Support Vector Machines

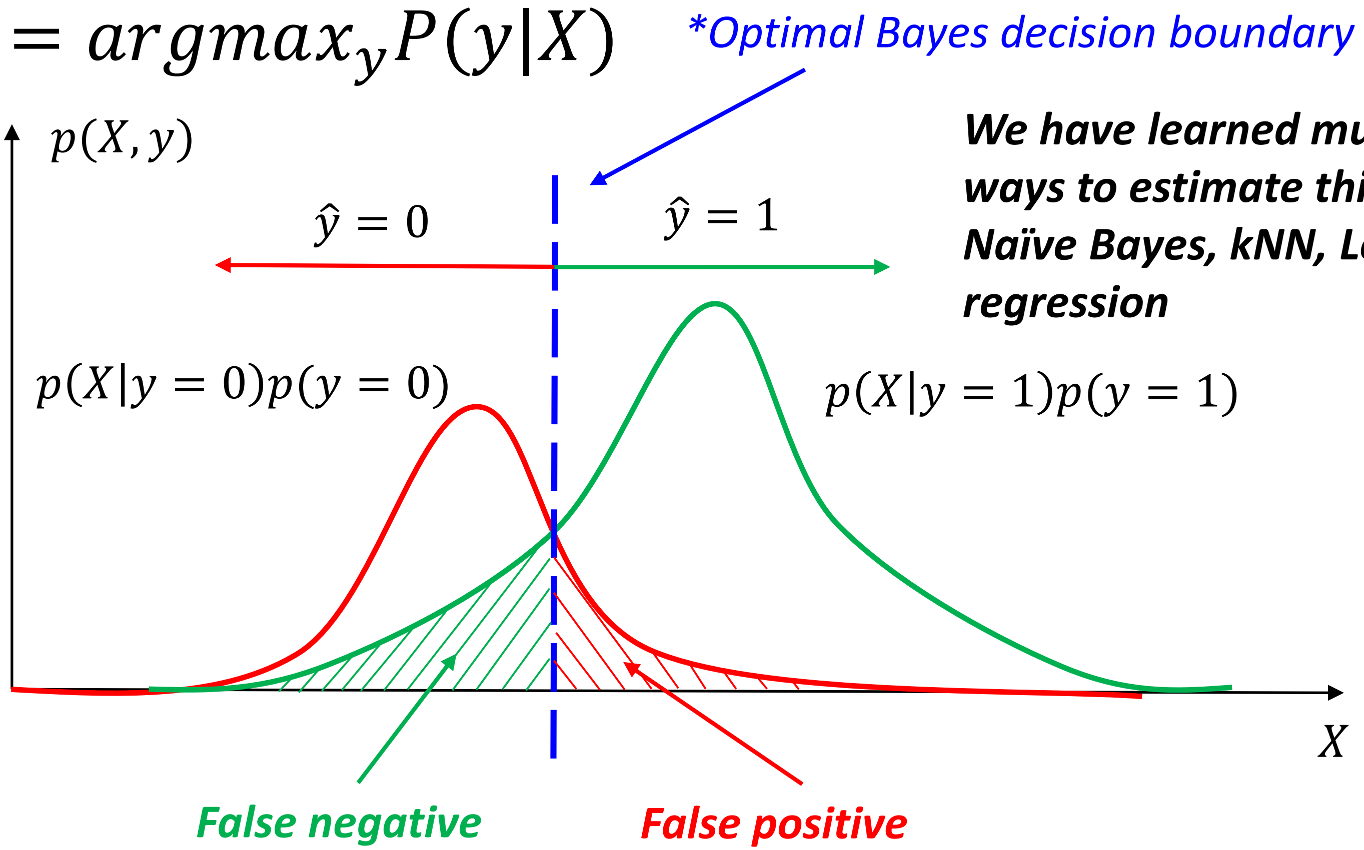
Hongning Wang
CS@UVa

Today's lecture

- Support vector machines
  - Max margin classifier
  - Derivation of linear SVM
    - Binary and multi-class cases
  - Different types of losses in discriminative models
  - Kernel method
    - Non-linear SVM
  - Popular implementations

Review: Bayes risk minimization

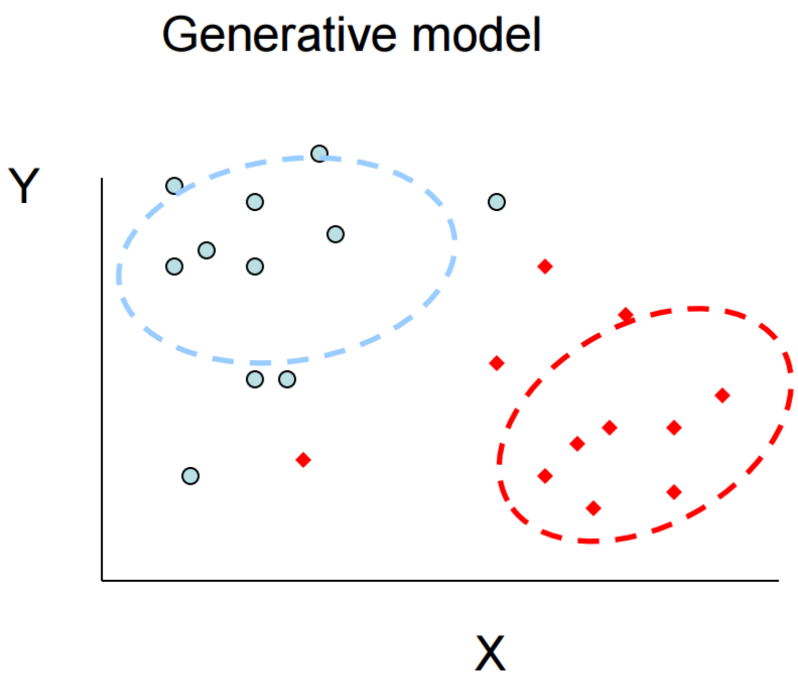
- Risk – assign instance to a wrong class
  - $y^* = argmax_y P(y|X)$

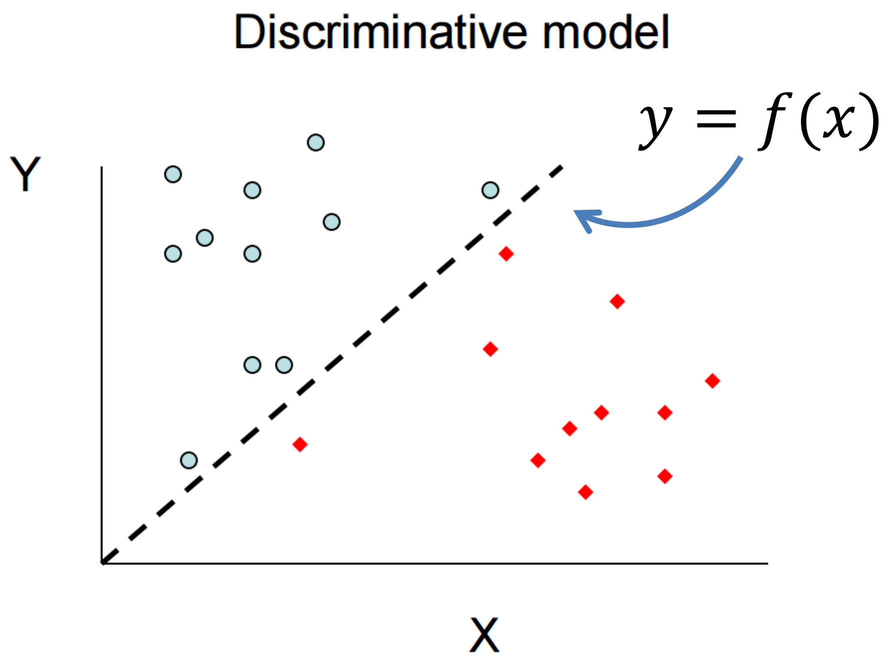


We have learned multiple ways to estimate this, e.g. Naïve Bayes, kNN, Logistic regression

