

# Logistic Regression

A decorative graphic consisting of six circles arranged in two rows of three. The top row has a white circle with a light purple outline on the left, and two solid light purple circles on the right. The bottom row has two solid light purple circles on the left, and a white circle with a light purple outline on the right. The text 'Logistic Regression' is centered over the top row of circles.

# Logistic Regression

- Idea:
- Naïve Bayes allows computing  $P(Y|X)$  by learning  $P(Y)$  and  $P(X|Y)$
- Why not learn  $P(Y|X)$  directly?

- Consider learning  $f: X \rightarrow Y$ , where
  - $X$  is a vector of real-valued features,  $\langle X_1 \dots X_n \rangle$
  - $Y$  is boolean
  - assume all  $X_i$  are conditionally independent given  $Y$
  - model  $P(X_i | Y = y_k)$  as Gaussian  $N(\mu_{ik}, \sigma_i)$
  - model  $P(Y)$  as Bernoulli ( $\pi$ )
- What does that imply about the form of  $P(Y|X)$ ?

$$P(Y = 1 | X = \langle X_1, \dots, X_n \rangle) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

# Derive form for $P(Y|X)$ for continuous $X_i$

$$\begin{aligned}
 P(Y=1|X) &= \frac{P(Y=1)P(X|Y=1)}{P(Y=1)P(X|Y=1) + P(Y=0)P(X|Y=0)} \\
 &= \frac{1}{1 + \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}} \quad \text{div by } P(Y=1)P(X|Y=1) \\
 &= \frac{1}{1 + \exp\left(\ln \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}\right)} \\
 &= \frac{1}{1 + \exp\left(\left(\ln \frac{1-\pi}{\pi}\right) + \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)}\right)}
 \end{aligned}$$

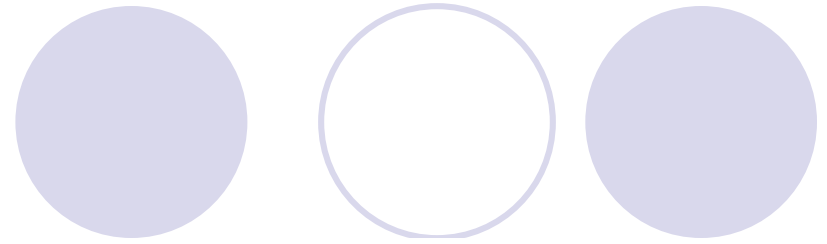
$\pi = P(Y=1)$

$$P(x | y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} e^{-\frac{(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$

$$\sum_i \left( \frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} X_i + \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} \right)$$

$$P(Y=1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)}$$

# Very convenient!



$$P(Y = 1|X = \langle X_1, \dots, X_n \rangle) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

implies

$$P(Y = 0|X = \langle X_1, \dots, X_n \rangle) =$$

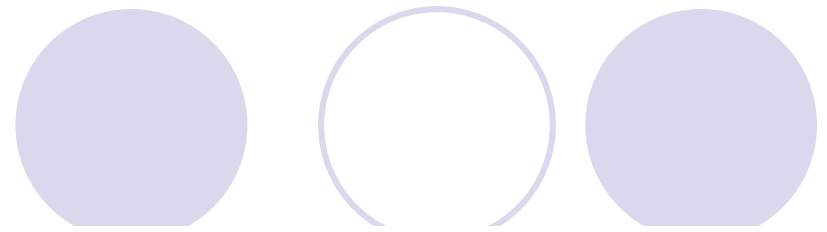
implies

$$\frac{P(Y = 0|X)}{P(Y = 1|X)} =$$

implies

$$\ln \frac{P(Y = 0|X)}{P(Y = 1|X)} =$$

# Very convenient!



$$P(Y = 1|X = \langle X_1, \dots, X_n \rangle) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

implies

$$P(Y = 0|X = \langle X_1, \dots, X_n \rangle) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

implies

$$\frac{P(Y = 0|X)}{P(Y = 1|X)} = \exp(w_0 + \sum_i w_i X_i)$$

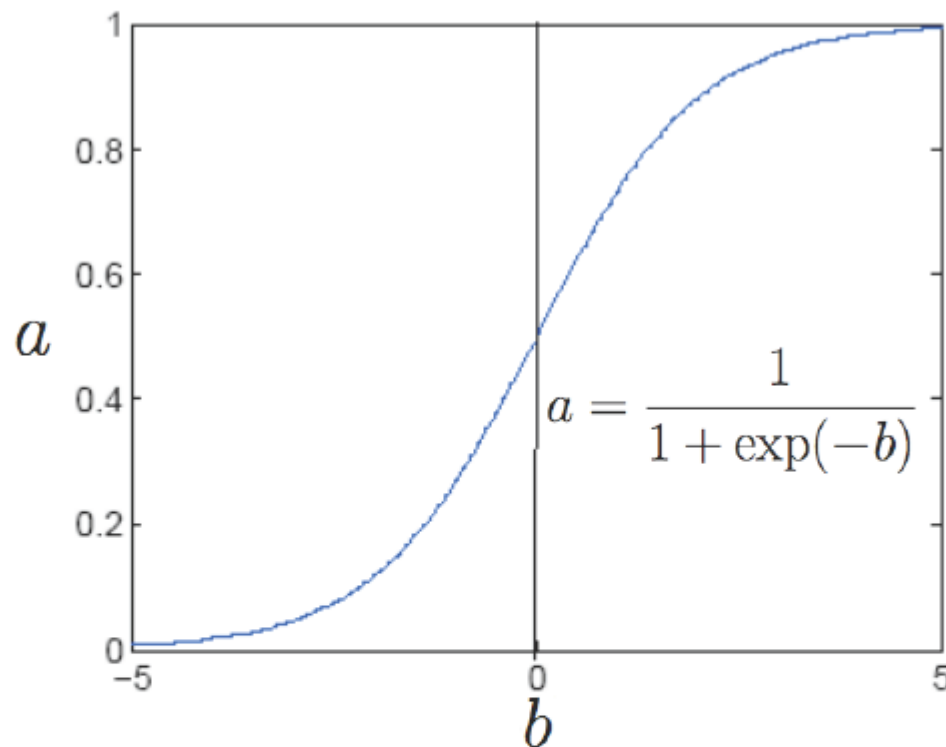
implies

$$\ln \frac{P(Y = 0|X)}{P(Y = 1|X)} = w_0 + \sum_i w_i X_i$$

linear  
classification  
rule!

log linear

# Logistic function



$$P(Y = 1|X) = \frac{1}{1 + \exp(\underbrace{w_0 + \sum_{i=1}^n w_i X_i}_b)}$$

# Logistic regression more generally

- Logistic regression in more general case,  
where  $y \in \{y_1 \dots y_R\}$  : learn  $R-1$  sets of weights

for  $k < R$

$$P(Y = y_k | X) = \frac{\exp(w_{k0} + \sum_{i=1}^n w_{ki} X_i)}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^n w_{ji} X_i)}$$

for  $k=R$

$$P(Y = y_R | X) = \frac{1}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^n w_{ji} X_i)}$$



# Training Logistic Regression: MCLE

- we have L training examples:  $\{\langle X^1, Y^1 \rangle, \dots, \langle X^L, Y^L \rangle\}$

- maximum likelihood estimate for parameters W

$$\begin{aligned} W_{MLE} &= \arg \max_W P(\langle X^1, Y^1 \rangle \dots \langle X^L, Y^L \rangle | W) \\ &= \arg \max_W \prod_l P(\langle X^l, Y^l \rangle | W) \end{aligned}$$

