', let's add "slack" $\beta_i \geq 0$ to each constraint:

$$\beta_i \geq 1 - y_i w^T x_i$$

Support Vector Machines

- For non-separable data, try to minimize violation of constraints:

We want $y_i w^T x_i \geq 1$

or equivalently $0 \geq 1 - y_i w^T x_i$

Since we can't satisfy this for all 'i', let's add "slack" $\beta_i \geq 0$ to each constraint: $\beta_i \geq 1 - y_i w^T x_i$

- For non-separable data, we usually define SVMs as minimum of:

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

Hinge loss for example 'i':

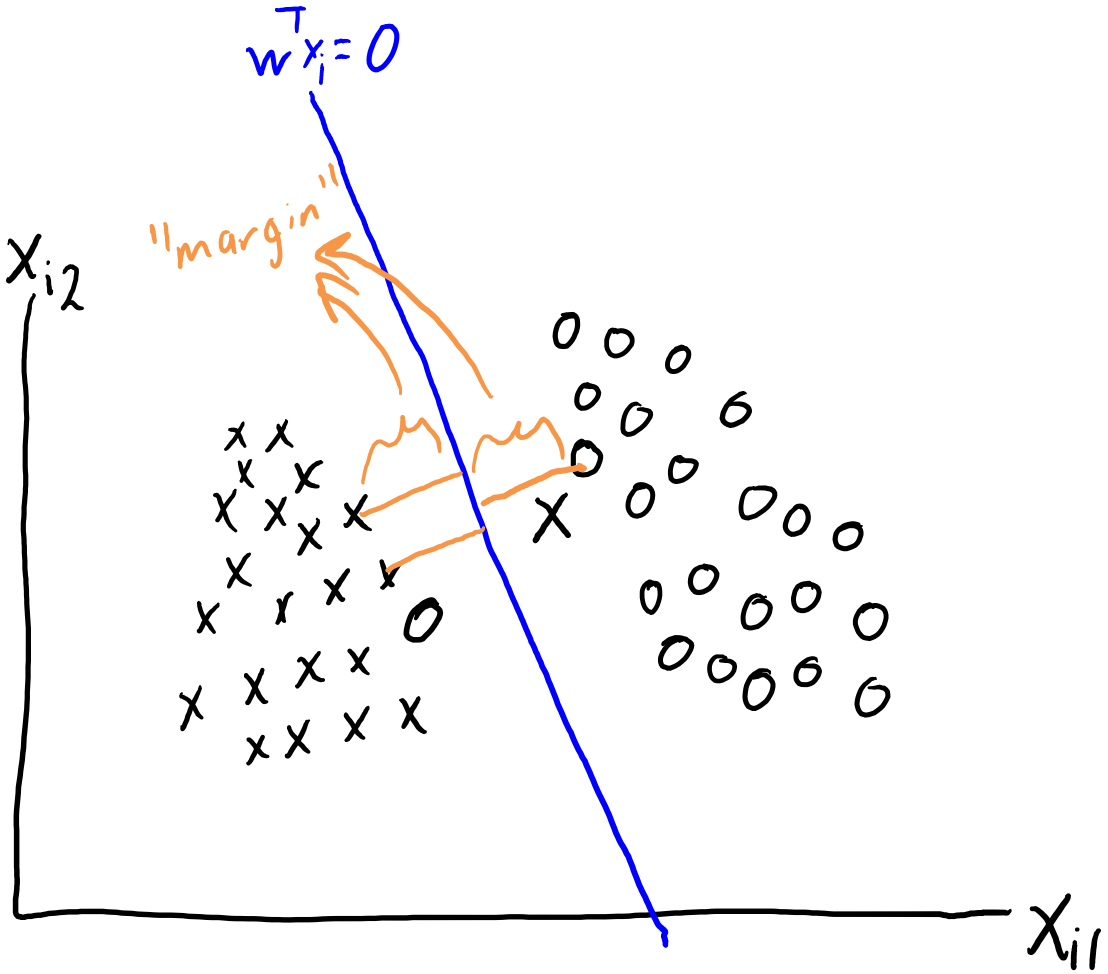
it's the amount we violate $y_i w^T x_i \geq 1$

"slack"

Original SVM objective: encourages large margin.

