ESS2222

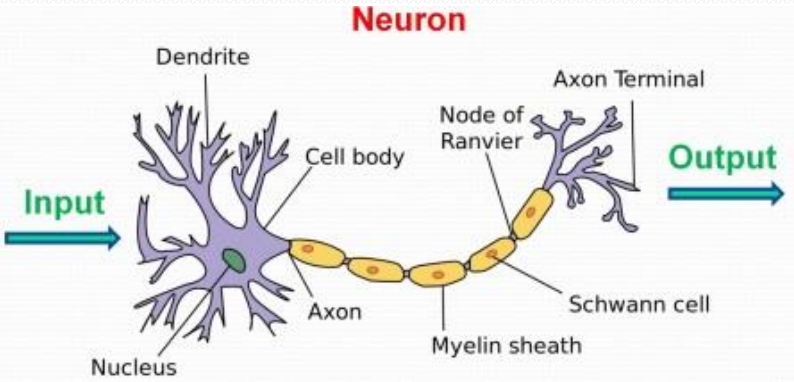
Lecture 5 – Support Vector Machine

Hosein Shahnas

University of Toronto, Department of Earth Sciences,

Outline

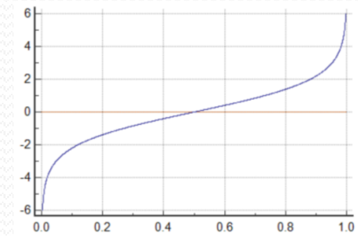
- ❑ **Support Vector Machine (SVM)**
- ❑ **Soft Margin SVM**
- ❑ **Multiclass Problems**
- ❑ **Image Recognition**



Review of Lecture 4

Logistic Regression

$\text{logit}(p) = log\left[\frac{p}{(1-p)}\right]$,   p: (0 – 1) → logit(p): (-∞ – ∞)

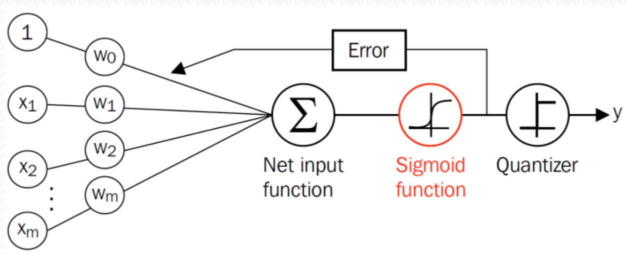


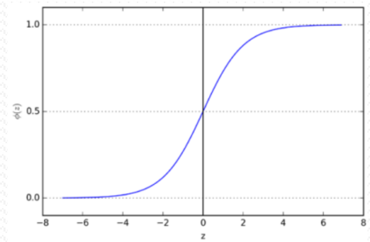
logit(p)

The inverse function

Logistic function: $\phi(z) = \frac{1}{1+e^{-z}}$,   z: (-∞ – ∞) → $\phi(z)$: (0 – 1)

$$\boldsymbol{\Delta wj = -\eta \frac{\partial J}{\partial w_j} = \eta \sum_i \left(y^i - \phi(z^i)\right) x^i{}_j}$$





$\boldsymbol{\phi(z)}$

Predicting Continuous Target Variables

Classification: Credit approval (good/bad)

(x_i, y_i)   **$y_i \in$ [0, 1]**

Regression: Credit line (dollar amount)

(x_i, y_i)   **$y_i \in \mathbb{R}$**

Review of Lecture 4

$E_{in}(w) = \equiv \frac{1}{N} \sum_{n=1}^{N}(w^T x_n - y_n)^2$

$\nabla_w E_{in}(w) = 0 \rightarrow X^T X w = X^T y$

$w = X^{\dagger} y$ where $X^{\dagger} = (X^T X)^{-1} X^T$ pseudo-inverse

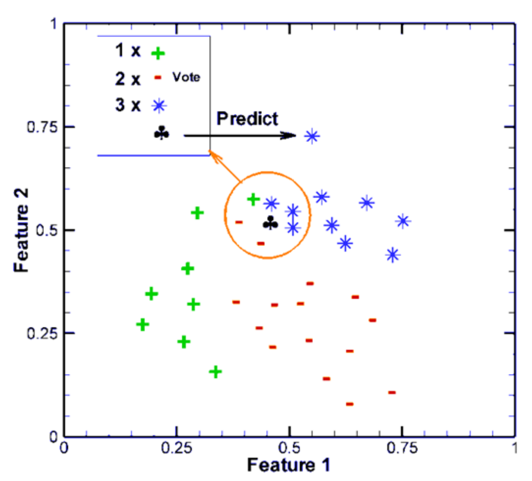
Linear Regression For Classification

1- Solve $w = X^{\dagger} y$ $(y \in R)$

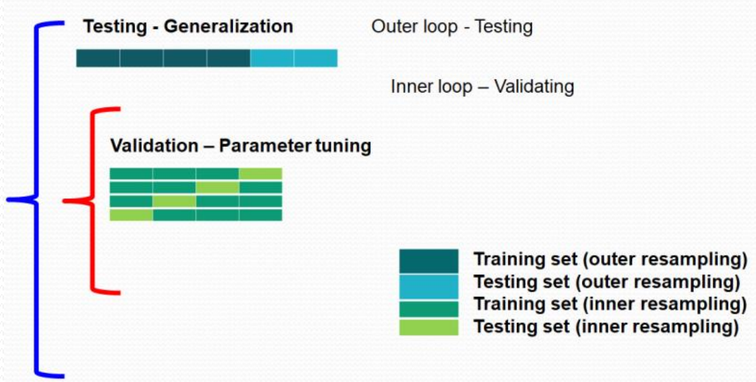
2 - Use the initial values for w obtained by the linear regression method cas good starting point for classification