- maximum conditional likelihood estimate

$$MCLE = \arg \max_W \prod_l P(Y^l | X^l w)$$

# Training Logistic Regression: MCLE

- Choose parameters  $W = \langle w_0, \dots, w_n \rangle$  to maximize conditional likelihood of training data

where 
$$P(Y = 0|X, W) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

$$P(Y = 1|X, W) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

- Training data  $D = \{\langle X^1, Y^1 \rangle, \dots, \langle X^L, Y^L \rangle\}$
- Data likelihood =  $\prod_l P(X^l, Y^l|W)$
- Data conditional likelihood =  $\prod_l P(Y^l|X^l, W)$

$$W_{MCLE} = \arg \max_W \prod_l P(Y^l|W, X^l)$$

# Expressing Conditional Log Likelihood

$$\underline{l(W)} \equiv \ln \prod_l \underline{P(Y^l|X^l, W)} = \sum_l \ln P(Y^l|X^l, W)$$

$$P(Y = 0|X, W) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

$$P(Y = 1|X, W) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

$$l(W) = \sum_l Y^l \ln P(Y^l = 1|X^l, W) + (1 - Y^l) \ln P(Y^l = 0|X^l, W)$$

$$= \sum_l Y^l \ln \frac{P(Y^l = 1|X^l, W)}{P(Y^l = 0|X^l, W)} + \ln P(Y^l = 0|X^l, W)$$

$$= \sum_l Y^l (w_0 + \sum_i^n w_i X_i^l) - \ln(1 + \exp(w_0 + \sum_i^n w_i X_i^l))$$

# Maximizing Conditional Log Likelihood

$$P(Y = 0|X, W) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

$$P(Y = 1|X, W) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

$$\begin{aligned} l(W) &\equiv \ln \prod_l P(Y^l | X^l, W) \\ &= \sum_l Y^l (w_0 + \sum_i^n w_i X_i^l) - \ln(1 + \exp(w_0 + \sum_i^n w_i X_i^l)) \end{aligned}$$

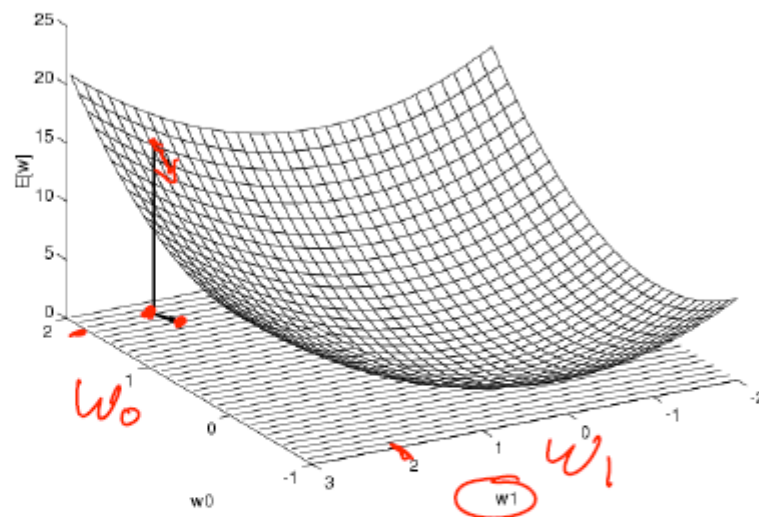
**Good news:**  $l(W)$  is concave function of  $W$

**Bad news:** no closed-form solution to maximize  $l(W)$

# Gradient Descent

---

$$E \leftarrow l(w)$$



Gradient

$$\nabla E[\vec{w}] \equiv \left[ \frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

# Maximize Conditional Log Likelihood: Gradient Ascent

$$\begin{aligned} l(W) &\equiv \ln \prod_l P(Y^l | X^l, W) \\ &= \sum_l Y^l (w_0 + \sum_i^n w_i X_i^l) - \ln(1 + \exp(w_0 + \sum_i^n w_i X_i^l)) \end{aligned}$$

$$\frac{\partial l(W)}{\partial w_i} = \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))$$

Gradient ascent algorithm: iterate until change  $< \varepsilon$

For all  $i$ ,  
repeat

$$w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))$$

# That's all for M(C)LE. How about MAP?

- One common approach is to define priors on  $W$ 
  - Normal distribution, zero mean, identity covariance
- Helps avoid very large weights and overfitting
- MAP estimate

$$W \leftarrow \arg \max_W \ln P(W) \prod_l P(Y^l | X^l, W)$$

- • let's assume Gaussian prior:  $W \sim N(0, \sigma)$

# MLE vs MAP

- Maximum conditional likelihood estimate

$$W \leftarrow \arg \max_W \ln \prod_l P(Y^l | X^l, W)$$

$$w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))$$

- Maximum a posteriori estimate with prior  $W \sim N(0, \sigma I)$

$$W \leftarrow \arg \max_W \ln [P(W) \prod_l \hat{P}(Y^l | X^l, W)]$$

$$w_i \leftarrow w_i - \eta \lambda w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))$$



# MAP estimates and Regularization

- Maximum a posteriori estimate with prior  $W \sim N(0, \sigma I)$

$$W \leftarrow \arg \max_W \ln[P(W) \prod_l P(Y^l | X^l, W)]$$

$$w_i \leftarrow w_i - \eta \lambda w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))$$

called a “regularization” term

- helps reduce overfitting, especially when training data is sparse
- keep weights nearer to zero (if  $P(W)$  is zero mean Gaussian prior), or whatever the prior suggests
- used very frequently in Logistic Regression

# The Bottom Line

- Consider learning  $f: X \rightarrow Y$ , where
  - $X$  is a vector of real-valued features,  $\langle X_1 \dots X_n \rangle$
  - $Y$  is boolean
  - assume all  $X_i$  are conditionally independent given  $Y$
  - model  $P(X_i | Y = y_k)$  as Gaussian  $N(\mu_{ik}, \sigma_i)$
  - model  $P(Y)$  as Bernoulli ( $\pi$ )
- Then  $P(Y|X)$  is of this form, and we can directly estimate  $W$

$$P(Y = 1 | X = \langle X_1, \dots, X_n \rangle) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

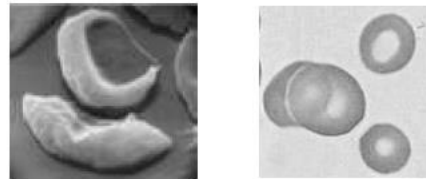
- Furthermore, same holds if the  $X_i$  are boolean
  - trying proving that to yourself

# Classification Tasks

## Features, X

## Labels, Y

Diagnosing sickle cell anemia



Anemic cell  
Healthy cell

Tax Fraud Detection

Refund	Marital Status	Taxable Income
No	Married	80K



Cheat
?