Discriminative v.s. generative models

All instances are considered for probability density estimation





More attention will be put onto the boundary points

Logistic regression for classification

- Decision boundary for binary case

$$- \hat{y} = \begin{cases} 1, p(y=1|X) > 0.5 \\ 0, \quad otherwise \end{cases}$$

$$p(y=1|X) = \frac{1}{1+\exp(-w^T X)} > 0.5$$

i.f.f.

$$\exp(-w^T X) < 1$$

i.f.f.

$$w^T X > 0$$

$$- \hat{y} = \begin{cases} 1, \quad w^T x > 0 \\ 0, otherwise \end{cases}$$

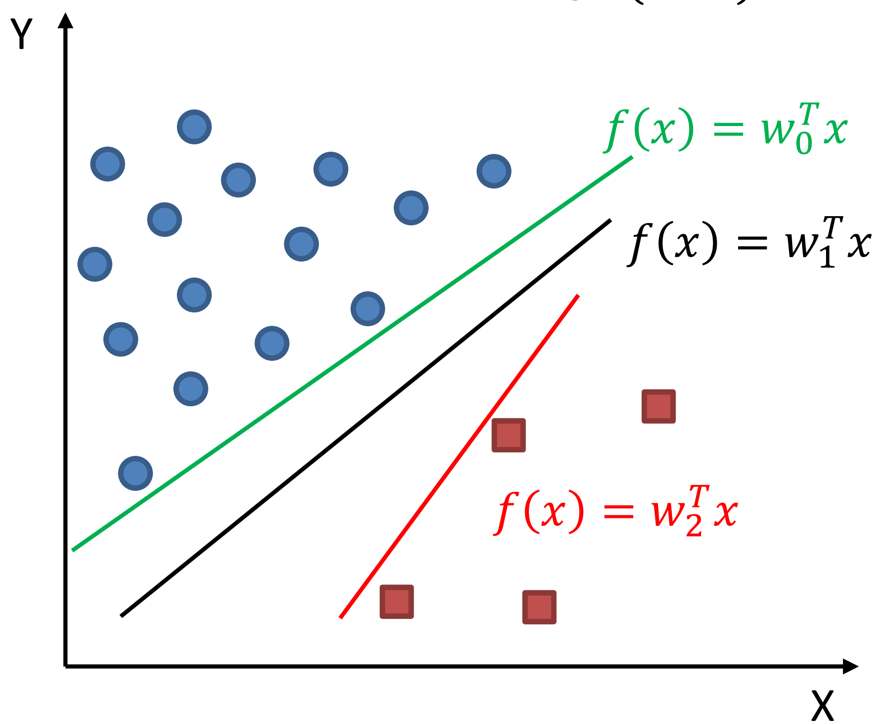
A linear model!

How about directly estimating this?

Which linear classifier do we prefer?

