TUAN TRAN
A20357888
CS584 - Summer 2020

<!!!!!> SEE NEXT PAGE

Tuan Tran
A20357888
CS 584 - Summer

1,

We can :

- Zero padding → create new boundaries with all 0s
- Mirror / duplicate padding → ~~rep~~ Create ~~a~~ new boundaries filled with values of old boundaries.
- Ignore → do nothing and let the image size shrink after ~~convol~~ convolution.

2,

- Using zero padding, Size of resulting image is 1000 × 1000
- ~~Ig~~ when ignoring, size of resulting image is 996 × 996

3,

Assuming image is 3 × 3 × 2 also :

→ Chanel 1 :

10	10	10
10	10	10
10	10	10

Chanel 2 :

20	20	20
20	20	20
20	20	20

→ Chanel 1 ~~convolve~~ after convolution :

10 × 9 = 90

Chanel 2 after convolution : 20 × 9 = 180

→ Image after convolution : 

4,

We store result in a new array instead of reusing input array because after many convolutions, ~~a~~ the image size maybe altered drastically,
For example, convolution without padding with change img width and height while ~~pot~~ pooling ~~to~~ will change number of channels.

5,

- Template matching interpretation:

Convolution is a dot product operation -> essentially a similarity ~~pe~~ operation

-> We are measuring similarity between the filter and each ~~img~~ image region where the weights are templates.

6,

Pooling is needed to perform multi-scale ~~sta~~ analysis in order to analyze the ~~r~~ image at different ~~Sc~~ scales

~~F~~ For ex: in an ~~img~~ image, ~~the~~ a car may be small (~~to~~ calling for small filters) or ~~a~~ very big (calling for bigger filters)

7,

To compensate for ~~sa~~ size shrinkage and avoid losing ~~i~~ image information, we can increase the number of filters which increases the number of channels in resulting image.

8,

- Max-pooling:

$$\begin{bmatrix} 1 & 0 & 2 & \cancel{8}3 \\ 4 & 6 & 5 & 8 \\ 3 & 1 & 1 & 0 \\ 1 & 2 & 2 & 4 \end{bmatrix} \xrightarrow[\text{Stride 2}]{\text{2x2 filter}} \begin{bmatrix} 6 & 8 \\ 3 & 4 \end{bmatrix}$$

- ~~Average p~~ Avg pooling:

$$\begin{bmatrix} 1 & 0 & 2 & 3 \\ 4 & 6 & 5 & 8 \\ 3 & 1 & 1 & 0 \\ 1 & 2 & 2 & 4 \end{bmatrix} \longrightarrow \begin{bmatrix} 11/4 & \cancel{18/4}\ 9/2 \\ 7/4 & 7/4 \end{bmatrix}$$

9. Say image is 100 x 100

- Flattening an image leads to a 10000 ~~(....)~~ dimensional vector, and has the following disadvantages:
  - +, 10000 dimensional input vector → 10000 parameters per unit
    - → too many parameters for each unit.
  - +, loss of spatial context since we collapsed matrix into vector.
  - +, Is not shift invariant, that is, it will be harder to learn same object but at ~~multiple~~ different locations
- By using convolution with, say, a 3x3 filter:
  - +, We have a total of 9 parameters → much fewer
  - +, We preserve spatial context as we can feed the image in its matrix form into ~~....~~ model
  - +, ~~B we~~ the model is shift invariant as the weights of the filter are shared between locations.