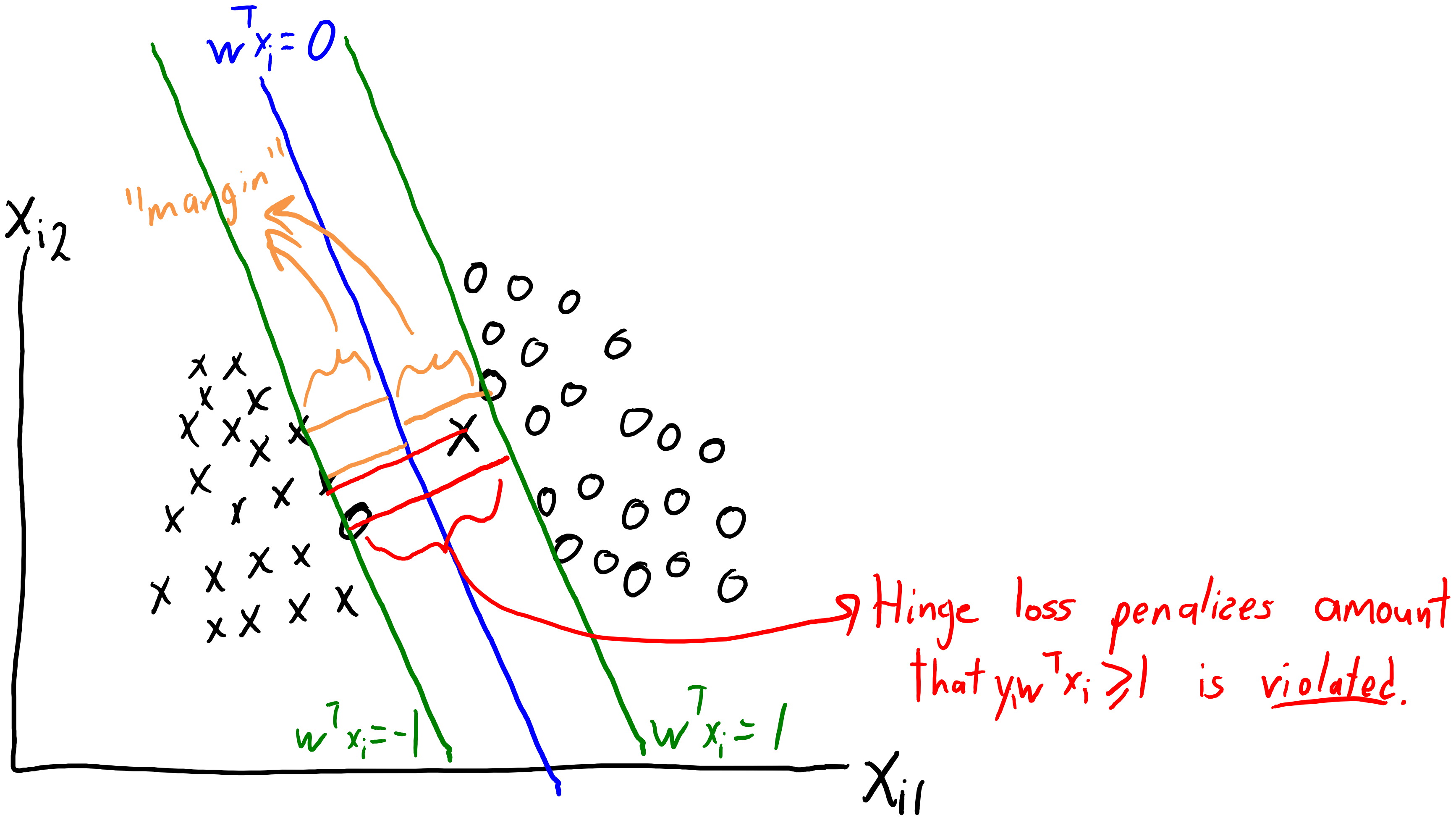
Support Vector Machines for Non-Separable

- Non-separable case:



Support Vector Machines for Non-Separable

- Non-separable case:



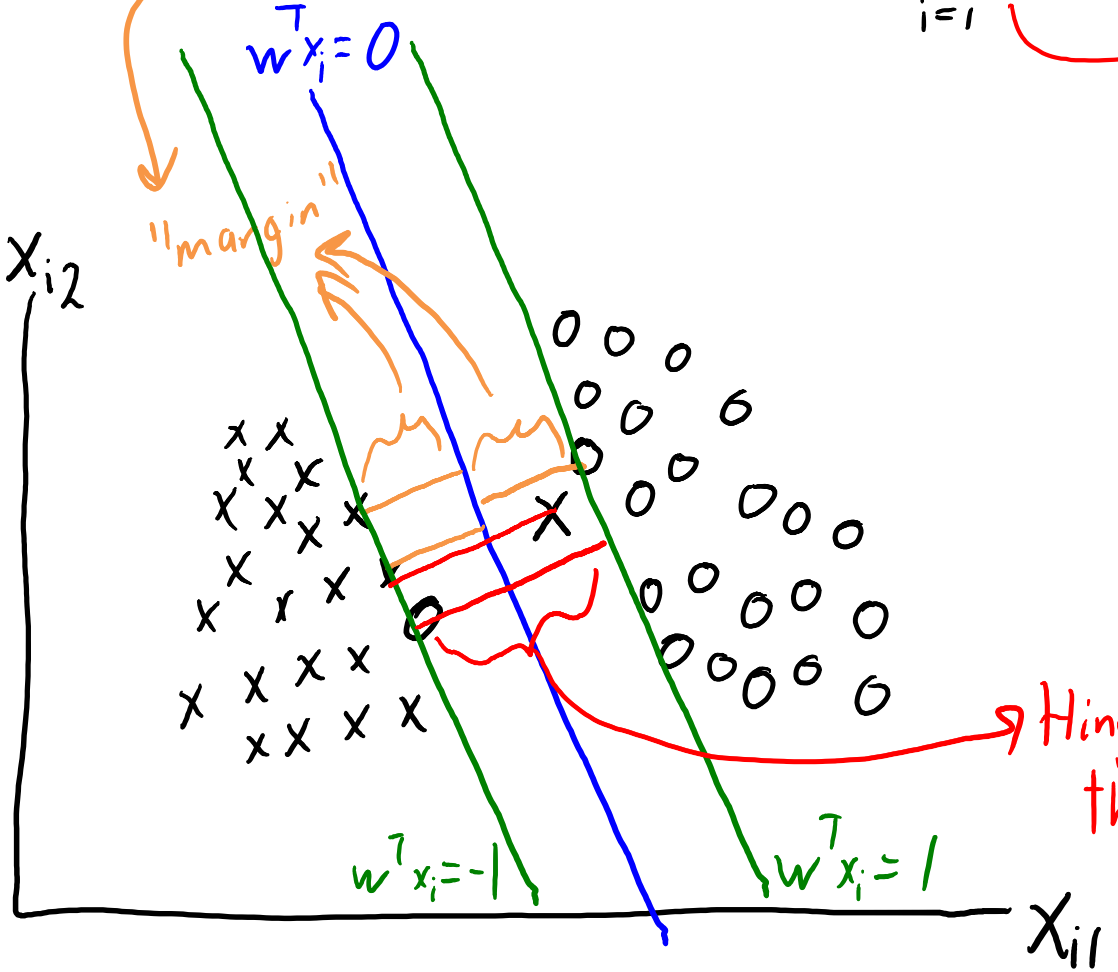
Support Vector Machines for Non-Separable

- Non-separable case:

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

Logistic regression can be viewed as smooth approximation to SVMs.

But, no concept of "support vectors" with logistic loss.



λ controls trade-off between having large margin and classifying examples correctly.

Hinge loss penalizes amount that $y_i w^T x_i \geq 1$ is violated.