- Choose the one with maximum separation margin

Linear classifier $y = sgn(w^T x)$



Instances are linearly separable

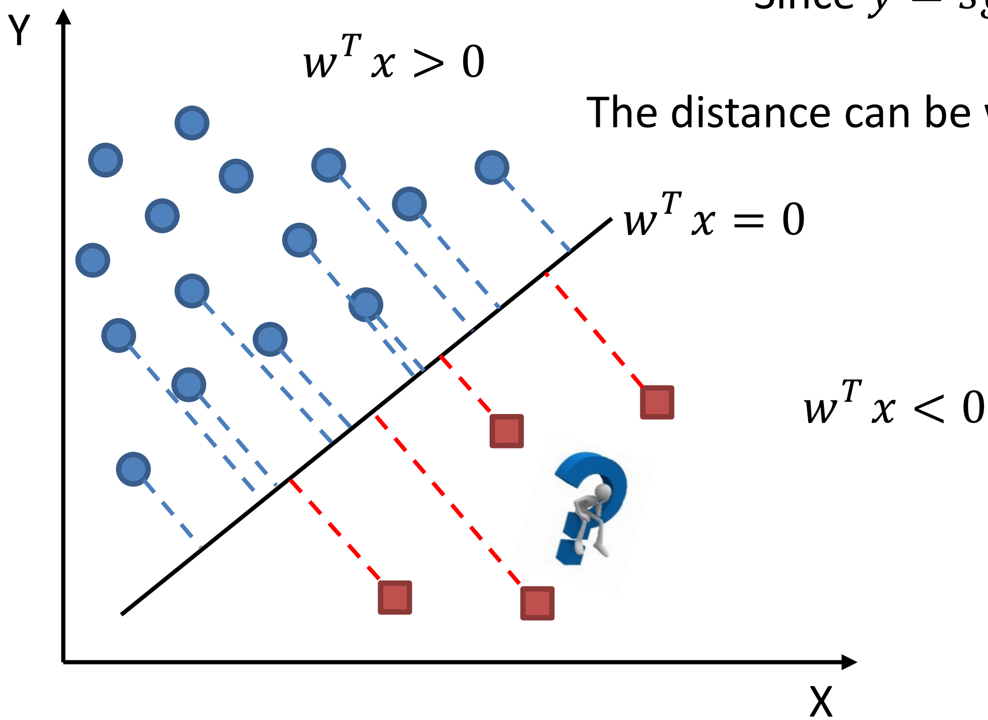
Parameterize the margin

- Margin = $\min_i \frac{y_i w^T x_i}{\sqrt{w^T w}}$

Distance from a point to a line $\frac{|w^T x|}{\sqrt{w^T w}}$

Since $y = sgn(w^T x)$

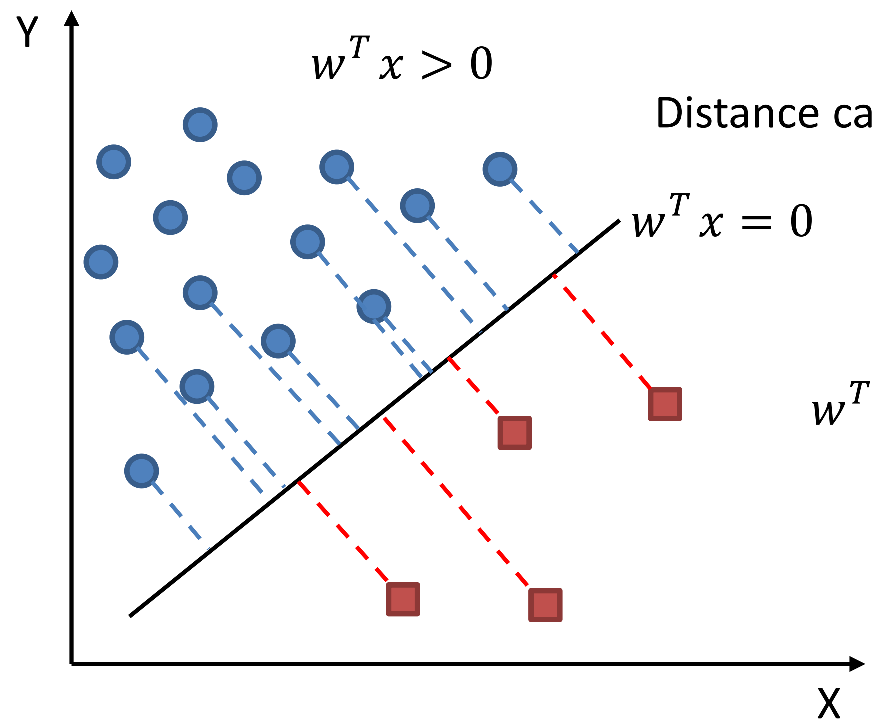
The distance can be written as $\frac{y w^T x}{\sqrt{w^T w}}$



Max margin classifier

- $w^* = \underset{w}{\operatorname{argmax}} \min_i \frac{y_i w^T x_i}{\sqrt{w^T w}} \quad s.t. \forall i, y_i w^T x_i \geq 0$

Distance can be written as $\frac{y w^T x}{\sqrt{w^T w}}$



Max margin classifier

- $\underset{w}{\operatorname{argmax}} \min_{i} \frac{y_i w^T x_i}{\sqrt{w^T w}}$ is difficult to be optimized in general
  - Insight: $\frac{y_i w^T x_i}{\sqrt{w^T w}}$ is invariant to scaling of w
  - Define $y_i w^T x_i = 1$ for the point that is closest to the surface
  - Then, $\forall i, y_i w^T x_i \geq 1$

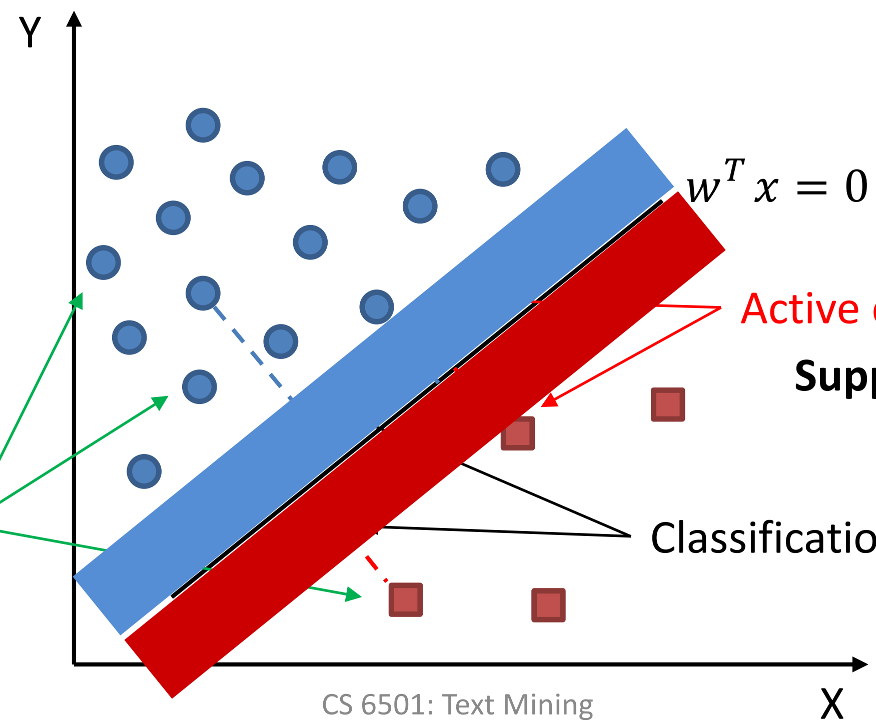
Max margin classifier

$$argmax_w \frac{1}{\sqrt{w^T w}}$$

$$argmin_w w^T w$$

Quadratic programming!
Easy to solve!

$$s.t.\ \forall i, y_i w^T x_i \geq 1$$



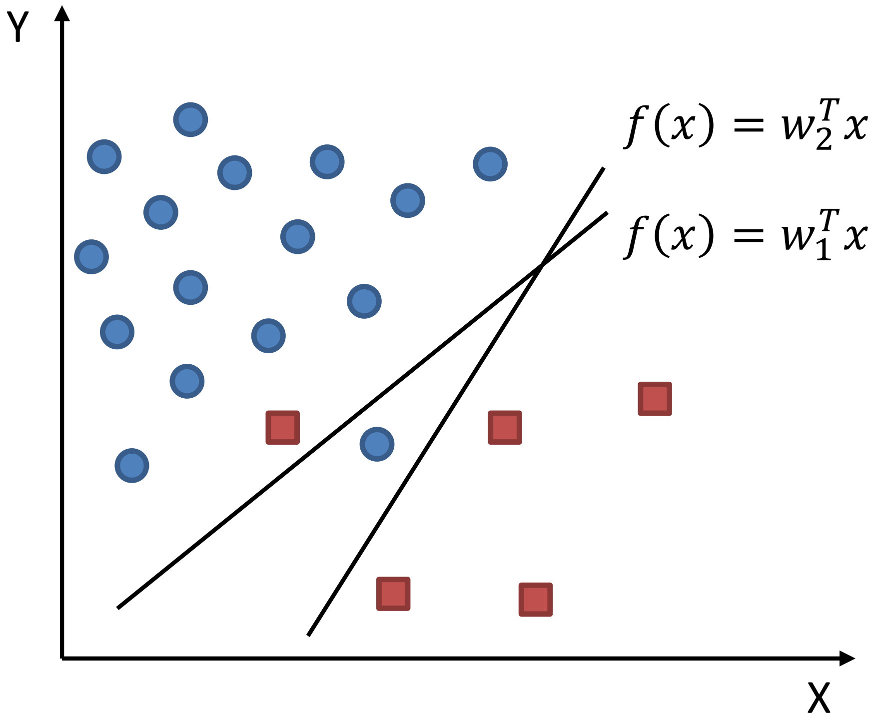
That's why it is called Support Vector Machines

No contribution in deciding the decision boundary

What if the instances are not linearly separable?

- Maximize the margin while minimizing the number of errors made by the classifier?