K-Nearest Neighbours (KNN) Algorithm

$d(X, Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$ n features



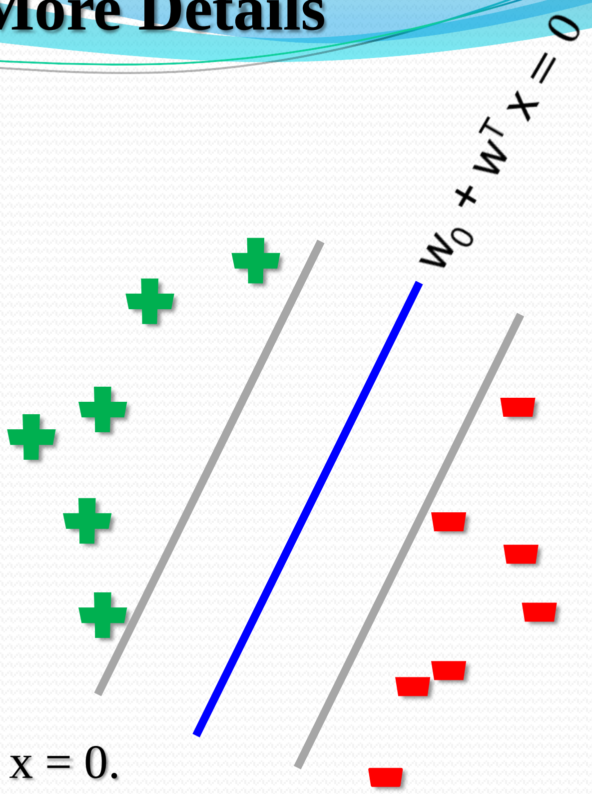
Cross-Validation



Support Vector Machine - More Details

In SVM the objective is to maximize the margin between different classes.
It can be shown that a model with a large margin Can show better performance on out-of-sample.

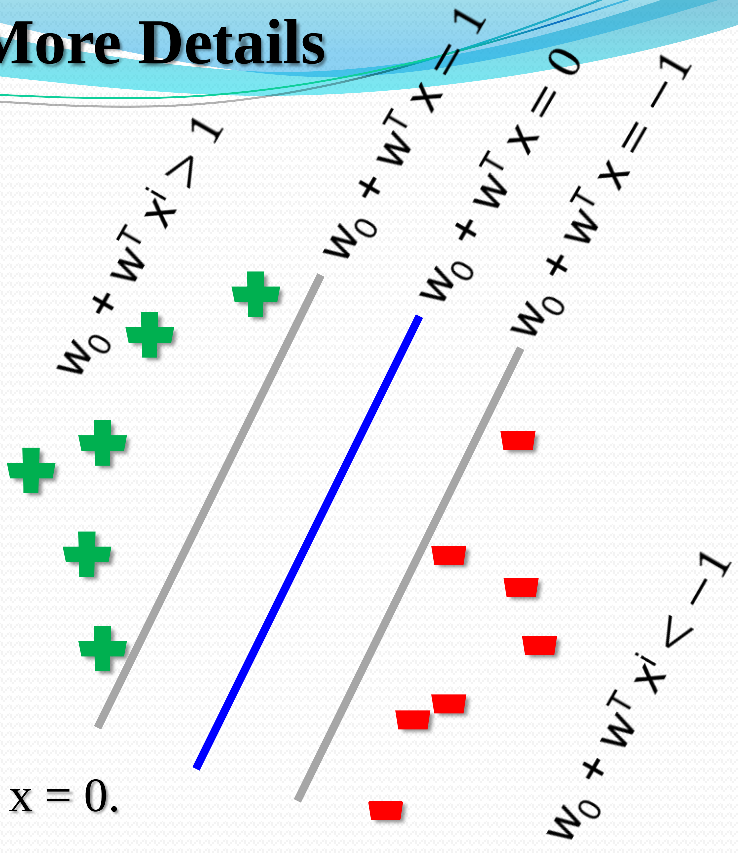
In other words in SVM we want to find the weight vector w such that not only classifies the samples correctly, but also maximizes the margin.



For any point on the class boundary we have $w_0 + w^T x = 0$.

Support Vector Machine - More Details

In SVM the objective is to maximize the margin between different classes.
It can be shown that a model with a large margin Can show better performance on out-of-sample.



In other words in SVM we want to find the weight vector w such that not only classifies the samples correctly, but also maximizes the margin.

For any point on the class boundary we have $w_0 + w^T x = 0$.