Web Classification



Sports  
Science  
News

Predict squirrel hill resident

Drive to CMU, Rachel's fan,  
Shop at SH Giant Eagle



Resident  
Not resident

# Classification

**Goal:** Construct a **predictor**  $f : X \rightarrow Y$  to minimize a risk (performance measure)  $R(f)$



**Features,  $X$**



Sports  
Science  
News

**Labels,  $Y$**

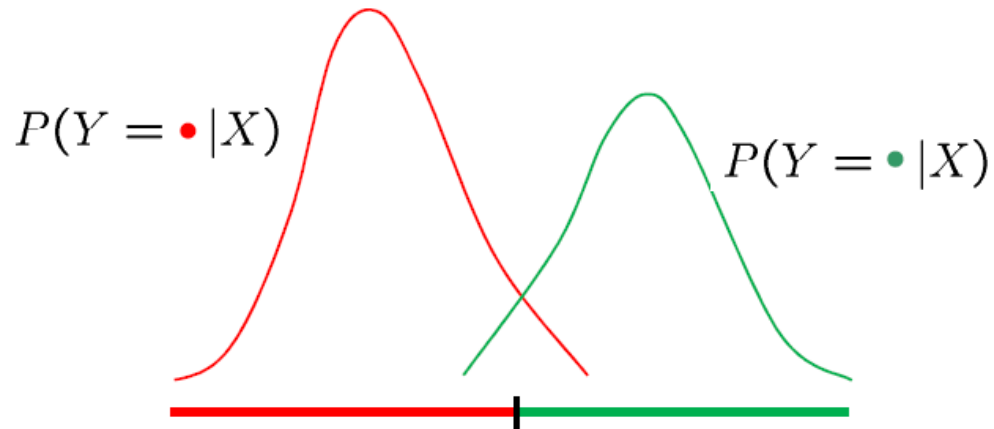
$$R(f) = P(f(X) \neq Y)$$

**Probability of Error**

# Classification

Optimal predictor:  
(Bayes classifier)

$$f^* = \arg \min_f P(f(X) \neq Y)$$



$$f^*(X) = \begin{cases} \bullet & P(Y = \bullet | X) > P(Y = \bullet | X) \\ \bullet & \text{otherwise} \end{cases}$$

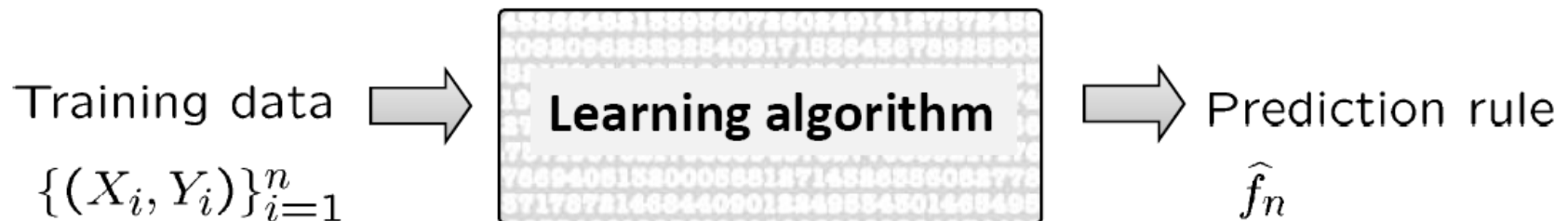
Depends on **unknown** distribution  $P_{XY}$

# Classification algorithms

However, we can **learn** a good prediction rule from **training data**

$$\{(X_i, Y_i)\}_{i=1}^n \stackrel{\text{iid}}{\sim} P_{XY}(\text{unknown})$$

Independent and identically distributed

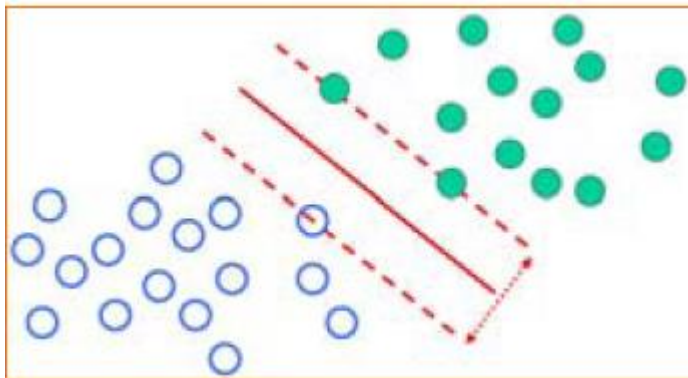


**So far ...**

Decision Trees  
K-Nearest Neighbor  
Naïve Bayes  
Logistic Regression

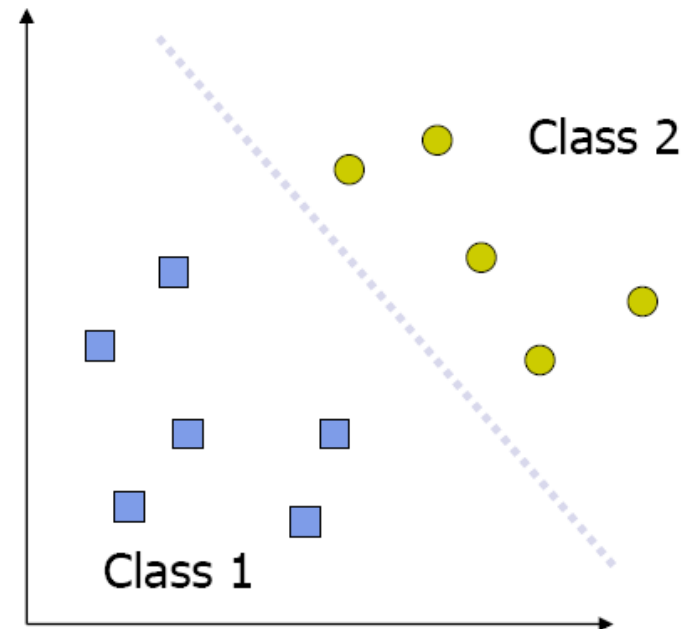
# Machine Learning

## Support Vector Machines



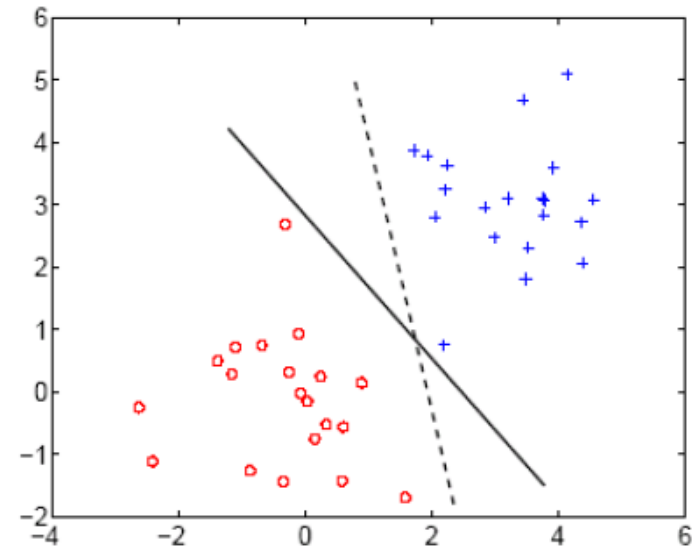
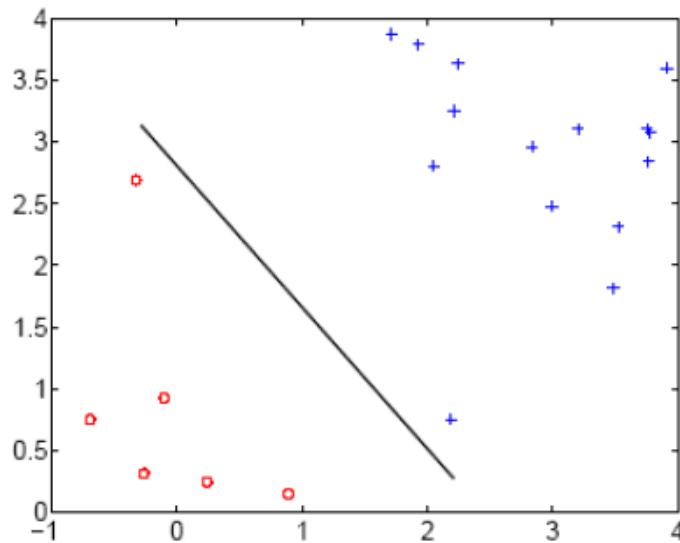
# What is a good Decision Boundary?