Linear classifier $y = sgn(w^T x)$



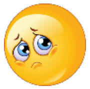
$f(x) = w_2^T x$

$f(x) = w_1^T x$

trade-off parameter manually set

$$argmin_w w^T w + C\#error$$

$$s.t.\ \forall i, y_i w^T x_i \geq 1$$

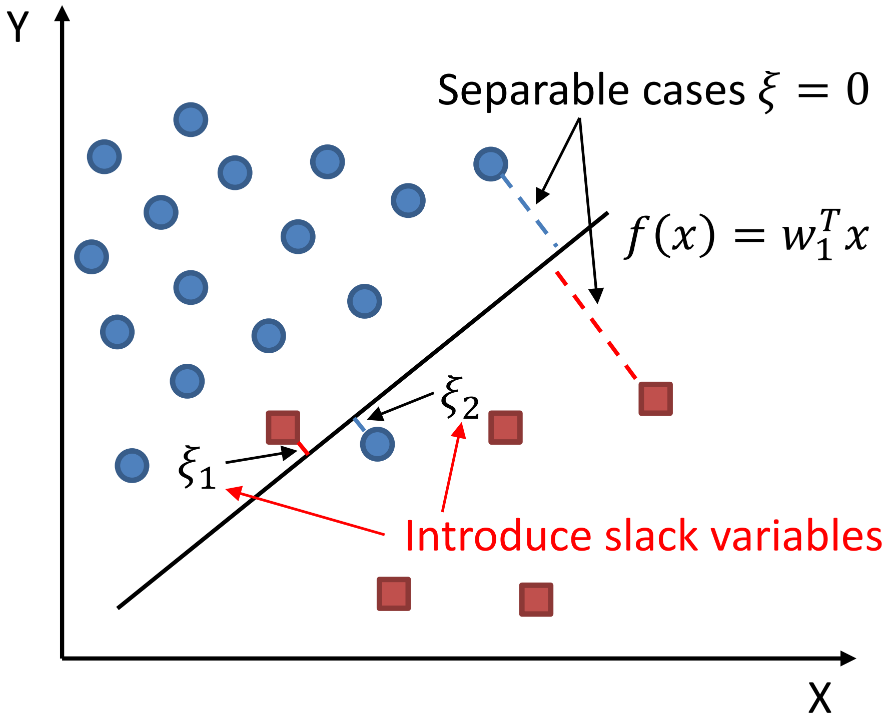


No longer a QP!
No idea how to optimize!

Soft-margin SVM

- Relax the constraints and penalize the misclassification error

Linear classifier $y = sgn(w^T x)$



$$argmin_{w,\xi} w^T w + C \sum_i \xi_i$$
$$s.t.\ \forall i, y_i w^T x_i \geq 1 - \xi_i$$
$$\xi_i \geq 0$$

Still a QP!
Easy to optimize!

What kind of loss is SVM optimizing?

$$argmin_{w,\xi} w^T w + C \sum_i \xi_i$$

$$s.t.\ \forall i, y_i w^T x_i \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

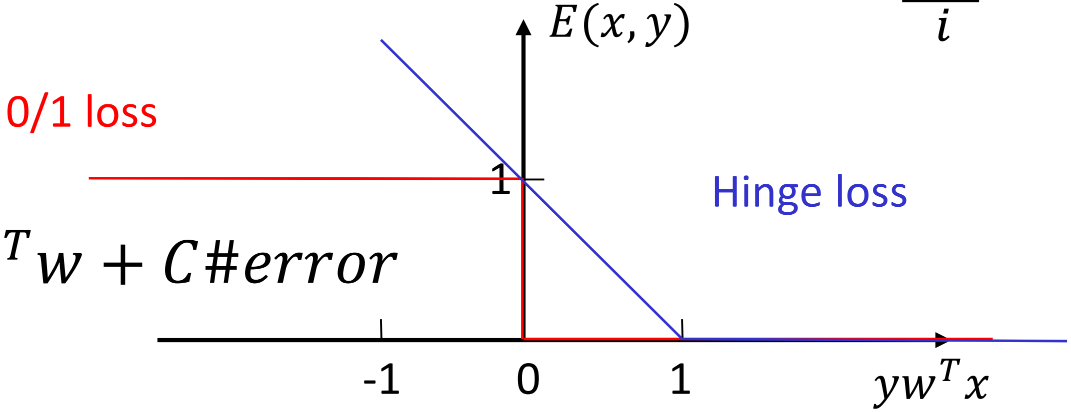
$$argmin_w w^T w + C \sum_i \underline{\max(0, 1 - y_i w^T x_i)}$$

What kind of error is SVM optimizing?

- Hinge loss

$$\underbrace{}_{}\ \text{regularization} \qquad \text{classification loss}$$

$$argmin_w \underbrace{w^T w}_{\text{regularization}} + C \sum_i \underbrace{\max(0, 1 - y_i w^T x_i)}_{\text{classification loss}}$$



$$argmin_w w^T w + C\#error$$

Think about logistic regression

- Optimized by maximum a posterior estimator
  - $argmax_w \sum_x \log p_w(y|x) - \frac{w^T w}{2\sigma^2}$ Note: $y = \{-1, +1\}$

$$\Rightarrow argmin_w w^T w - C \sum_x \log p_w(y|x)$$ Note: $C = 2\sigma^2$

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

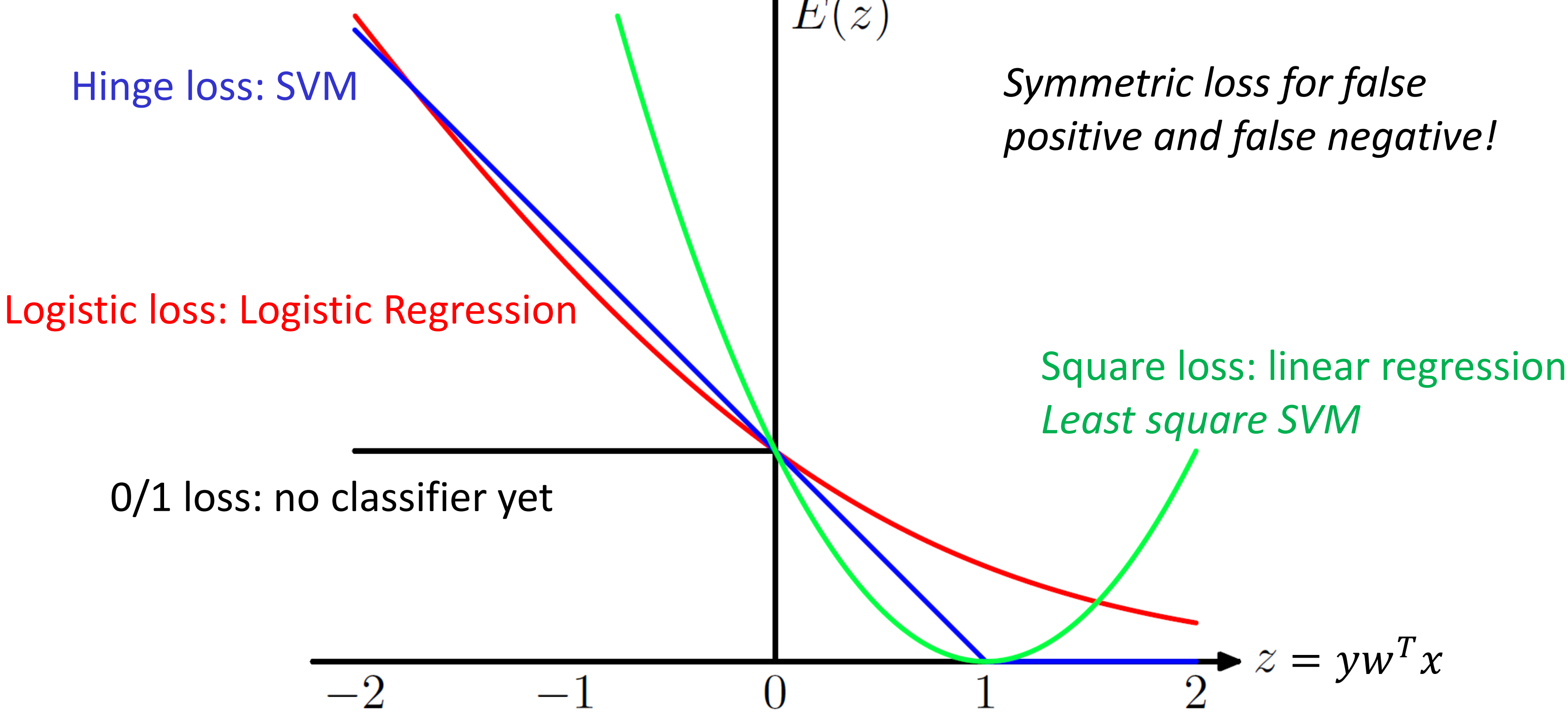
$$\Rightarrow argmin_w \underbrace{w^T w}_{\text{Regularization}} + C \sum_x \underbrace{\log\left(1 + \exp(-y w^T x)\right)}_{\text{Logistic loss}}$$