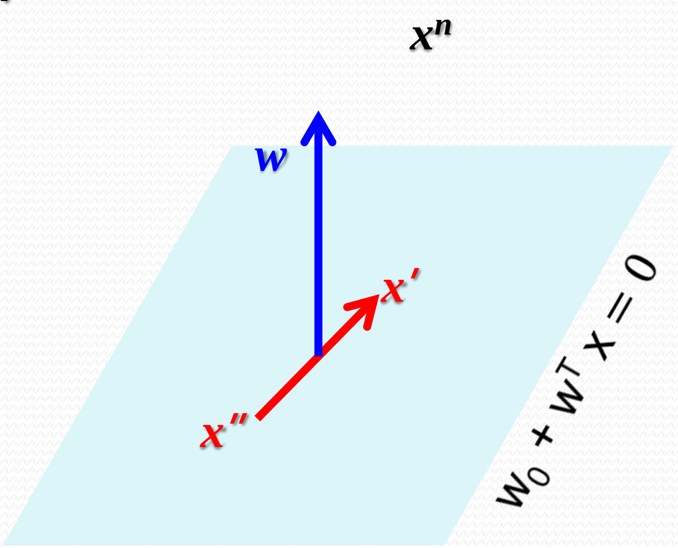
Support Vector Machine - More Details

Suppose x^n is a data point at the margin: $|w_0 + w^T x_n| = 1$

What is the distance between x^n and the plane $w_0 + w^T x = 0$?

The vector w is perpendicular to the plane in the X-space.

$w_0 + wT\, x' = 0$
$w_0 + wT\, x'' = 0$
$w^T (x' - x'') = 0 \quad \rightarrow \quad w^T \perp (x' - x'')$



Support Vector Machine - More Details

Projection of (x^n - x) on w

$$\hat{w} = \frac{w}{\|w\|}$$

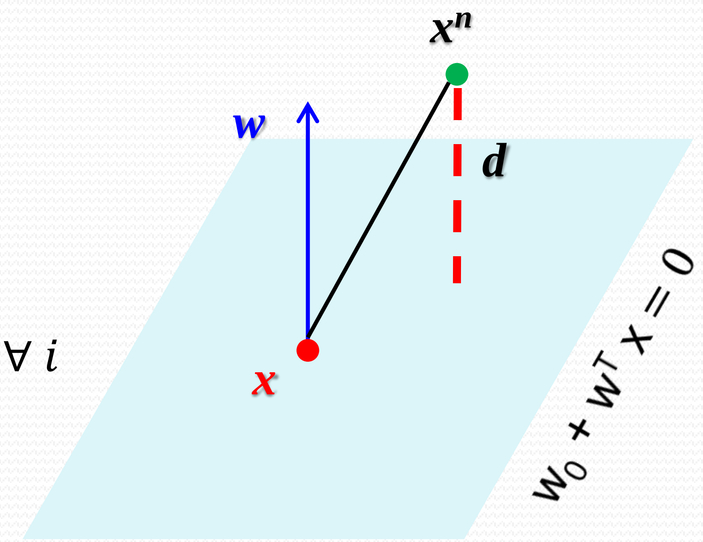
Distance: $d = |\hat{w}^T(x_n - x)| = \frac{1}{\|w\|} |w^T x^n - w^T x|$

$$d = \frac{1}{\|w\|} |w^T x^n + w_0 - w^T x - w_0| = \frac{1}{\|w\|} |w^T x^n + w_0|$$

0

For x^n at the margin:

$$d = = \frac{1}{\|w\|} |w^T x^n + w_0| = \frac{1}{\|w\|}$$



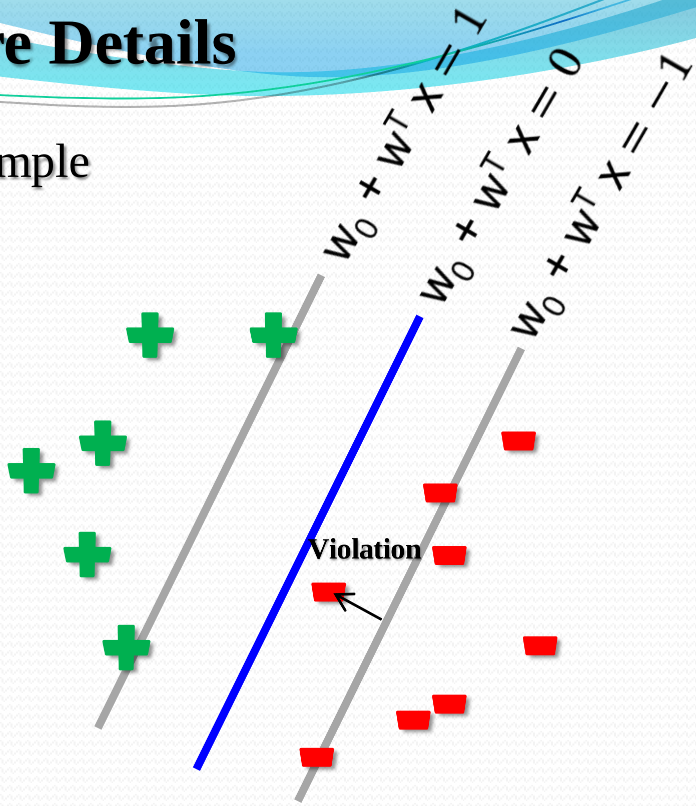
This can be achieved by minimizing $\frac{1}{2} \|w\|^2$

Subject to the condition: $y^i (w_0 + w^T x^i) \geq 1 \quad \forall i$

This is the confidence condition.