Support Vector Machines for Non-Separable

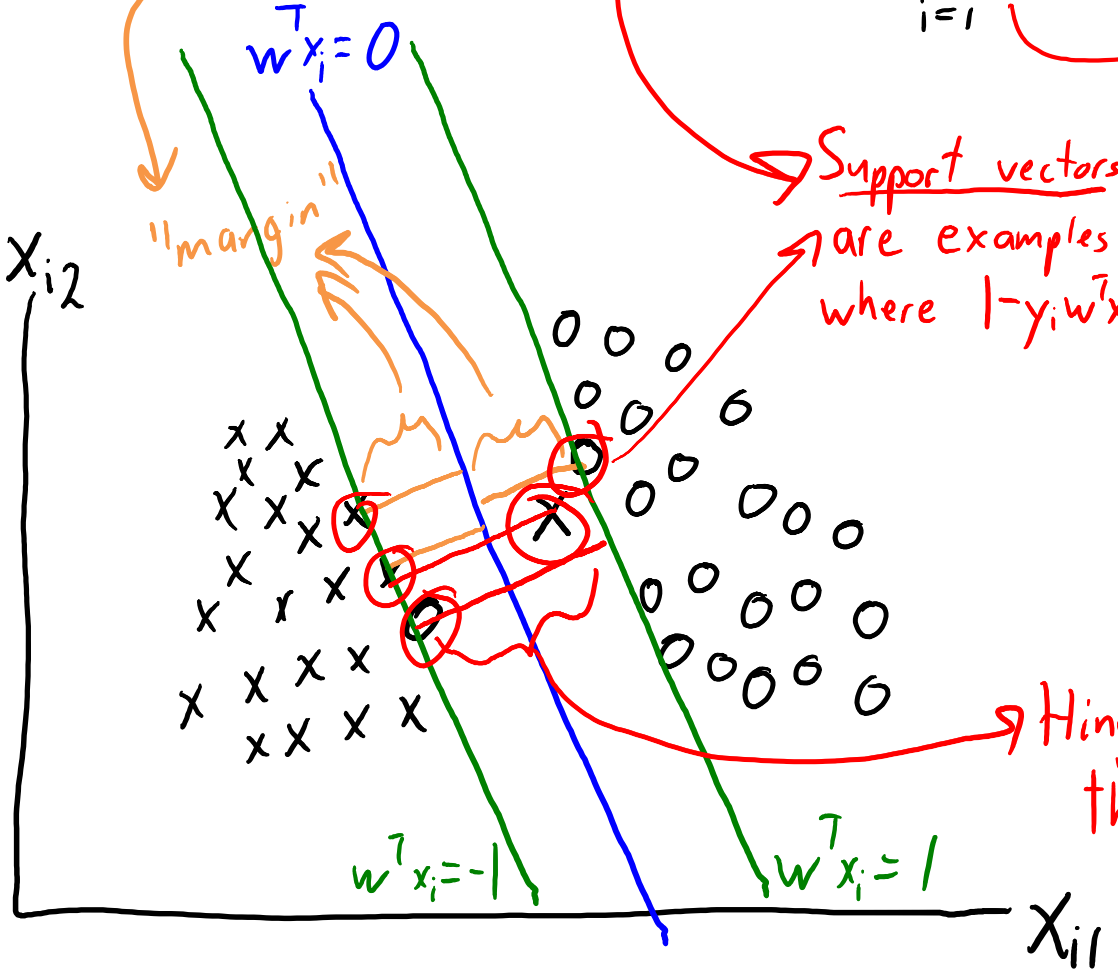
- Non-separable case:

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

Support vectors are examples 'i' where $1 - y_i w^T x_i \geq 0$

Hinge loss penalizes amount that $y_i w^T x_i \geq 1$ is violated.

λ controls trade-off between having large margin and classifying examples correctly.

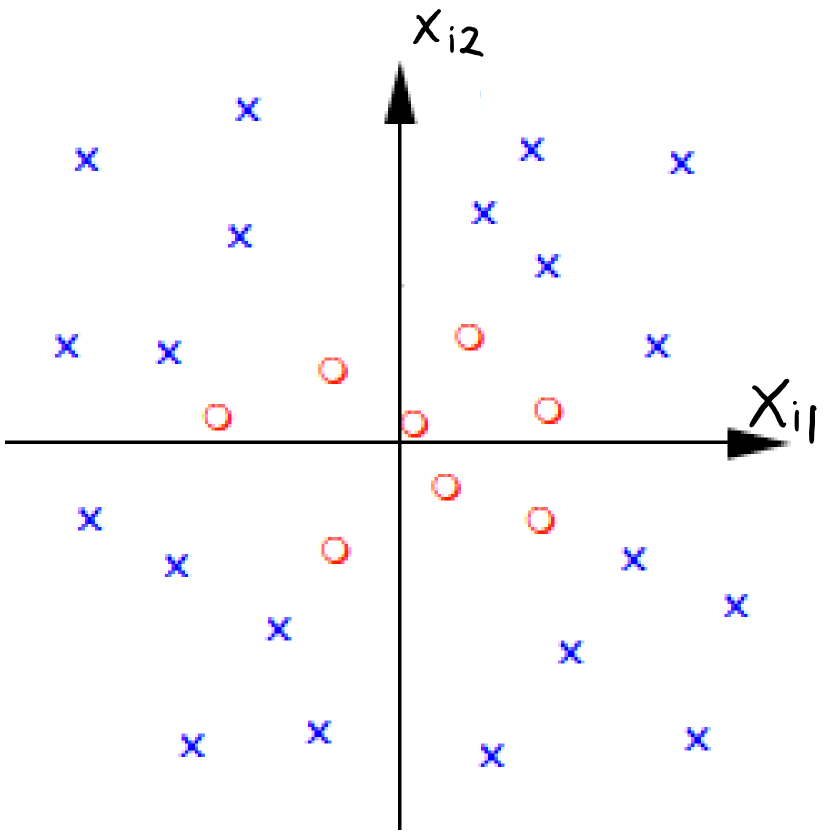


Logistic regression can be viewed as smooth approximation to SVMs.