- Consider a binary classification task with  $y = \pm 1$  labels (not 0/1 as before).
- When the training examples are linearly separable, we can set the parameters of a linear classifier so that all the training examples are classified correctly
- Many decision boundaries!
  - Generative classifiers
  - Logistic regressions ...
- Are all decision boundaries equally good?





# Not All Decision Boundaries Are Equal!



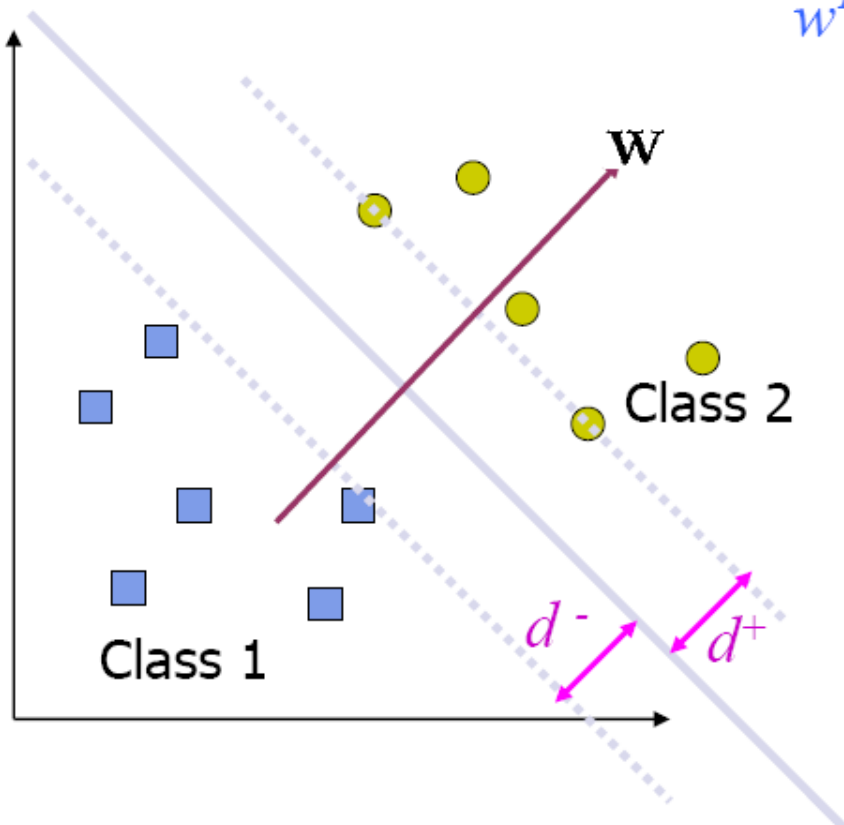
- Why we may have such boundaries?
  - Irregular distribution
  - Imbalanced training sizes
  - outliers

# Classification and Margin

- Parameterizing decision boundary

- Let  $w$  denote a vector orthogonal to the decision boundary, and  $b$  denote a scalar "offset" term, then we can write the decision boundary as:

$$w^T x + b = 0$$



# Classification and Margin

## Parameterizing decision boundary

- Let  $w$  denote a vector orthogonal to the decision boundary, and  $b$  denote a scalar "offset" term, then we can write the decision boundary as:

$$w^T x + b = 0$$

### Margin

$$w^T x_i + b > +c \quad \text{for all } x_i \text{ in class 2}$$

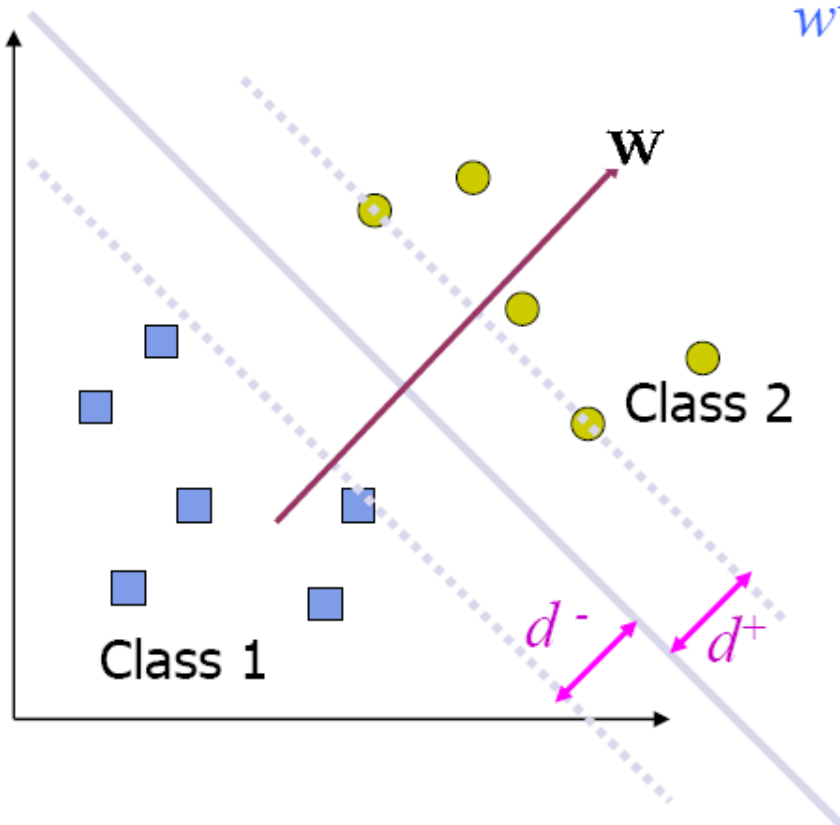
$$w^T x_i + b < -c \quad \text{for all } x_i \text{ in class 1}$$

Or more compactly:

$$(w^T x_i + b) y_i > c$$

The margin between any two points

$$m = d^- + d^+ =$$



# Maximum Margin Classification

- The "minimum" permissible margin is:

$$m = \frac{w^T}{\|w\|} (x_{i^*} - x_{j^*}) = \frac{2c}{\|w\|}$$

- Here is our Maximum Margin Classification problem:

$$\begin{array}{ll} \max_w & \frac{2c}{\|w\|} \\ \text{s.t} & y_i (w^T x_i + b) \geq c, \quad \forall i \end{array}$$

# Maximum Margin Classification, con'd.

- The optimization problem:

$$\begin{aligned} \max_{w,b} \quad & \frac{c}{\|w\|} \\ \text{s.t} \quad & y_i(w^T x_i + b) \geq c, \quad \forall i \end{aligned}$$

- But note that the magnitude of **c** merely scales **w** and **b**, and does not change the classification boundary at all! (why?)
- So we instead work on this cleaner problem:

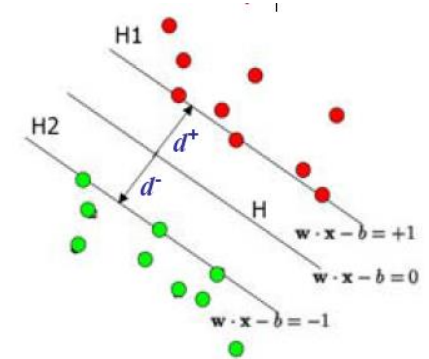
$$\begin{aligned} \max_{w,b} \quad & \frac{1}{\|w\|} \\ \text{s.t} \quad & y_i(w^T x_i + b) \geq 1, \quad \forall i \end{aligned}$$

- The solution to this leads to the famous **Support Vector Machines** --- believed by many to be the best "off-the-shelf" supervised learning algorithm

# Support vector machine

- A convex quadratic programming problem with linear constraints:

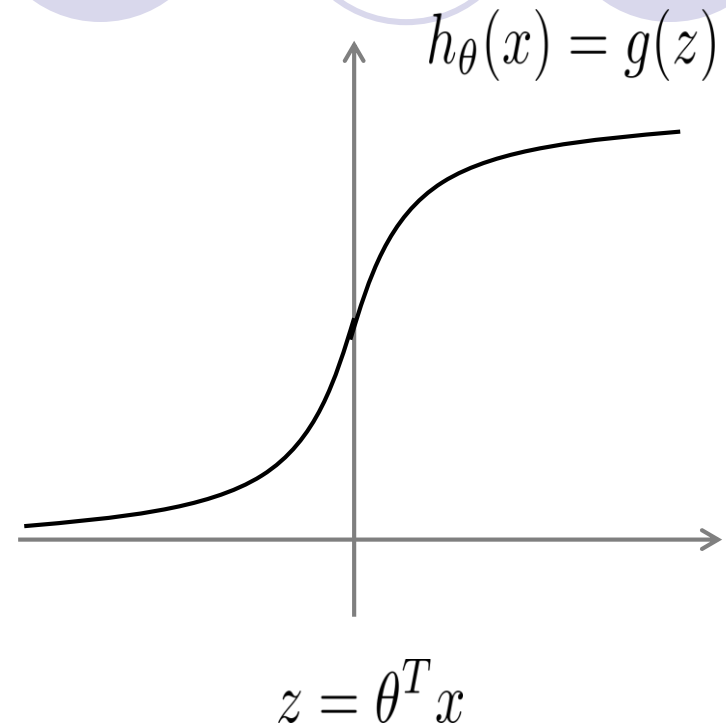
$$\begin{aligned} & \max_{w,b} \quad \frac{1}{\|w\|} \\ & \text{s.t.} \quad y_i(w^T x_i + b) \geq 1, \quad \forall i \end{aligned}$$



- The attained margin is now given by  $\frac{1}{\|w\|}$
- Only a few of the classification constraints are relevant → **support vectors**
- Constrained optimization
  - We can directly solve this using commercial quadratic programming (QP) code
  - But we want to take a more careful investigation of Lagrange duality, and the solution of the above in its dual form.
  - deeper insight: support vectors, kernels ...
  - more efficient algorithm

# Alternative view of logistic regression

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$



If  $y = 1$  , we want  $h_{\theta}(x) \approx 1$  ,  $\theta^T x \gg 0$

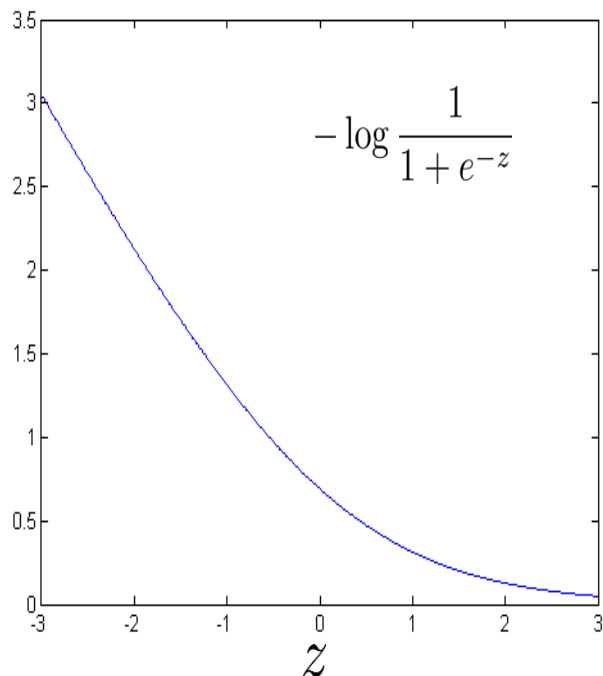
If  $y = 0$  , we want  $h_{\theta}(x) \approx 0$  ,  $\theta^T x \ll 0$

## Alternative view of logistic regression

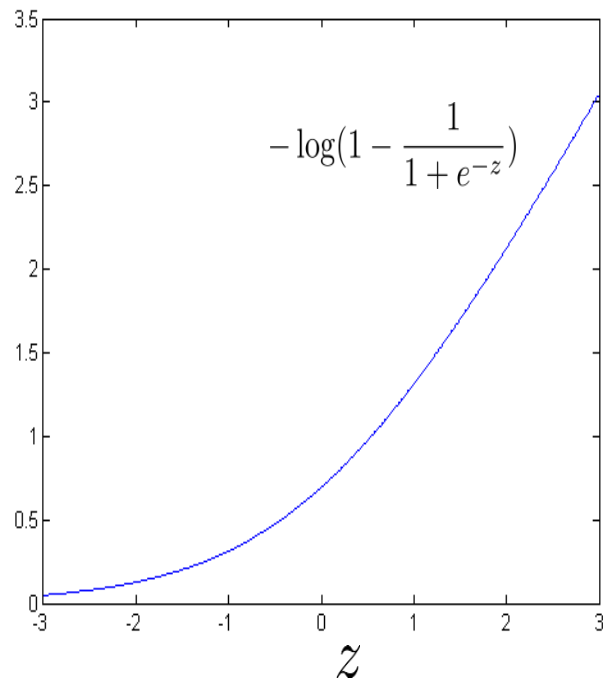
Cost of example:  $-(y \log h_{\theta}(x) + (1 - y) \log(1 - h_{\theta}(x)))$

$$= -y \log \frac{1}{1 + e^{-\theta^T x}} - (1 - y) \log\left(1 - \frac{1}{1 + e^{-\theta^T x}}\right)$$

If  $y = 1$  (want  $\theta^T x \gg 0$ ):



If  $y = 0$  (want  $\theta^T x \ll 0$ ):





# Support vector machine

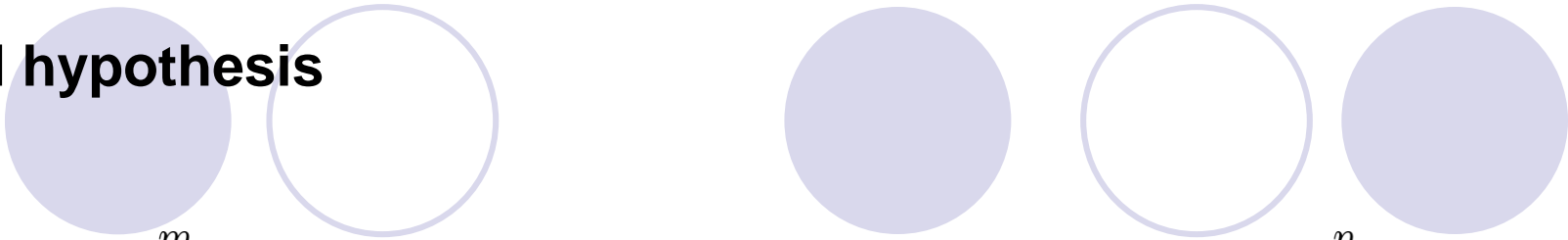
Logistic regression:

$$\min_{\theta} \frac{1}{m} \left[ \sum_{i=1}^m y^{(i)} \left( -\log h_{\theta}(x^{(i)}) \right) + (1 - y^{(i)}) \left( -\log(1 - h_{\theta}(x^{(i)})) \right) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

Support vector machine:

$$\min_{\theta} C \sum_{i=1}^m \left[ y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