SUPPORT VECTOR MACHINES

1. Q1:

- The similarity between SVM and other discriminative models is that we aim to find a discriminative plane that separates the data. However, there may be a lot of such planes => Unlike other approaches, SVM aims to find **the best** plane in which the plane must be as far away from each class as possible => the plane must have a "margin"

- Functional margin is an equation that enforces 2 conditions: examples need to be on the right side **AND** outside the margin
  Geometric margin is essentially functional margin normalized with the magnitude of parameter vector, since functional margin has one problem that it's always possible to maximize it by multiplying the coefficients w by a big constant

- Support vectors are instances that lie **on the margin**. These are points that affect the margin when we want to maximize the margin

2. Q2:

- Functional margin:
  If $y^{(i)} = 1$, $w^T x^{(i)} + w_0 > 1$
  If $y^{(i)} = -1$, $w^T x^{(i)} + w_0 <- 1$
  => Combining into one:
  $y^{(i)}(w^T x^{(i)} + w_0) > 1$

- Geometric margin is functional margin normalized with the magnitude of parameter vector:
  $\frac{y^{(i)}(w^T x^{(i)}+w_0)}{||w||} > \frac{1}{||w||}$

3. Q3:

- We want to maximize the geometric margin => maximize 1/||w|| => minimize ||w||. Thus we can set up the following primal optimization problem:
  Minimize $L_P = (1/2) * w^T w$ **such that** $y^{(i)}(w^T x^{(i)} + w_0) > 1$ **for all examples**

  => We want to minimize L_p subject to the margin constraint. Thus, we can add to L_P a penalty term to penalize when $y(w^T x - w_0) \leq 1$ => We come to an updated objective that takes into account both the optimization problem and the constraint:
  $L_P = 1/2 * w^T w - \sum_{i=1}^{m} \alpha_i [y^{(i)}(w^T x^{(i)} + w_0) - 1]$

- Taking the partial derivative of L_P w.r.t w and equate it to 0, we will come to the solution:
  $w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$
  Similarly, taking the partial derivative w.r.t w_0 and equate to 0 leads to $\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$

- We then plug the solution for w back into L_P and arrive at the dual objective:
  $L_D = 1/2 * (\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)T})(\sum_{j=1}^{m} \alpha_j y^{(j)} x^{(j)}) - \sum_{i=1}^{m} \alpha_i y^{(i)} (\sum_{j=1}^{m} \alpha_j y^{(j)} x^{(j)T}) x^{(i)} - \sum_{i=1}^{m} \alpha_i y^{(i)} w_0 + \sum_{i=1}^{m} \alpha_i$
  We know $\sum_{i=1}^{m} \alpha_i y^{(i)} w_0 = 0$ because $\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$ from previous results
  => $L_D = 1/2 * (\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)T})(\sum_{j=1}^{m} \alpha_j y^{(j)} x^{(j)}) - (\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)T})(\sum_{j=1}^{m} \alpha_j y^{(j)} x^{(j)}) + \sum_{i=1}^{m} \alpha_i$
  => $L_D =- 1/2 * \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} x^{(i)T} x^{(j)} + \sum_{i=1}^{m} \alpha_i$
  => We have the following dual optimization problem:
  Maximize L_D **subject to** $\alpha_i \geq 0,\ \sum_{i=1}^{m} \alpha_i y^{(i)} = 0$

4. Q4:

- In the primal problem, the unknown are the coefficients w and w_0, as well as m lagrange multipliers (one for each of m instances)
  In dual problem, the unknown are the m lagrange multipliers

- The lagrange multipliers determine the importance of the constraint in our constrained optimization problem, the higher the multiplier for an instance, the more we want to enforce the constraint for such instances. For example, in the case of hard margin, we would have the following constraint:
  $y(wx + w_0) > 1$ (meaning the point has to be on correct side **AND** strictly outside the margin)

- The lagrange multipliers can be used to identify support vectors since all support vectors will have their respective lagrange multipliers > 0 while all other points will have lagrange multipliers = 0

5. Q5:

- After finding the lagrange multipliers, we can find the main parameter vector:
  $w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$
  => obtained from solving the primal problem when equating the gradient with respect to w = 0
  As well as the bias:
  $w_0 = (1/\#sv) * \sum_{x^{(i)} \in sv} (y^{(i)} - w^T x^{(i)})$
  Where $sv = \{x^{(i)} | \alpha^i > 0\}$, we obtain w_0 by selecting all examples on the margin (support vectors) which makes $y^{(i)}(w^T x^{(i)} + w_0) - 1 = 0$