Regularization

Logistic loss

Different types of classification loss

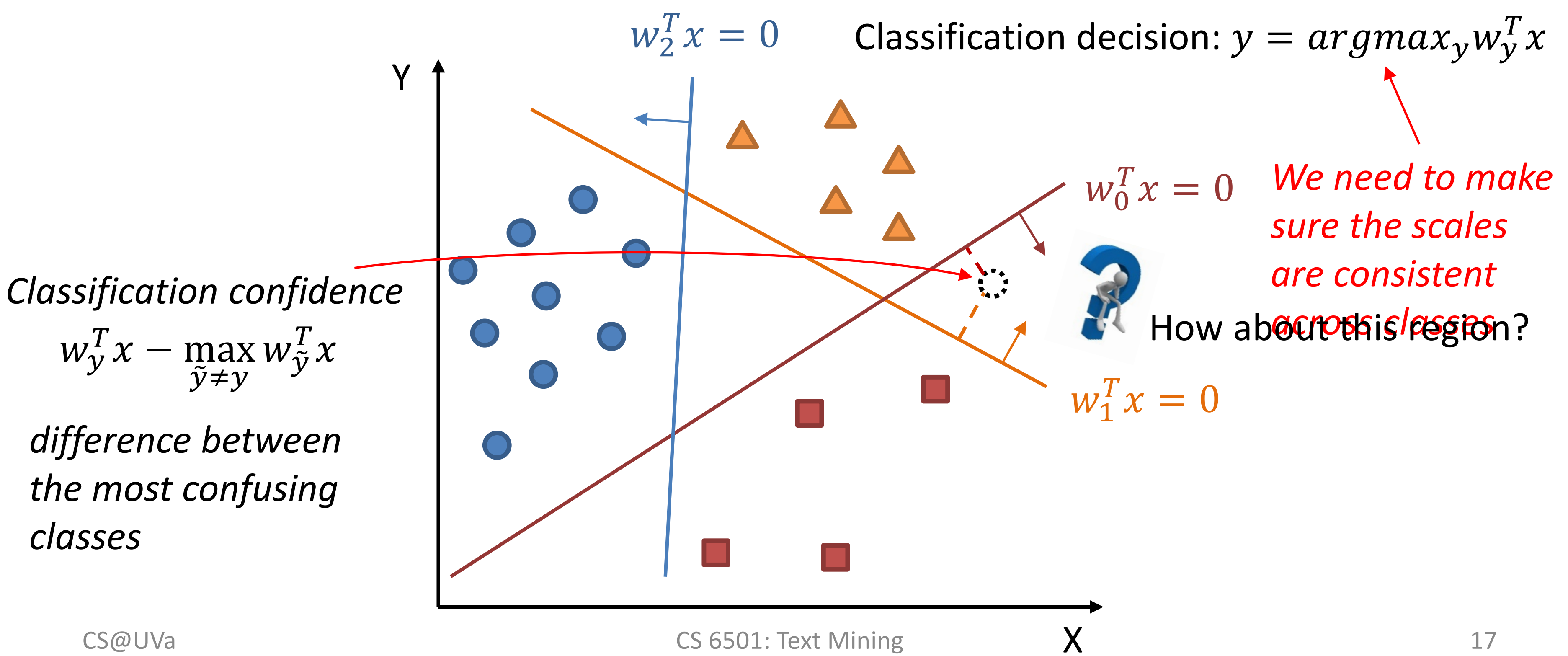
- Discriminative classifiers aim at optimizing those losses



Pattern Recognition and Machine Learning, p337

What about multi-class classification?

- One v.s. All
  - <u>Simultaneously</u> learn a set of classifiers



What about multi-class classification?

- One v.s. All
  - Simultaneously learn a set of classifiers

For binary classification, we have:

$$argmin_w w^T w + C \sum_i \xi_i$$

$$s.t.\ \forall i, y_i w^T x_i \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

Generalize it!

What about multi-class classification?

- One v.s. All
  - Simultaneously learn a set of classifiers

$$argmin_w \sum_y w_y^T w_y + C \sum_i \sum_{y \neq y_i} \xi_i^y$$

$$s.t.\ \forall i, y \neq y_i, w_{y_i}^T x_i \geq w_y^T x_i + 1 - \xi_i^y$$

$$\xi_i^y \geq 0$$

Scale the margin by the rest classes

Parameter estimation

- A constrained optimization problem

$$argmin_w w^T w + C \sum_i \xi_i$$

$$s.t.\ \forall i, y_i w^T x_i \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

  – Can be directly optimized with gradient-based method
    - *Chapelle, Olivier. "Training a support vector machine in the primal." Neural Computation 19.5 (2007): 1155-1178.*

$$argmin_w w^T w + C \sum_i \underline{\max(0,1 - y_i w^T x_i)}$$

piece-wise linear

Dual form of SVM

- A constrained optimization problem

Just to simplify the follow-up derivations

Primal

$$\text{argmin}_w \frac{w^T w}{2} + C \sum_i \xi_i$$

$$s.t.\ \forall i, y_i w^T x_i \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

Lagrangian multipliers: α_i (for $y_i w^T x_i \geq 1 - \xi_i$), β_i (for $\xi_i \geq 0$)

Lagrangian dual

$$L(w, \xi, \alpha, \beta) = \frac{w^T w}{2} + \sum_i \left(C\xi_i - \alpha_i \left(y_i w^T x_i - 1 + \xi_i \right) - \beta_i \xi_i \right)$$

$$s.t.\ \forall i, \alpha_i \geq 0, \beta_i \geq 0$$

Dual form of SVM

- Lagrangian dual

$$L(w,\xi,\alpha,\beta) = \frac{w^T w}{2} + \sum_i \left(C\xi_i - \alpha_i\left(y_i w^T x_i - 1 + \xi_i\right) - \beta_i \xi_i\right)$$

$$s.t.\ \forall i, \alpha_i \geq 0, \beta_i \geq 0$$

Lemma

$$\max_{\alpha\geq 0,\beta\geq 0} L(w,\xi,\alpha,\beta) = \begin{cases} \inf f(w,\xi) & \text{if } (w,\xi) \text{ is feasible} \\ +\infty & \text{otherwise} \end{cases}$$