But, no concept of "support vectors" with logistic loss.

Support Vector Machines for Non-Separable

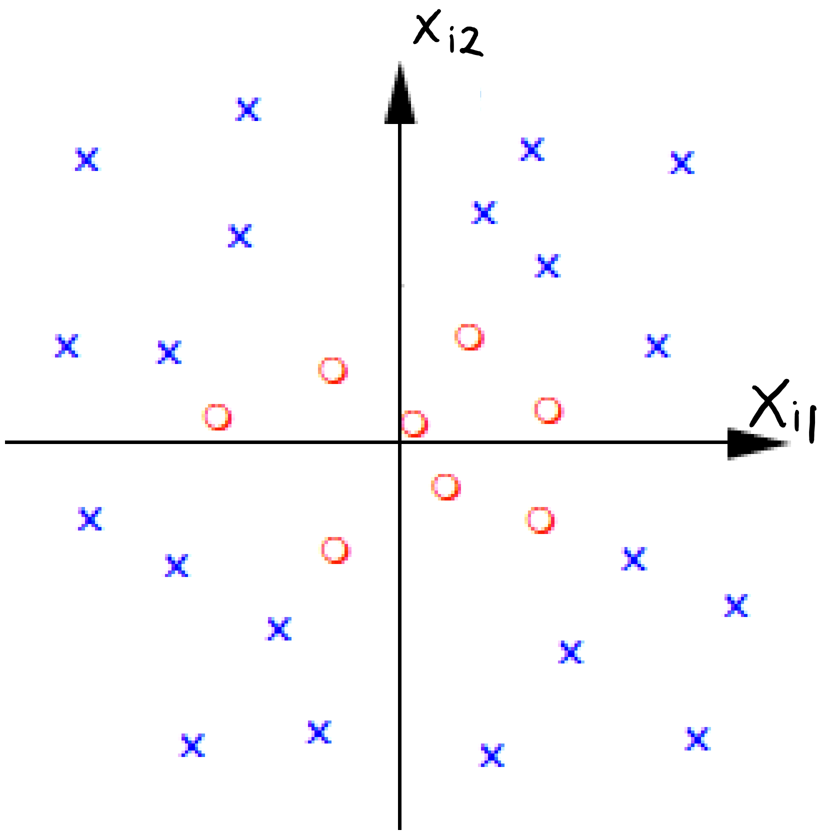
- What about data that is not even close to separable?