Soft Margin SVM - More Details

Suppose that there is a margin violation. Note that the sample may still be correctly classified with zero error.



With this violation the condition

$y^i (w_0 + w^T x^i) \geq 1 \quad \forall\, i$

will change to:

$y^i (w_0 + w^T x^i) \geq 1 - \xi^i \quad \forall\, i$

where $\xi^i \geq 0$ slack vaiable

Total violation = $\sum_i \xi^i$

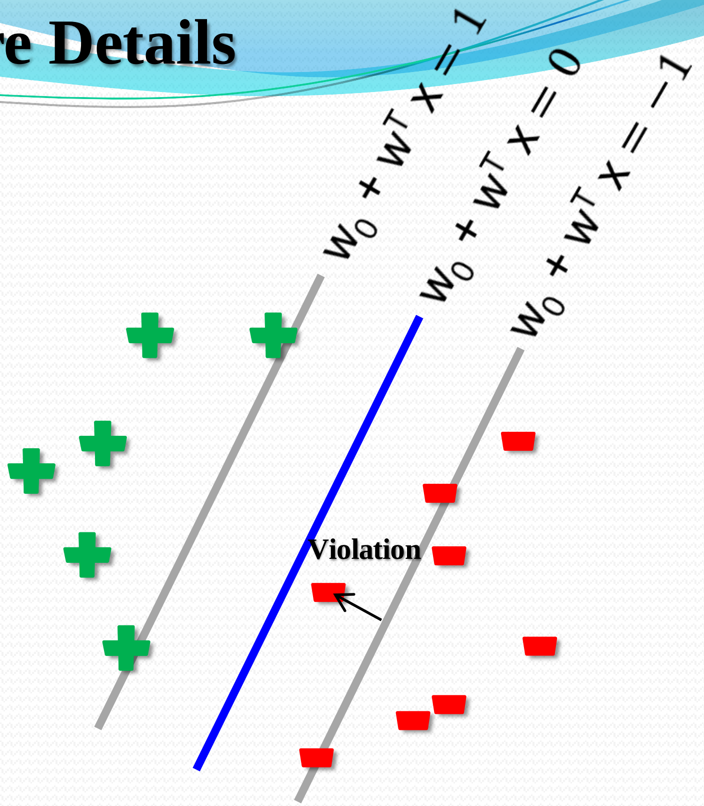
New optimization:

$$\frac{1}{2} \|w\|^2 \rightarrow \frac{1}{2} \|w\|^2 + C \sum_i \xi^i \ , \ \xi^i \geq 0$$

C (the slack coefficient) determines the relative importance of the first term wrt the second term.

C is obtained by cross-validation.

Soft Margin SVM - More Details



C → ∞ : Not violate the margins (hard margin)
C → 0 : Margin violations are allowed

Minimize $\frac{1}{2}\|w\|^2 + C\sum_i \xi^i$, $\xi^i \geq 0$

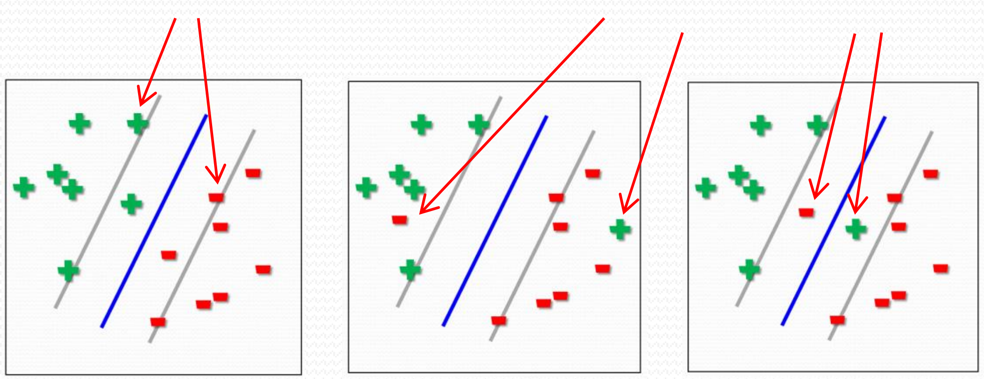
Subject to:

$y^i (w_0 + w^T x^i) \geq 1 - \xi^i \quad \forall i$ where $\xi^i \geq 0$

$\sum_i \xi^i = \sum_i \max\{0, (1-yi(w.xi))\}$

Types of violations:

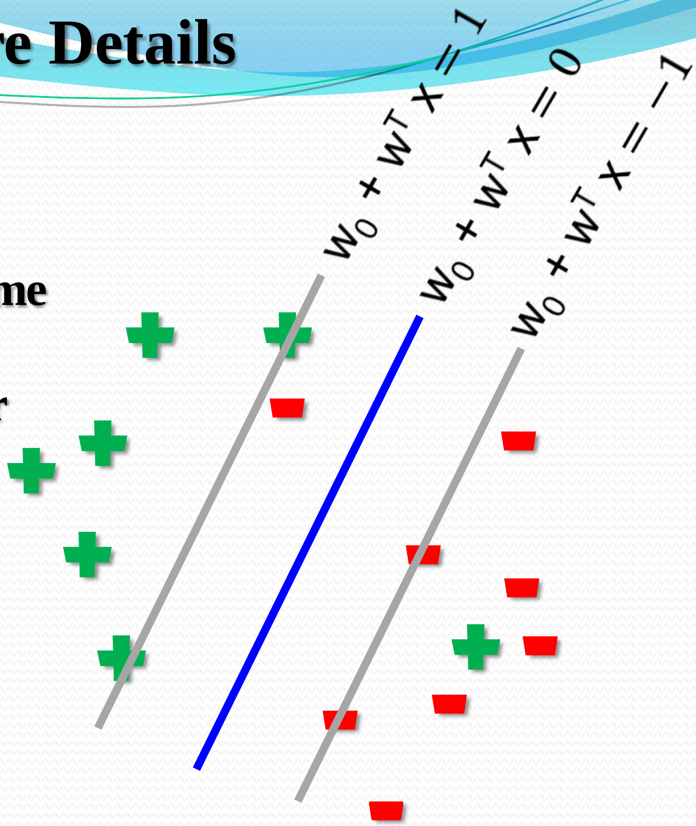
Margin support vectors **Non-margin support vectors**



Soft Margin SVM - More Details

Remarks

The method is called soft margin because it allows some misclassifications.
Suppose the data is slightly nonlinear. This may occur when there is noise in data.

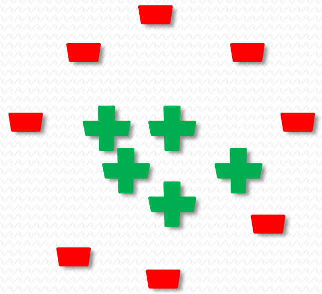


Slightly nonlinear

Soft margin SVM deals with slightly nonlinear problems.

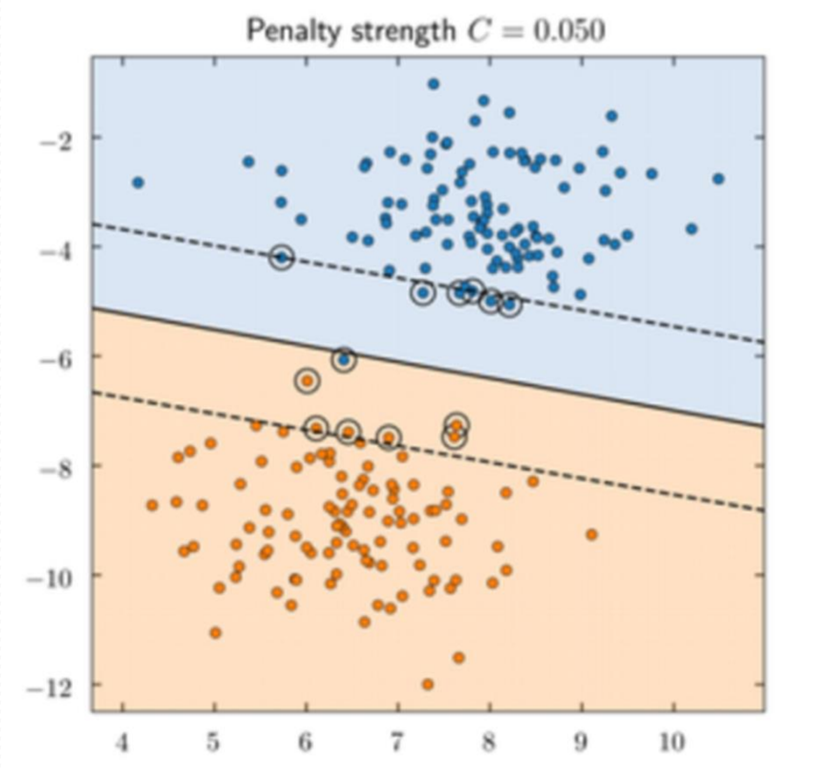
Kernel method deals with seriously nonlinear problems.