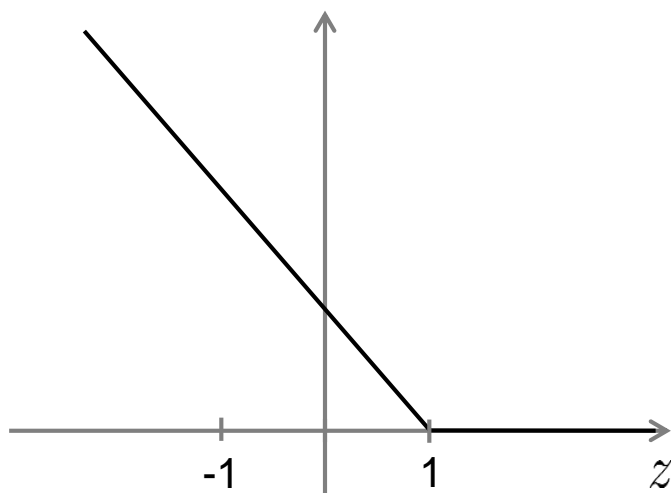
# SVM hypothesis


$$\min_{\theta} C \sum_{i=1}^m \left[ y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

Hypothesis:

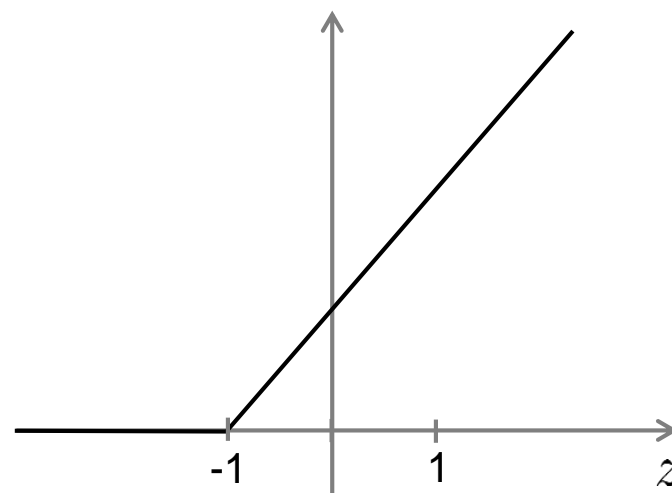
# Support Vector Machine

$$\min_{\theta} C \sum_{i=1}^m \left[ y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^n \theta_j^2$$

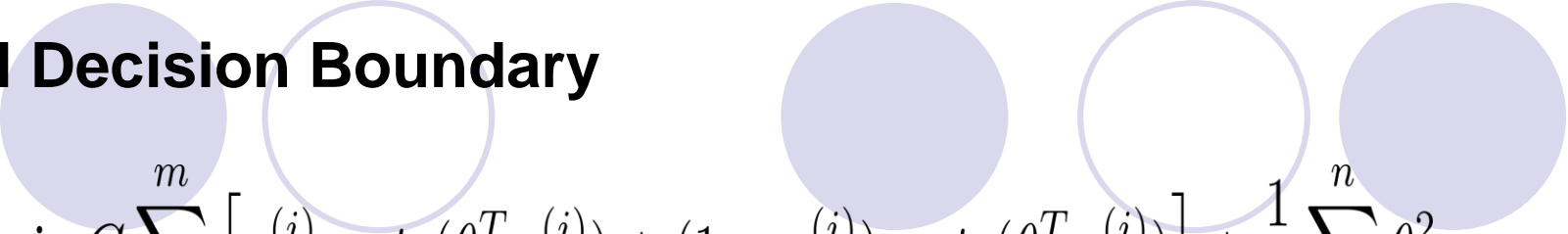


If  $y = 1$  , we want  $\theta^T x \geq 1$  (not just  $\geq 0$ )

If  $y = 0$  , we want  $\theta^T x \leq -1$  (not just  $< 0$ )



# SVM Decision Boundary


$$\min_{\theta} C \sum_{i=1}^m \left[ y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

Whenever  $y^{(i)} = 1$  :

Whenever  $y^{(i)} = 0$  :

# Digression to Lagrangian Duality

- The Primal Problem

Primal:

$$\begin{aligned} \min_w \quad & f(w) \\ \text{s.t.} \quad & g_i(w) \leq 0, \quad i = 1, \dots, k \\ & h_i(w) = 0, \quad i = 1, \dots, l \end{aligned}$$

The generalized Lagrangian:

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^k \alpha_i g_i(w) + \sum_{i=1}^l \beta_i h_i(w)$$

the  $\alpha$ 's ( $\alpha_i \geq 0$ ) and  $\beta$ 's are called the Lagrangian multipliers

Lemma:

$$\max_{\alpha, \beta, \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta) = \begin{cases} f(w) & \text{if } w \text{ satisfies primal constraints} \\ \infty & \text{o/w} \end{cases}$$

A re-written Primal:

$$\min_w \max_{\alpha, \beta, \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

# Lagrangian Duality, cont.

- Recall the Primal Problem:

$$\min_w \max_{\alpha, \beta, \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

- The Dual Problem:

$$\max_{\alpha, \beta, \alpha_i \geq 0} \min_w \mathcal{L}(w, \alpha, \beta)$$

- Theorem (weak duality):

$$d^* = \max_{\alpha, \beta, \alpha_i \geq 0} \min_w \mathcal{L}(w, \alpha, \beta) \leq \min_w \max_{\alpha, \beta, \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta) = p^*$$

- Theorem (strong duality):

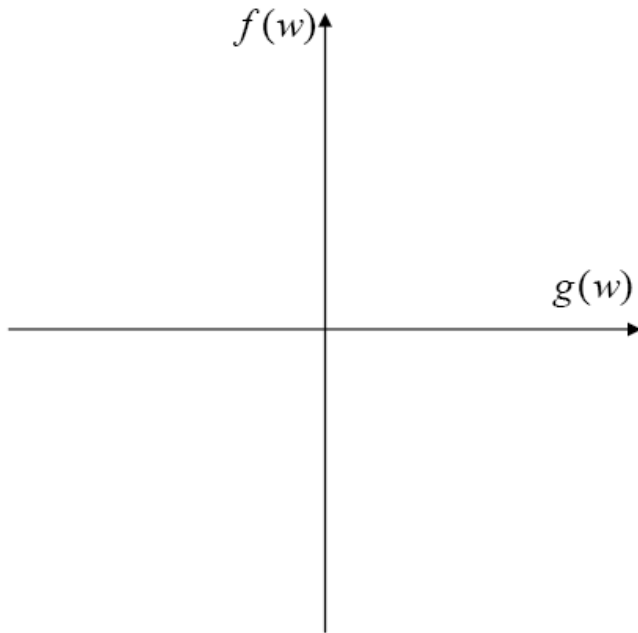
Iff there exist a saddle point of  $\mathcal{L}(w, \alpha, \beta)$ , we have

$$d^* = p^*$$

# A sketch of strong and weak duality

- Now, ignoring  $h(x)$  for simplicity, let's look at what's happening graphically in the duality theorems.

$$d^* = \max_{\alpha_i \geq 0} \min_w f(w) + \alpha^T g(w) \leq \min_w \max_{\alpha_i \geq 0} f(w) + \alpha^T g(w) = p^*$$



# The KKT conditions

- If there exists some saddle point of  $\mathcal{L}$ , then the saddle point satisfies the following "Karush-Kuhn-Tucker" (KKT) conditions:

$$\frac{\partial}{\partial w_i} \mathcal{L}(w, \alpha, \beta) = 0, \quad i = 1, \dots, k$$

$$\frac{\partial}{\partial \beta_i} \mathcal{L}(w, \alpha, \beta) = 0, \quad i = 1, \dots, l$$

$$\alpha_i g_i(w) = 0, \quad i = 1, \dots, m$$

Complementary slackness

$$g_i(w) \leq 0, \quad i = 1, \dots, m$$

Primal feasibility

$$\alpha_i \geq 0, \quad i = 1, \dots, m$$

Dual feasibility

- **Theorem:** If  $w^*$ ,  $\alpha^*$  and  $\beta^*$  satisfy the KKT condition, then it is also a solution to the primal and the dual problems.