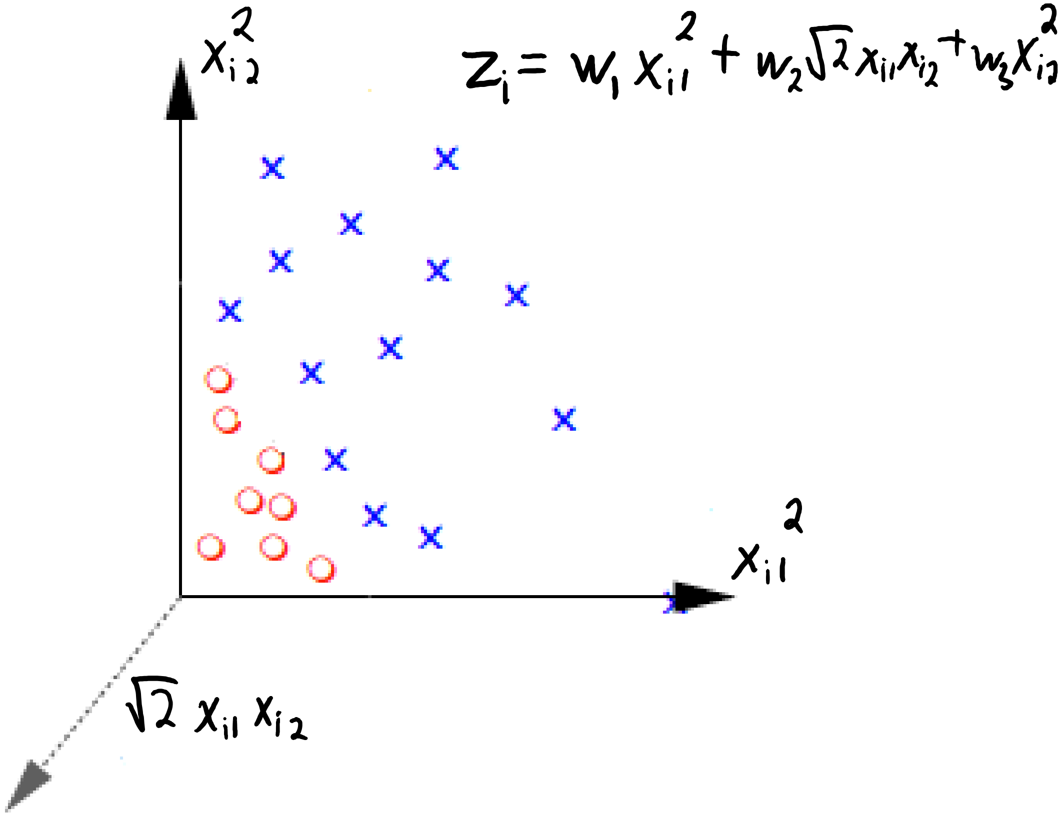


http://math.stackexchange.com/questions/353607/how-do-inner-product-space-determine-half-planes

Support Vector Machines for Non-Separable

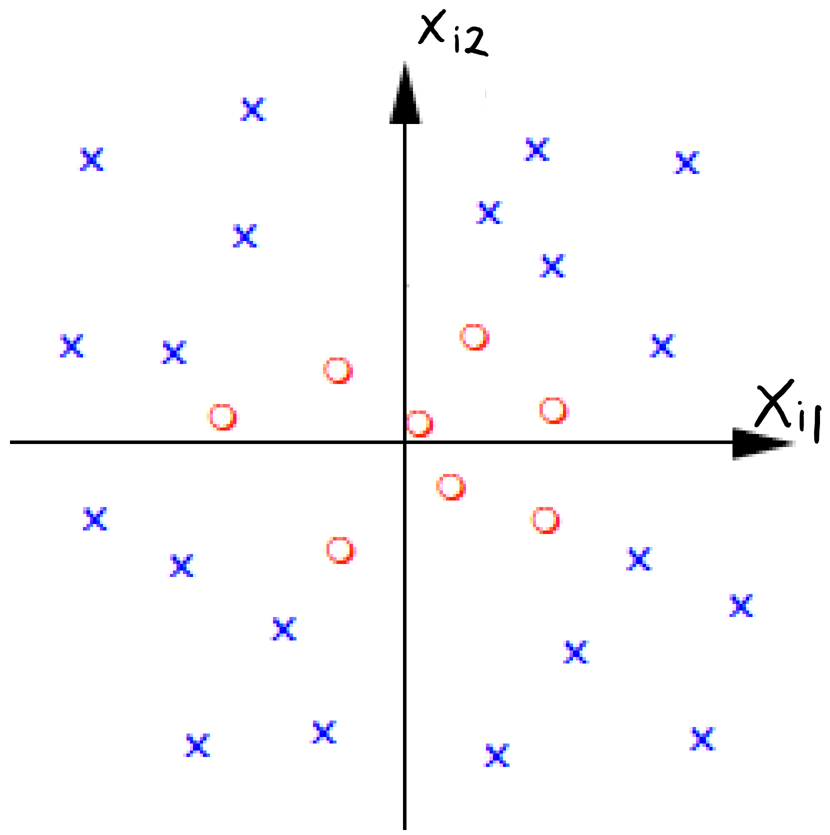
- What about data that is not even close to separable?
  - It may be separable under change of basis (or closer to separable).