We need to maximize $L(w,\xi,\alpha,\beta)$ with respect to (α,β) so as to minimize $f(w,\xi)$ with respect to (w,ξ)

Dual form of SVM

- Lagrangian dual

$$L(w,\xi,\alpha,\beta) = \frac{w^T w}{2} + \sum_i \left(C\xi_i - \alpha_i\left(y_i w^T x_i - 1 + \xi_i\right) - \beta_i \xi_i\right)$$

$$s.t.\ \forall i, \alpha_i \geq 0, \beta_i \geq 0$$

Set it to zero

$$\frac{\partial L(w,\xi,\alpha,\beta)}{\partial w} = w - \sum_i \alpha_i y_i x_i \quad \Rightarrow \quad w = \sum_i \alpha_i y_i x_i$$

$$\frac{\partial L(w,\xi,\alpha,\beta)}{\partial \xi_i} = C - \alpha_i - \beta_i \quad \Rightarrow \quad \alpha_i + \beta_i = C$$

take them back to dual form

Dual form of SVM

- Lagrangian dual

$$L(\alpha) = \frac{1}{2}\left(\sum_i \alpha_i y_i x_i\right)^T \left(\sum_i \alpha_i y_i x_i\right) - \sum_i \left(\alpha_i \left(y_i \left(\sum_j \alpha_j y_j x_j\right)^T x_i - 1\right)\right)$$

$$s.t.\ \forall i, 0 \le \alpha_i \le C$$

Dual form of SVM

- Lagrangian dual

In dual form, we need to maximize it!

$$L(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$s.t.\ \forall i, 0 \le \alpha_i \le C$$

QP again!
Easy to optimize!

Complementary slackness

In the optimal solution: $\alpha_i\left(y_i w^T x_i - 1 + \xi_i\right) = 0$

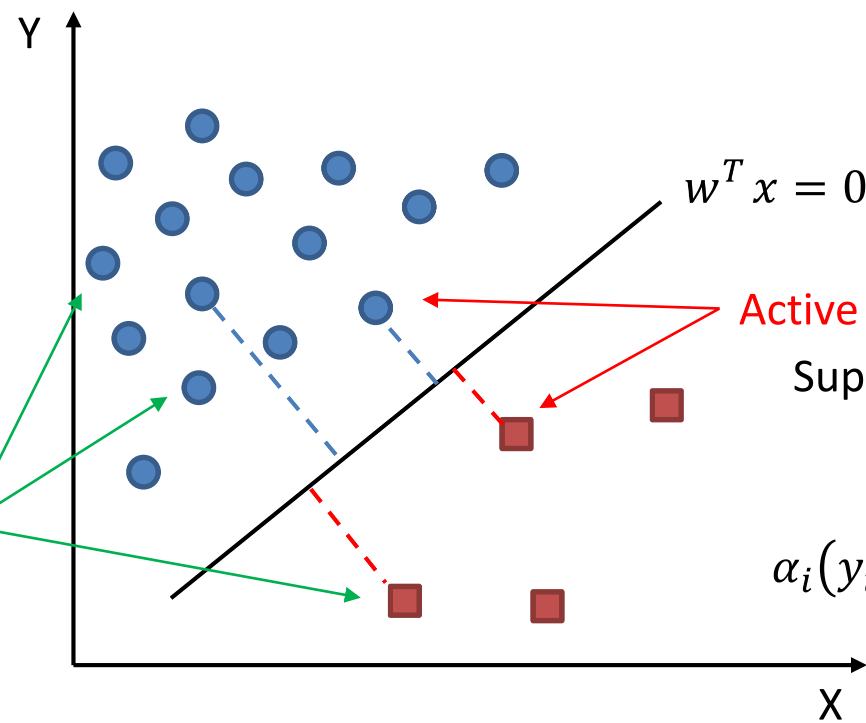
which means $\alpha_i = 0$ if the constraint is satisfied (correct classification)

$\alpha_i > 0$ if the constraint is not satisfied (misclassification)

Sparsity in dual SVM

- Only a few αs can be non-zero

Classification hyperplane $w = \sum_i \alpha_i y_i x_i$



That's why it is called Support Vector Machines

Active constraints $\alpha > 0$

Support vectors

No contribution in deciding the decision boundary

Inactive constraints $\alpha = 0$

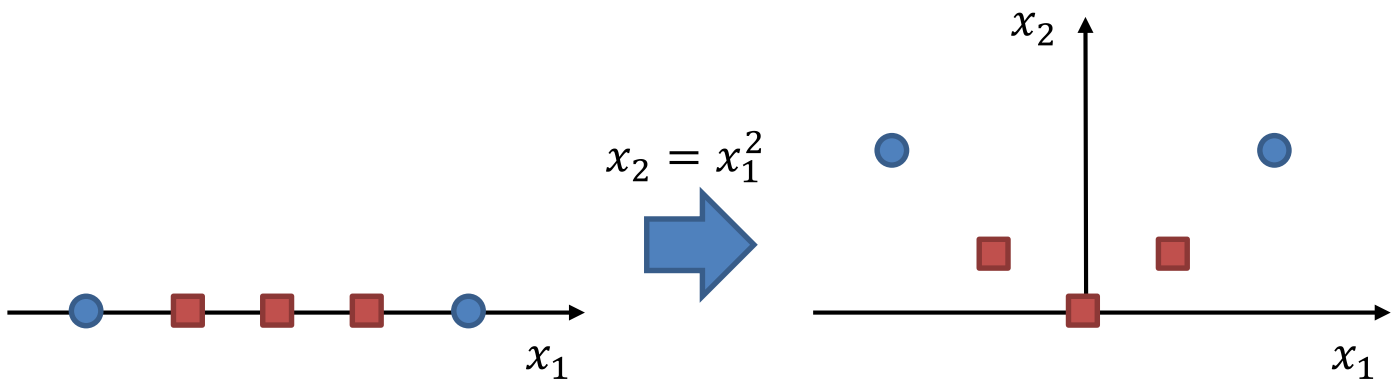
$$\alpha_i(y_i w^T x_i - 1 + \xi_i) = 0$$

Why dual form SVM?

- Primal SVM v.s. dual SVM
  - Primal: QP in feature space
  - Dual: QP in instance space
  - If we have a lot more features than training instances, dual optimization will be more efficient
  - More importantly, the kernel trick!