# Solving optimal margin classifier

- Recall our opt problem:

$$\begin{array}{ll}\max_{w,b} & \frac{1}{\|w\|} \\ \text{s.t} & y_i(w^T x_i + b) \geq 1, \quad \forall i\end{array}$$

- This is equivalent to

$$\begin{array}{ll}\min_{w,b} & \frac{1}{2} w^T w \\ \text{s.t} & 1 - y_i(w^T x_i + b) \leq 0, \quad \forall i\end{array} \quad (*)$$

- Write the Lagrangian:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^m \alpha_i [y_i(w^T x_i + b) - 1]$$

- Recall that (\*) can be reformulated as  $\min_{w,b} \max_{\alpha_i \geq 0} \mathcal{L}(w, b, \alpha)$   
Now we solve its **dual problem**:  $\max_{\alpha_i \geq 0} \min_{w,b} \mathcal{L}(w, b, \alpha)$

# The Dual Problem

$$\max_{\alpha_i \geq 0} \min_{w, b} \mathcal{L}(w, b, \alpha)$$

- We minimize  $\mathcal{L}$  with respect to  $w$  and  $b$  first:

$$\nabla_w \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^m \alpha_i y_i x_i = 0, \quad (*)$$

$$\nabla_b \mathcal{L}(w, b, \alpha) = \sum_{i=1}^m \alpha_i y_i = 0, \quad (**)$$

Note that (\*) implies:

$$w = \sum_{i=1}^m \alpha_i y_i x_i \quad (***)$$

- Plus (\*\*\*) back to  $\mathcal{L}$ , and using (\*\*), we have:

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

# The Dual problem, cont.

- Now we have the following dual opt problem:

$$\max_{\alpha} \mathcal{J}(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t. } \alpha_i \geq 0, \quad i = 1, \dots, k$$

$$\sum_{i=1}^m \alpha_i y_i = 0.$$

- This is, (again,) a **quadratic programming** problem.

- A global maximum of  $\alpha_i$  can always be found.
- But what's the big deal??
- Note two things:

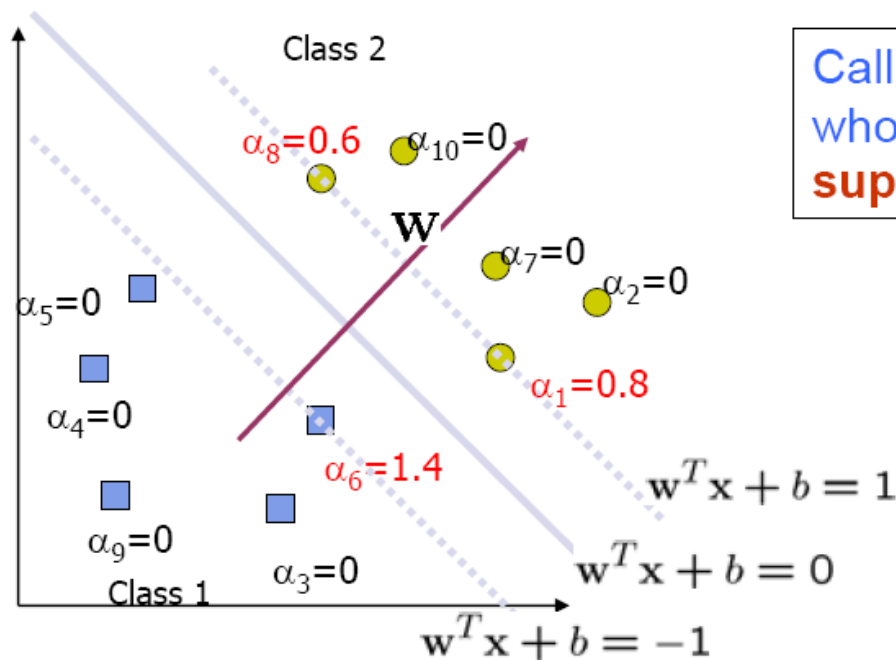
1.  $w$  can be recovered by  $w = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$  See next ...

2. The "kernel"  $\mathbf{x}_i^T \mathbf{x}_j$  More later ...

# Support vectors

- Note the KKT condition --- only a few  $\alpha_i$ 's can be nonzero!!

$$\alpha_i g_i(w) = 0, \quad i = 1, \dots, m$$



Call the training data points whose  $\alpha_i$ 's are nonzero the **support vectors (SV)**

# Support vector machines

- Once we have the Lagrange multipliers  $\{\alpha_i\}$ , we can reconstruct the parameter vector  $w$  as a weighted combination of the training examples:

$$w = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i$$

- For testing with a new data  $z$

- Compute

$$w^T z + b = \sum_{i \in SV} \alpha_i y_i (\mathbf{x}_i^T z) + b$$

and classify  $z$  as class 1 if the sum is positive, and class 2 otherwise

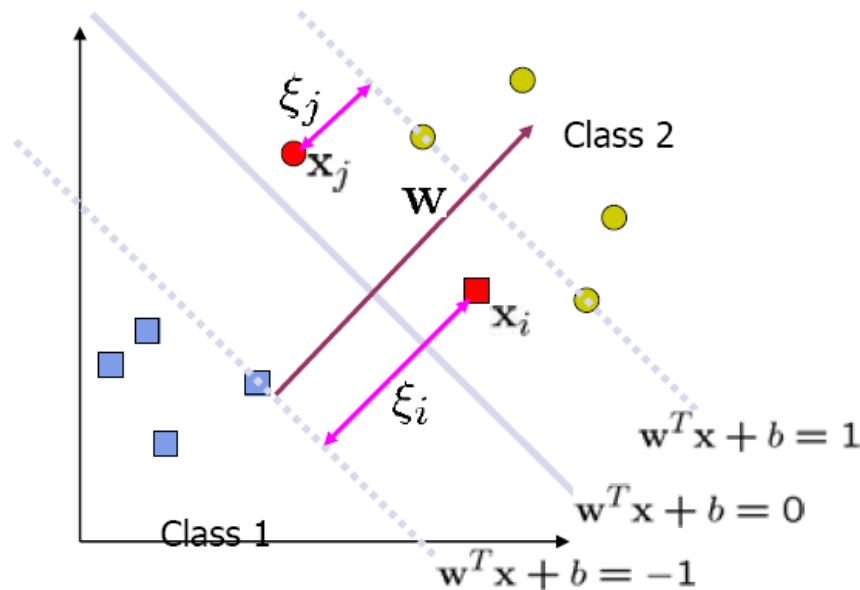
- Note:  $w$  need not be formed explicitly

# Interpretation of support vector machines

- The optimal  $\mathbf{w}$  is a linear combination of a small number of data points. This “sparse” representation can be viewed as data compression as in the construction of kNN classifier
- To compute the weights  $\{\alpha_i\}$ , and to use support vector machines we need to specify only the inner products (or kernel) between the examples  $\mathbf{x}_i^T \mathbf{x}_j$
- We make decisions by comparing each new example  $\mathbf{z}$  with only the support vectors:

$$y^* = \text{sign} \left( \sum_{i \in SV} \alpha_i y_i (\mathbf{x}_i^T \mathbf{z}) + b \right)$$