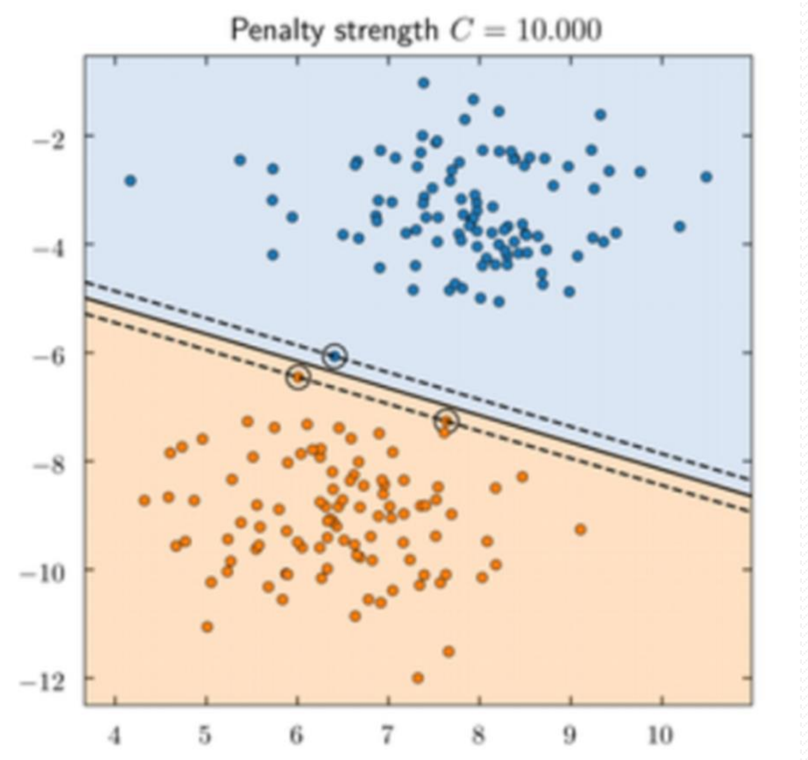
But in reality we deal with practical problems where most datasets have the aspects of both, so we usually combine Kernel and soft margin SVM in almost all problems.



Seriously nonlinear

Soft Margin SVM - More Details

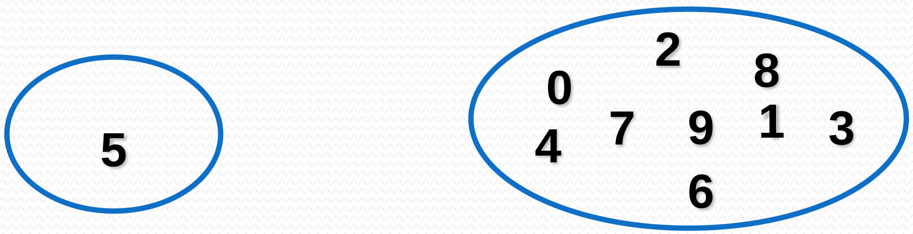




Multiclass Problems

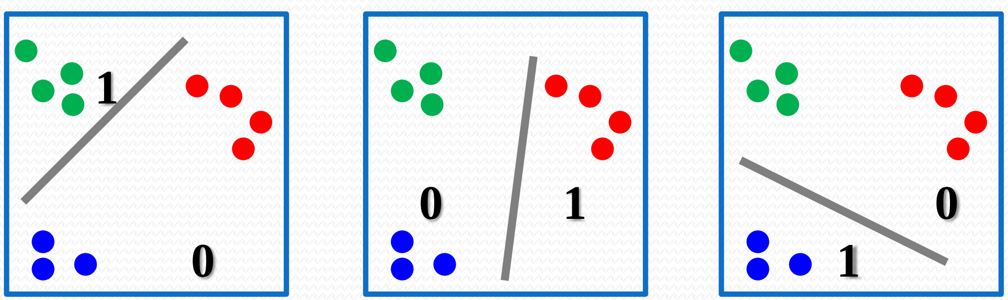
One-vs.-All (OvA)

The perceptron algorithm can be extended to multi-class classification-for example, through the One-vs.-All (OvA) technique.



Class 1 **Class 0**

For 3-class problem



Multiclass Problems

Softmax

This is a generalization of the logistic function to compute meaningful class-probabilities in multi-class settings (multinomial logistic regression).

$$\Phi(z_j) = \frac{1}{1+e^{-z_j}} \quad \Longrightarrow \quad \text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} = \text{softmax}\begin{pmatrix} W_{1,1}x_1 + W_{1,2}x_1 + W_{1,3}x_1 + b_1 \\ W_{2,1}x_2 + W_{2,2}x_2 + W_{2,3}x_2 + b_2 \\ W_{3,1}x_3 + W_{3,2}x_3 + W_{3,3}x_3 + b_3 \end{pmatrix}$$