Non-linearly separable cases

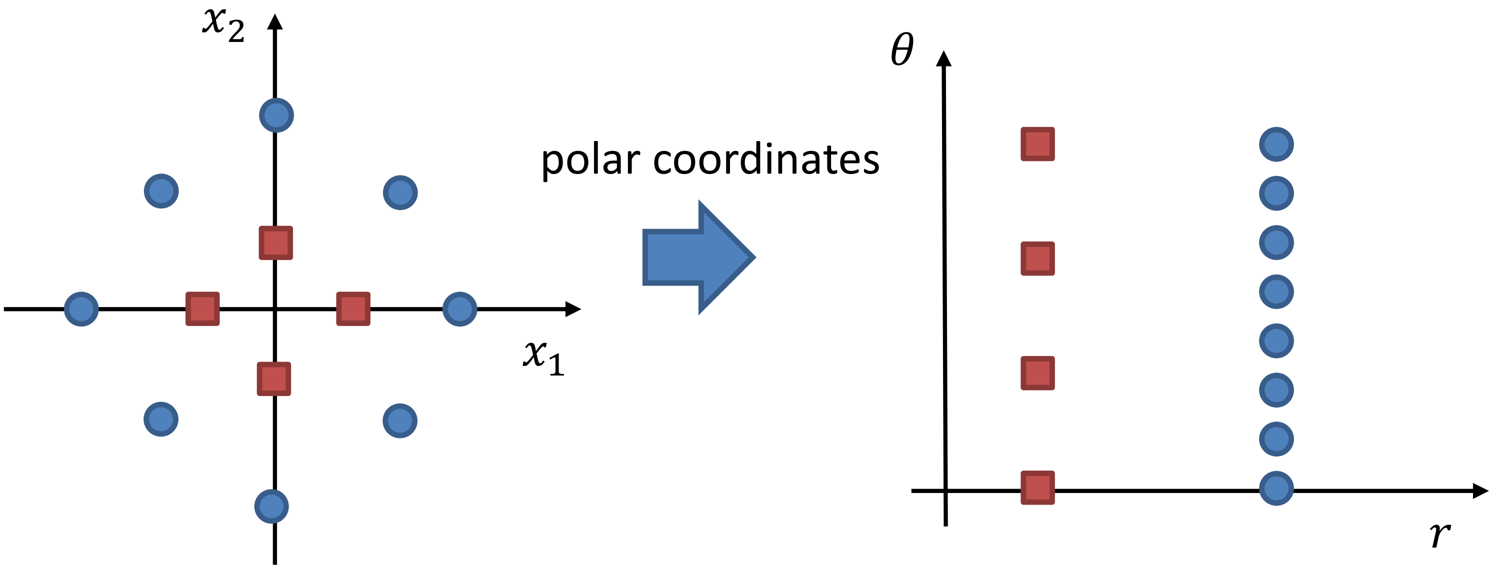
- Non-linear mapping to linearly separable case



Polynomial mapping

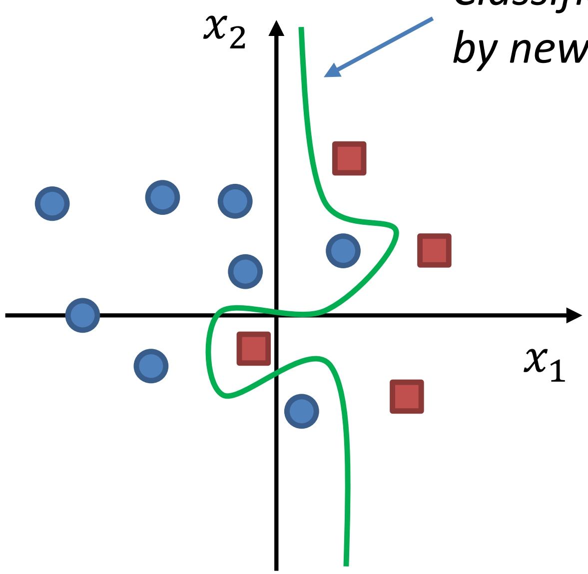
Non-linearly separable cases

- Non-linear mapping to linearly separable case



Non-linearly separable cases

- Explore new features
  - Use features of features of features....



Classification hyperplane determined by new feature space $\phi(x)$

(x_1, x_2)

feature mapping $\phi(x)$

$$\phi(x) = \left(x_1, x_2, x_1^2, x_2^2, x_1 x_2, \dots, \exp(x_1 x_2)\right)$$

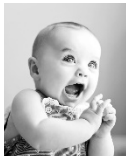
E.g.,. in polynomial mapping: #features~ $O(V^d)$

Feature space explodes very quickly!

Rethink about dual form SVM

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$s.t. \ \forall i, 0 \le \alpha_i \le C$$



What we need is only the inner product between instances!

Take order 2 polynomial as an example:

$$\phi(x, y) = (x^2, y^2, \sqrt{2}xy)$$

If we take the feature mapping first and then compute the inner product:

$$\phi(x_a, y_a)^T \phi(x_b, y_b) = x_a^2 x_b^2 + y_a^2 y_b^2 + 2x_a x_b y_a y_b$$

If we compute the inner product first:

$$[(x_a, y_a)^T (x_b, y_b)]^2 = x_a^2 x_b^2 + y_a^2 y_b^2 + 2x_a x_b y_a y_b$$

No need to take feature mapping at all!

Rethink about dual form SVM

- Kernel SVM

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$s.t.\ \forall i, 0 \le \alpha_i \le C$$

  – Kernel function

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

$\phi(x)$ is some high dimensional feature mapping, but never needed to be explicitly defined

Rethink about dual form SVM

- Kernel SVM

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$s.t.\ \forall i, 0 \le \alpha_i \le C$$

  – Decision boundary
    - $f(x) = w^T \phi(x)$

We still don't need this explicit feature mapping!

Similarity between a testing case and support vectors!

How to construct a kernel

- Sufficient and necessary condition for $K(x, y)$ to be valid kernel
  - Symmetric
  - Semi-positive definite

Just like the requirement for a distance metric

- Operations that preserve kernel properties
  - $K^*(x, y) = cK(x, y)$, where $c > 0$
  - $K^*(x, y) = K_1(x, y) + K_2(x, y)$
  - $K^*(x, y) = \exp\big(K(x, y)\big)$
  - $K^*(x, y) = K_1(x, y)K_2(x, y)$

Common kernels

- Polynomials of degree up to d
  - $K(x, y) = \left(x^T y + 1\right)^d$
- Radial basis function kernel/Gaussian kernel
  - $K(x, y) = \exp\left(-\frac{(x-y)^T(x-y)}{2\sigma^2}\right)$
  - Polynomials of all orders – recall series expansion

Special kernels for text data

- String kernel
  - x and y are two text <u>sequences</u>
  - $K(x, y) = \sum_n \sum_{u \in A^n} \sum_{i:u=x[i]} \sum_{j:u=y[j]} 1$
    - where A is an finite alphabet of symbols

N-gram kernel (length n substrings)

All character sequence of length n

All occurrences of sequence u in x

All occurrences of sequence u in y

Insight of string kernel:

Counting the overlapping of all subsequences with length up to n in x and y

Lodhi, Huma, et al. "Text classification using string kernels." The Journal of Machine Learning Research 2 (2002): 419-444.