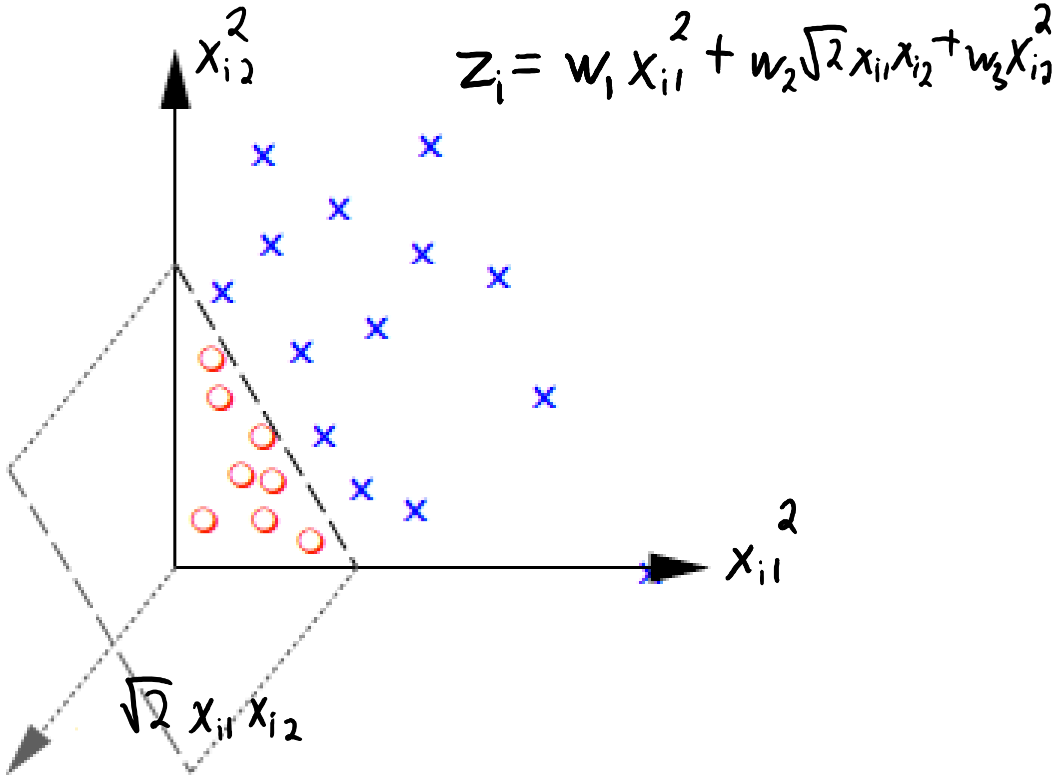




Support Vector Machines for Non-Separable

- What about data that is not even close to separable?
  - It may be separable under change of basis (or closer to separable).





Multi-Dimensional Polynomial Basis

- Recall fitting polynomials when we only have 1 feature:

$$y_i = w_0 + w_1 x_i + w_2 x_i^2$$

- We can fit these models using a change of basis:

$$X = \begin{bmatrix} 0.2 \\ -0.5 \\ 1 \\ 4 \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & 0.2 & (0.2)^2 \\ 1 & -0.5 & (-0.5)^2 \\ 1 & 1 & (1)^2 \\ 1 & 4 & (4)^2 \end{bmatrix}$$

- How can we do this when we have a lot of features?

Multi-Dimensional Polynomial Basis

- Approach 1: use polynomial basis for each variable.

$$X = \begin{bmatrix} 0.2 & 0.3 \\ 1 & 0.5 \\ -0.5 & -0.1 \end{bmatrix} \longrightarrow Z = \begin{bmatrix} \underbrace{\begin{matrix} 1 \\ 1 \\ 1 \end{matrix}}_{\text{bias}} & \underbrace{\begin{matrix} 0.2 & (0.2)^2 \\ 1 & (1)^2 \\ 0.5 & (0.5)^2 \end{matrix}}_{\text{quadratic function of } x_{i1}} & \underbrace{\begin{matrix} 0.3 & (0.3)^2 \\ 0.5 & (0.5)^2 \\ -0.1 & (-0.1)^2 \end{matrix}}_{\text{quadratic function of } x_{i2}} \end{bmatrix}$$

- But this is restrictive:
  - We should allow terms like '$x_{i1}x_{i2}$' that depend on feature interaction.
  - But number of terms in X_{poly} is huge:
    - Degree-5 polynomial basis has $O(d^5)$ terms:

$$x_{i1}^5,\ x_{i1}^4 x_{i2},\ x_{i1}^4 x_{i3}, \dots, x_{i1}^4 x_{id},\ x_{i1}^3 x_{i2}^2,\ x_{i1}^3 x_{i3}^2, \dots, x_{i1}^3 x_{id}^2, \dots, x_{i2}^5,\ x_{i2}^4 x_{i3}, \dots\dots, x_{id}^5$$

- If reasonable 'n', we can do this efficiently using the kernel trick.