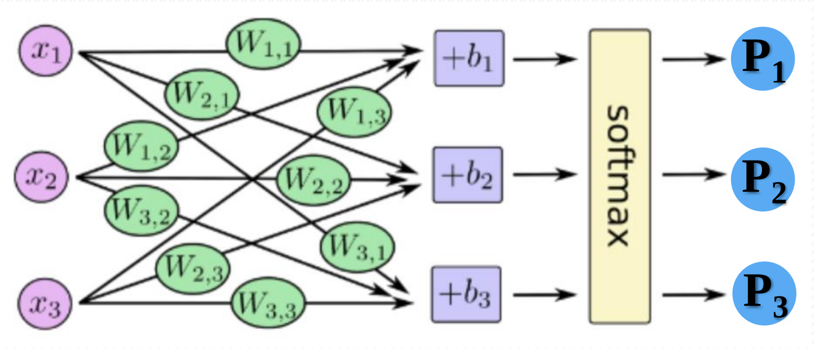
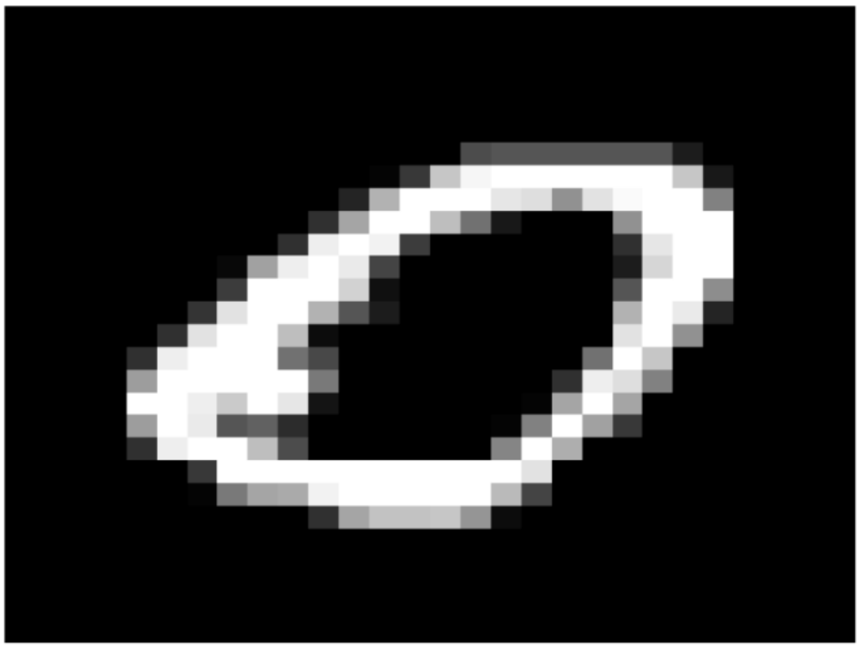


Image Recognition

Each pixel of image has an intensity in the range (0-255)



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	51	159	253	159	50	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	48	238	252	252	252	237	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	54	227	253	252	239	233	252	57	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	10	60	224	252	253	252	202	84	252	253	122	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	163	252	252	252	253	252	252	96	189	253	167	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	51	238	253	253	190	114	253	228	47	79	255	168	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	48	238	252	252	179	12	75	121	21	0	0	253	243	50	0	0	0	0	0
0	0	0	0	0	0	0	0	38	165	253	233	208	84	0	0	0	0	0	0	253	252	165	0	0	0	0	0
0	0	0	0	0	0	0	7	178	252	240	71	19	28	0	0	0	0	0	0	253	252	195	0	0	0	0	0
0	0	0	0	0	0	0	57	252	252	63	0	0	0	0	0	0	0	0	0	253	252	195	0	0	0	0	0
0	0	0	0	0	0	0	198	253	190	0	0	0	0	0	0	0	0	0	0	255	253	196	0	0	0	0	0
0	0	0	0	0	0	76	246	252	112	0	0	0	0	0	0	0	0	0	0	253	252	148	0	0	0	0	0
0	0	0	0	0	0	85	252	230	25	0	0	0	0	0	0	0	0	7	135	253	186	12	0	0	0	0	0
0	0	0	0	0	0	85	252	223	0	0	0	0	0	0	0	0	7	131	252	225	71	0	0	0	0	0	0
0	0	0	0	0	0	85	252	145	0	0	0	0	0	0	0	48	165	252	173	0	0	0	0	0	0	0	0
0	0	0	0	0	0	86	253	225	0	0	0	0	0	0	114	238	253	162	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	85	252	249	146	48	29	85	178	225	253	223	167	56	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	85	252	252	252	229	215	252	252	252	196	130	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	28	199	252	252	253	252	252	233	145	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	25	128	252	253	252	141	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

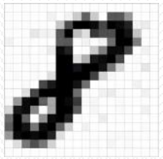
28x28 pixels

Image Recognition

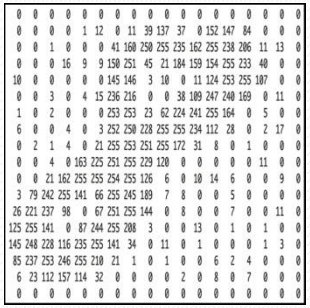
Handwritten Digits Classification
Learning numbers 0-9

n samples (images)