Special kernels for text data

- String kernel v.s. Ngram kernel v.s. word kernel

Category	Kernel	Length	F1		Precision		Recall	
			Mean	SD	Mean	SD	Mean	SD
earn	SSK	3	0.925	0.036	0.981	0.030	0.878	0.057
		4	0.932	0.029	0.992	0.013	0.888	0.052
		5	0.936	0.036	0.992	0.013	0.888	0.067
		6	0.936	0.033	0.992	0.013	0.888	0.060
		7	0.940	0.035	0.992	0.013	0.900	0.064
		8	0.934	0.033	0.992	0.010	0.885	0.058
		10	0.927	0.032	0.997	0.009	0.868	0.054
		12	0.931	0.036	0.981	0.025	0.888	0.058
		14	0.936	0.027	0.959	0.033	0.915	0.041
	NGK	3	0.919	0.035	0.974	0.036	0.873	0.062
		4	0.943	0.030	0.992	0.013	0.900	0.055
		5	**0.944**	0.026	0.992	0.013	0.903	0.051
		6	0.943	0.030	0.992	0.013	0.900	0.055
		7	0.940	0.035	0.992	0.013	0.895	0.064
		8	0.940	0.045	0.992	0.013	0.895	0.063
		10	0.932	0.032	0.990	0.015	0.885	0.053
		12	0.917	0.033	0.975	0.024	0.868	0.053
		14	0.923	0.034	0.973	0.033	0.880	0.055
	WK		0.925	0.033	0.989	0.014	0.867	0.057

SVM classification performance on Reuters categories

Special kernels for text data

- Tree kernel

Similar?

Barack Obama is the president of the United States.

Elon Musk is the CEO of Tesla Motors.

```
(ROOT
  (S
    (NP (NNP        ) (NNP      ))
    (VP (VBZ   )
      (NP
        (NP (DT    ) (NN           ))
        (PP (IN   )
          (NP (DT    ) (NNP        ) (NNPS        )))))
    (. .)))
```

```
(ROOT
  (S
    (NP (NNP      ) (NNP      ))
    (VP (VBZ   )
      (NP
        (NP (DT    ) (NN     ))
        (PP (IN   )
          (NP (NNP      ) (NNPS       )))))
    (. .)))
```

Almost identical in their dependency parsing tree!

Special kernels for text data

- Tree kernel

Can be relaxed to allow subsequent computation under unlatching nodes

  - $K(x,y) = \begin{cases} 0 & \text{If } r_1 = r_2 \\ 1 + K(x[r_1], y[r_2]) & \text{otherwise} \end{cases}$

Search through all the sub-trees starting from root node r

Culotta, Aron, and Jeffrey Sorensen. "Dependency tree kernels for relation extraction." Proceedings of the ACL. P423-429, 2004.

Special kernels for text data

- Tree kernel

Can be relaxed to allow subsequent computation under unlatching nodes

$$- K(x,y) = \begin{cases} 0 & \text{If } r_1 = r_2 \\ 1 + K(x[r_1], y[r_2]) & \text{otherwise} \end{cases}$$

Search through all the sub-trees starting from root node r

K_0 = sparse kernel
K_1 = contiguous kernel
K_2 = bag-of-words kernel
K_3 = $K_0 + K_2$
K_4 = $K_1 + K_2$

	Avg. Prec.	Avg. Rec.	Avg. F1
K_1	69.6	25.3	36.8
K_2	47.0	10.0	14.2
K_3	68.9	24.3	35.5
K_4	**70.3**	**26.3**	**38.0**

Relation classification performance

Popular implementations

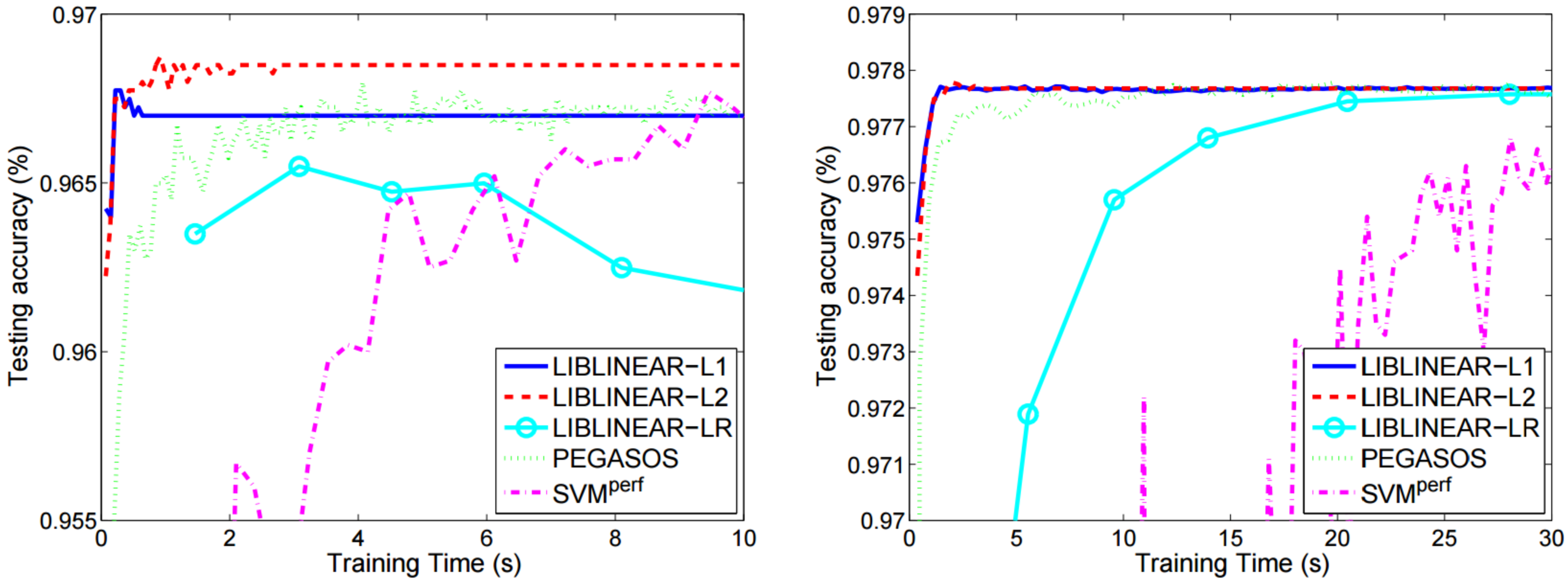
- General SVM
  - **SVM***light* (http://svmlight.joachims.org)
  - libSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm)
  - SVM classification and regression
  - Various types of kernels

Popular implementations

- Linear SVM
  - LIBLINEAR (http://www.csie.ntu.edu.tw/~cjlin/liblinear)
  - Just for linear kernel SVM (also logistic regression)
  - Efficient optimization by dual coordinate descent

Popular implementations

- LIBLINEAR v.s. general SVM



(a) news20, l: 19,996, n: 1,355,191, #nz: 9,097,916 (b) rcv1, l: 677,399, n: 47,236, #nz: 156,436,656

Fan, Rong-En, et al. "LIBLINEAR: A library for large linear classification." The Journal of Machine Learning Research 9 (2008): 1871-1874.

What you should know

- The idea of max margin
- Support vector machines
  - Linearly separable v.s. non-separable cases
  - Slack variable and dual form
  - Kernel method
    - Different types of kernels
  - Popular implementations of SVM

Today’s reading

- Introduction to Information Retrieval
  - Chapter 15: Support vector machines and machine learning on documents
  - Chapter 14: Vector space classification
    - 14.4 Linear versus nonlinear classifiers
    - 14.5 Classification with more than two classes