Equivalent Form of Ridge Regression

- Recall L2-regularized least squares objective with basis matrix 'Z':

$$f(w) = \frac{1}{2}\|Zw - y\|^2 + \frac{\lambda}{2}\|w\|^2$$

- We showed that the solution is given by:

$$w = \underbrace{(Z^T Z + \lambda I)}_{d \times d}^{-1}(Z^T y)$$

- Using a "matrix inversion lemma" we can re-write this as:

$$w = Z^T \underbrace{(Z Z^T + \lambda I)}_{n \times n}^{-1} y$$

- This is faster if n << d:
  - Z^TZ is 'd' by 'd' while ZZ^T is 'n' by 'n'.

Predictions using Equivalent Form

- Given test data $\hat{X}$, predict $\hat{y}$ by forming and $\hat{Z}$ using:

$$\hat{y} = \hat{Z}w \leftarrow w = Z^T(ZZ^T + \lambda I)^{-1}y$$

$$= \underbrace{\hat{Z}Z^T}_{\hat{K}}(\underbrace{ZZ^T}_{K} + \lambda I)^{-1}y$$

$$= \hat{K}(K + \lambda I)^{-1}y$$

- Key observation behind kernel trick:
  - Predictions $\hat{y}$ only depend on features through K and $\hat{K}$.
  - If we have function that computes K and $\hat{K}$, we don't need the features.

Gram Matrix

- The Gram matrix 'K' is defined by:

$$K = ZZ^T = \underbrace{\begin{bmatrix} \text{---} \; z_1 \; \text{---} \\ \text{---} \; z_2 \; \text{---} \\ \vdots \\ \text{---} \; z_n \; \text{---} \end{bmatrix}}_{Z} \underbrace{\begin{bmatrix} | & | & & | \\ z_1 & z_2 & \cdots & z_n \\ | & | & & | \end{bmatrix}}_{Z^T}$$

$$= \begin{bmatrix} z_1^T z_1 & z_1^T z_2 & \cdots & z_1^T z_n \\ z_2^T z_1 & z_2^T z_2 & \cdots & z_2^T z_n \\ \vdots & & \ddots & \\ z_n^T z_1 & z_n^T z_2 & \cdots & z_n^T z_n \end{bmatrix}$$

- 'K' contains the inner products between all training examples.

Gram Matrix

- The Gram matrix 'K' is defined by:

$$K = ZZ^T = \begin{bmatrix} z_1^T z_1 & z_1^T z_2 & \cdots & z_1^T z_n \\ z_2^T z_1 & z_2^T z_2 & \cdots & z_2^T z_n \\ \vdots & & \ddots & \\ z_n^T z_1 & z_n^T z_2 & \cdots & z_n^T z_n \end{bmatrix}$$

- 'K' contains the inner products between all training examples.
- '$\hat{K}$' contains the inner products between training and test examples.
- Kernel trick:
  - I want to use a basis z_i that is too huge to store.
  - But I only need z_i to compute $K = ZZ^T$ and $\hat{K} = \hat{Z}Z^T$.
  - I can use this basis if I have a kernel function that computes $k(x_i,x_j) = z_i^T z_j$.

Polynomial Kernel

- Consider two examples x_i and x_j for a 2-dimensional dataset:

$$x_i = (x_{i1}, x_{i2}) \qquad x_j = (x_{j1}, x_{j2})$$

- And consider a particular degree-2 basis:

$$z_i = (x_{i1}^2, \sqrt{2}\, x_{i1}x_{i2}, x_{i2}^2) \qquad z_j = (x_{j1}^2, \sqrt{2}\, x_{j1}x_{j2}, x_{j2}^2)$$

- We can compute inner product $z_i^T z_j$ without forming z_i and z_j:

$$z_i^T z_j = x_{i1}^2 x_{j1}^2 + (\sqrt{2}\, x_{i1} x_{i2})(\sqrt{2}\, x_{j1} x_{j2}) + x_{i2}^2 x_{j2}^2$$

$$= x_{i1}^2 x_{j1}^2 + 2 x_{i1} x_{i2} x_{j1} x_{j2} + x_{i1}^2 x_{i2}^2$$

$$= (\underbrace{x_{i1} x_{j1} + x_{i2} x_{j2}}_{x_i^T x_j})^2 \qquad \text{"completing the square"}$$

$$= (x_i^T x_j)^2 \leftarrow \text{No } \underline{\text{need}} \text{ for } z_i \text{ to compute } z_i^T z_j$$

Summary

- Support vector machines maximize margin to nearest data points.
- High-dimensional bases allows us to separate non-separable data.
- Kernel trick allows us to use high-dimensional bases efficiently.

- Next time:
  - How could we train on all of Gmail?