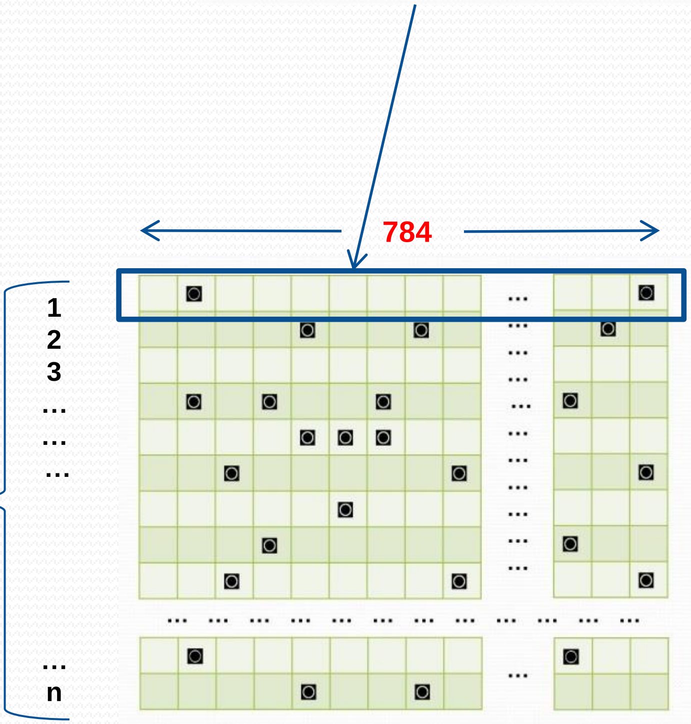
2D: 28x28 pixels



2D to 1D array



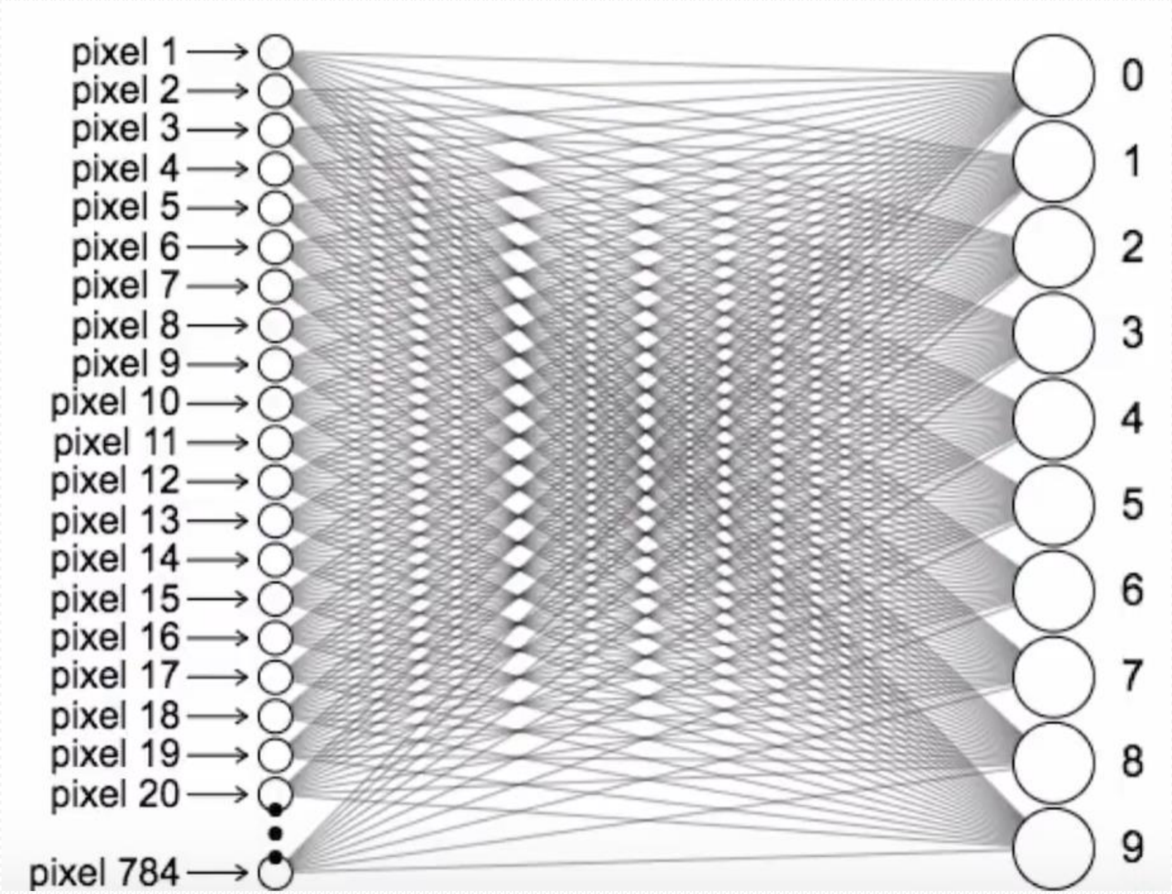
1D: 784



n images, one per line

Learning

Image Recognition



$$Q\begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \\ P_8 \\ P_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

=

$$\sum_i P_i = 1$$

SVM-Nonlinear Problems

sklearn.svm.SVC

class sklearn.svm.SVC(C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None)

C: Penalty parameter of the error term **(default=1.0).**
Kernel: 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable **(default='rbf').**
degree : Degree of the polynomial kernel function ('poly'). Ignored by all other kernels **(default=3).**
gamma: Kernel coefficient **(default ='auto' which uses 1 / n_features).**
Coef0: Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.
shrinking : To save the training time, the shrinking technique tries to identify and remove some bounded elements **(default=True).**
Probability: Whether to enable probability estimates **(default=False).**
tol : Tolerance for stopping criterion **(default=1e-3).**
cache_size: The size of the kernel cache (in MB).
Verbose: Enable verbose output **(default: False)**
max_iter: Hard limit on iterations within solver, or -1 for no limit **(default=-1).**
decision_function_shape: 'ovo', 'ovr', **(default='ovr').**
random_state: The seed of the pseudo random number generator used when shuffling **(default=None; random generator is np.random).**