# Non-linearly Separable Problems



- We allow “error”  $\xi_i$  in classification; it is based on the output of the discriminant function  $\mathbf{w}^T \mathbf{x} + b$
- $\xi_i$  approximates the number of misclassified samples

# Soft Margin Hyperplane

- Now we have a slightly different opt problem:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^m \xi_i \\ \text{s.t} \quad & y_i (w^T x_i + b) \geq 1 - \xi_i, \quad \forall i \\ & \xi_i \geq 0, \quad \forall i \end{aligned}$$

- $\xi_i$  are “slack variables” in optimization
- Note that  $\xi_i=0$  if there is no error for  $\mathbf{x}_i$
- $\xi_i$  is an upper bound of the number of errors
- $C$  : tradeoff parameter between error and margin



# The Optimization Problem

- The dual of this new constrained optimization problem is

$$\begin{aligned} \max_{\alpha} \quad & \mathcal{J}(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, m \\ & \sum_{i=1}^m \alpha_i y_i = 0. \end{aligned}$$

- This is very similar to the optimization problem in the linear separable case, except that there is an upper bound  $C$  on  $\alpha_i$  now
- Once again, a QP solver can be used to find  $\alpha_i$

# Non-linear Decision Boundary

- So far, we have only considered large-margin classifier with a linear decision boundary
- How to generalize it to become nonlinear?
- Key idea: transform  $\mathbf{x}_i$  to a higher dimensional space to “make life easier”
  - Input space: the space the point  $\mathbf{x}_i$  are located
  - Feature space: the space of  $\phi(\mathbf{x}_i)$  after transformation
- Why transform?
  - Linear operation in the feature space is equivalent to non-linear operation in input space
  - Classification can become easier with a proper transformation. In the XOR problem, for example, adding a new feature of  $x_1x_2$  make the problem linearly separable

# The XOR problem

A classic problem is the XOR problem. The training data for this is:

$$\left\{ \left\{ \begin{bmatrix} 1.0 \\ -1.0 \end{bmatrix}, 1 \right\}, \left\{ \begin{bmatrix} -1.0 \\ 1.0 \end{bmatrix}, 1 \right\}, \left\{ \begin{bmatrix} 1.0 \\ 1.0 \end{bmatrix}, -1 \right\}, \left\{ \begin{bmatrix} -1.0 \\ -1.0 \end{bmatrix}, -1 \right\} \right\}$$

This data is clearly not separable with a linear decision boundary. Consider the following mapping:

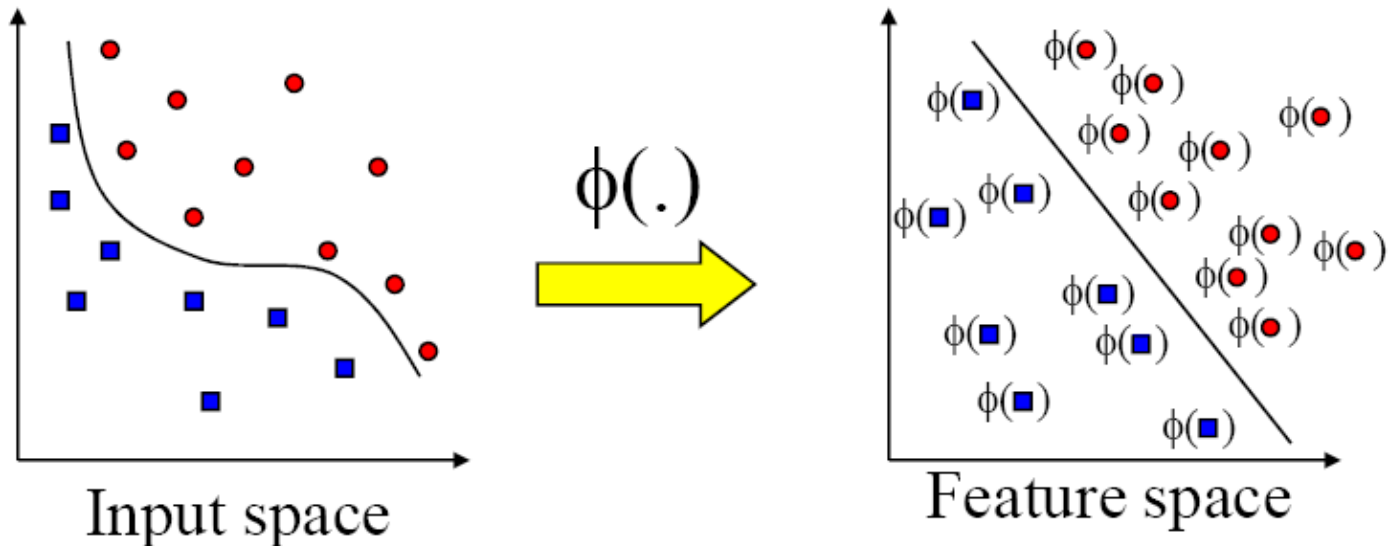
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \Phi(\mathbf{x}) = \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \sqrt{2}x_1x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix}$$

Each point is now mapped from a 2-dimensional space to a 5-dimensional space. The data is separable in this high dimensional space.

# Non-linear Decision Boundary



# Transforming the Data



Note: feature space is of higher dimension than the input space in practice

# The Kernel Trick

- Recall the SVM optimization problem

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \dots, m$$

$$\sum_{i=1}^m \alpha_i y_i = 0.$$

- The data points only appear as **inner product**
- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations (angles, distances) can be expressed by inner products
- Define the kernel function  $K$  by  $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

# An Example for feature mapping and kernels

- Consider an input  $\mathbf{x}=[x_1, x_2]$
- Suppose  $\phi(\cdot)$  is given as follows

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = 1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2$$

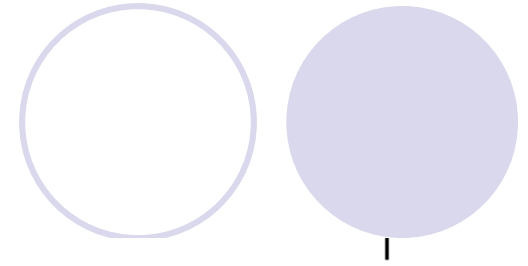
- An inner product in the feature space is

$$\left\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} x_1' \\ x_2' \end{bmatrix}\right) \right\rangle =$$

- So, if we define the **kernel function** as follows, there is no need to carry out  $\phi(\cdot)$  explicitly

$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{1} + \mathbf{x}^T \mathbf{x}')^2$$

# More examples of kernel functions



- Linear kernel (we've seen it)

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

- Polynomial kernel (we just saw an example)

$$K(\mathbf{x}, \mathbf{x}') = \left(1 + \mathbf{x}^T \mathbf{x}'\right)^p$$

where  $p = 2, 3, \dots$  To get the feature vectors we concatenate all  $p$ th order polynomial terms of the components of  $\mathbf{x}$  (weighted appropriately)

- Radial basis kernel

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2\right)$$

In this case the feature space consists of functions and results in a non-parametric classifier.



# The essence of kernel

- Feature mapping, but “without paying a cost”

- E.g., polynomial kernel

$$K(x, z) = (x^T z + c)^d$$

- How many dimensions we've got in the new space?
- How many operations it takes to compute  $K()$ ?

- Kernel design, any principle?

- $K(x, z)$  can be thought of as a similarity function between  $x$  and  $z$
- This intuition can be well reflected in the following “Gaussian” function  
(Similarly one can easily come up with other  $K()$  in the same spirit)

$$K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$$

- Is this necessarily lead to a “legal” kernel?

(in the above particular case,  $K()$  is a legal one, do you know how many dimension  $\phi(x)$  is?)