6. Q6:

- The problem with hard margins is that it is unrealistic since perfect separation is not always possible
  => We aim to relax constraint for certain instances via slack variables

- The slack variable controls how much "slack" we allow the system to be, and we will have a total of m slack variables, one for each example. We have the new constraint with slack variables added:
  $y^{(i)}(w^T x^{(i)} + w_0) > 1 - \delta_i$
  Where δ_i is the slack variable for instance i
  For example, if slack variable = 0.5
  => $y^{(i)}(w^T x^{(i)} + w_0) > 0.5$ => we allow the instance to be at most half-way inside the margin
- Range of values for slack variables:
  + $\delta_i = 0$ => $y^{(i)}(w^T x^{(i)} + w_0) > 1$ => satisfy the constraint (outside the margin)
  + $0 \le \delta_i \le 1$: allow example to be inside the margin
  + $\delta_i > 1$: allow example to be on the the wrong side

7. Q7:

- We have almost the same primal problem as hard margin but with terms added to account for the new slack variables as well as new constraints:

  $minimize\ L_p = \frac{1}{2} w^T w + C \sum_{i=1}^{m} \delta_i$, such that $\delta_i > 0$ and $y^{(i)}(w^T x^{(i)} + w_0) > 1 - \delta_i$

  => C is a hyperparameters to control the importance we want to give to minimizing the slack variables (if C is large, a lot of emphasis is put on the slack to be minimized)
  => Our new objective which takes into account both constraints:

  $$L_p = \frac{1}{2} w^T w + C \sum_{i=1}^{m} \delta_i - \sum_{i=1}^{m} \alpha_i [y^{(i)}(w^T x^{(i)} + w_0) - 1 + \delta_i] - \sum_{i=1}^{m} \beta_i \delta_i$$

  => We introduce a separate lagrange multipliers term β_i for the slack variable

- After solving for w by equating gradient w.r.t w to 0, we get:
  $$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$$
  + We plug w in the primal objective and arrive at the objective for dual problem:

$$L_D = \frac{1}{2}(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)T})(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}) + C \sum_{i=1}^{m} \delta_i - \sum_{i=1}^{m} \alpha_i y^{(i)} [(\sum_{j=1}^{m} \alpha_j y^{(j)} x^{(j)T}) x^{(i)}] - \sum_{i=1}^{m} \alpha_i y^{(i)} w_0 \sum_{i=1}^{m} \alpha_i - \sum_{i=1}^{m} \alpha_i \delta_i - \sum_{i=1}^{m} \beta_i \delta_i$$

+ We have: $C\sum_{i=1}^{m}\delta_i - \sum_{i=1}^{m}\alpha_i\delta_i - \sum_{i=1}^{m}\beta_i\delta_i = \sum_{i=1}^{m}\delta_i(C-\alpha_i-\beta_i) = 0$ since $C-\alpha_i-\beta_i = 0$ as we equate partial derivative of the primal objective w.r.t δ_i to 0

=> We arrive at the simplified objective:

$$L_D = -\tfrac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}x^{(i)T}x^{(j)} + \sum_{i=1}^{m}\alpha_i$$

Such that:

$\alpha_i \geq 0,\ \sum_{i=1}^{m}\alpha_i y^{(i)} = 0,\ C-\alpha_i-\beta_i = 0,\ \beta_i \geq 0$

+ We can further combine the constraints:

$C-\alpha_i-\beta_i = 0 \Rightarrow \beta_i = C-\alpha_i \geq 0 \Rightarrow C \geq \alpha_i$

=> $0 \leq \alpha_i \leq C$

=> Final optimization problem:

$Max\ L_D$ such that $0 \leq \alpha_i \leq C,\ \sum_{i=1}^{m}\alpha_i y^{(i)} = 0$

8. Q8:

- We have the following standard QP solver problem:
  Minimize (w.r.t X) $1/2\ X^T P X + q^T X$ such that $AX = b$ and $GX \leq h$
  => We need to determine $P, q, A, b, G\ and\ h$
  => We need rearrange the objective L_D into a standard QP problem

- First, we convert the optimization problem from maximizing L_D to **minimizing** $-L_D$:
  $-L_D = 1/2 * \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}x^{(i)T}x^{(j)} - \sum_{i=1}^{m}\alpha_i$
  Rearranging this into matrix form:
  $-L_D = 1/2 * \alpha^T(yy^T * XX^T)\alpha + [-1 ... -1]\alpha$
  Where $\alpha = [\alpha_1 ... \alpha_m].T,\ y = [y^{(1)} ... y^{(m)}].T,\ X = mxm$

- In this form, we can now assign and rearrange the objective into a standard QP problem:
  $P = yy^T * XX^T$ and $q = [-1 ... -1].T$

  For the first constraint $\sum\alpha_i y^{(i)} = 0$, we map:

  $A = y^T,\ b = 0$

  For the second constraint $0 \leq \alpha_i \leq C$:

  G = 2mxm = diagonal matrix of -1s on the diagonal vertically stacked on diagonal matrix of 1s

  $h = [0...0, C...C] = 2mx1$

9. Q9:

- In the matrix form of dual objective, we see the appearance of the gram matrix XX.T
  => We know that the gram matrix measures similarity between each instance, and in the above case is a dot product
  => We can replace this dot product (which is a valid kernel function), with **other kernel** (for ex: gaussian kernel), and thus forming **Kernel SVM**
- The kernel trick is where we map data to higher dimensional space without the need to specify the basis function, allowing us to bypass computational expensiveness, or intractable results
- It is not necessary to define basis function in order to form kernel SVM, we just have to choose the kernel and use the kernel trick to map to higher dimensional space
- Common kernels:
  + Linear: dot product
  + Polynomial: $\kappa(x^{(i)}, x^{(j)}) = (x^{(i)T}x^{(j)} + 1)^q$
  + Radial: $\kappa(x^{(i)}, x^{(j)}) = exp[\frac{1}{\sigma}(x^{(i)} - x^{(j)})^T(x^{(i)} - x^{(j)})$

10. Q10:

- SVR primal:
  We want $-\varepsilon \le (w^Tx^{(i)} - w_0) - y^{(i)} \le \varepsilon$
  => we arrive at $(w^Tx^{(i)} - w_0) - y^{(i)} \le \varepsilon$ and $y^{(i)} - (w^Tx^{(i)} - w_0) \le \varepsilon$

  + Primal formulation:
  Minimize $L_P = \frac{1}{2}||w||^2$, with constraint to $(w^Tx^{(i)} - w_0) - y^{(i)} \le \varepsilon$ and $y^{(i)} - (w^Tx^{(i)} - w_0) \le \varepsilon$

  + With slack variables:
  Minimize $L_P = \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}(\delta_i + \delta_i^{'})$, with constraint to $(w^Tx^{(i)} - w_0) - y^{(i)} \le \varepsilon + \delta_i^{'}$ and $y^{(i)} - (w^Tx^{(i)} - w_0) \le \varepsilon + \delta_i$

- SVM Ranking:
  The output is a rank, and we have the following primal problem with slack variables:
  Minimize $L_P = 1/2 * ||w||^2 + C\sum_{i=1}^{m}\delta_i$ such that $w^Tx^{(i)} + w_0 > w^Tx^{(j)} + w_0 + 1 - \delta_{ij}$ => $w^T(x^{(i)} - x^{(j)}) > 1 - \delta_{ij}$

11. Q11:

- SMO is an iterative algorithm that breaks the optimization problem into a series of smallest possible sub-problems, which are then solved analytically. We have the linear equality constraint with α_i , because of this, the smallest possible sub-problem will involve two such multipliers α_1 and α_2 .

Then, the algorithm finds α_1 that violates KKT conditions. It then pick a second lagrange multiplier α_2 and optimize (α_1, α_2). The algorithm repeats the first step until convergence