Pedestrian Detection

Shen Zhengwei

November 4, 2019

Pedestrian Detection

What is Object Detection

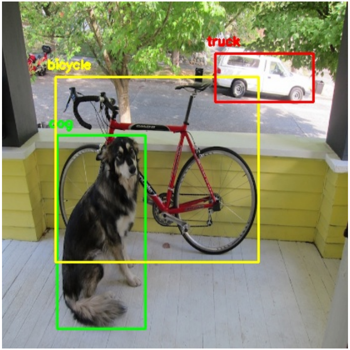
Object detection is a problem that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include **face detection** and **pedestrian detection**.

What is Pedestrian Detection

A kind of object detection, goal is localizing all subjects that are human in an image or video sequence

Usage

Robotics, Motion analysis, Autonomous vehicles, Surveillance camera's early-warning system



Feature Descriptor

- A feature descriptor is a representation of an image or an image patch that simplifies the image by extracting useful information and throwing away extraneous information.
- What is "useful"? , i.e. is very useful for tasks like image recognition and object detection.The feature vector produced by these algorithms when fed into an image classification algorithms like Support Vector Machine (SVM) produce good results.
- Typically, a feature descriptor converts an image of size $width \times height \times 3$ (channels) to a feature vector/array of length n. In the case of the HOG feature descriptor, the input image is of size $64 \times 128 \times 3$ and the output feature vector is of length 3780.

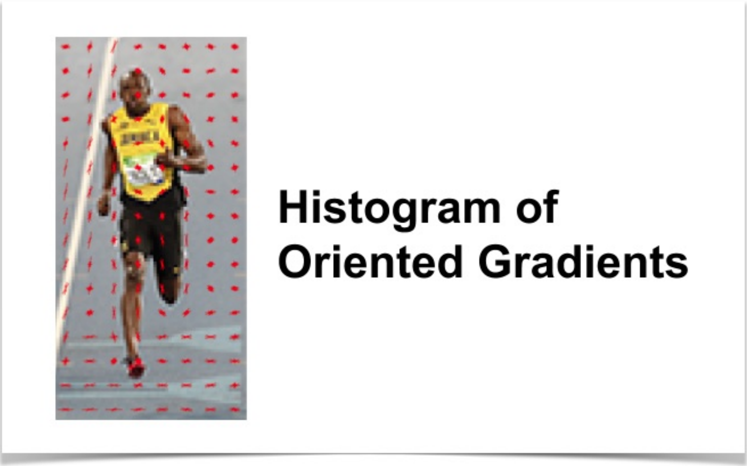


Figure: Histogram of Oriented Gradient(HOG)

Histogram of Oriented Gradient(HOG)

- The widespread use of HOG presented in the paper by N.Dalal and B.Triggs(France) at the CVPR 2005[1].
- The essential thought behind the HOG descriptor is that local object appearance and shape within an image can be described by the distribution of gradients(oriented gradient), i.e. using the statistics of the Histogram of the local oriented gradient as the local feature descriptor.

[1] Dalal N, Triggs B. Histograms of oriented gradients for human detection, CVPR, 1: 886-893, 2005

Histogram of Oriented Gradient(HOG)

- The implementation of HOG is:
  - Divide the image into small connected regions called cells;
  - For each pixels within each cell, a histogram of gradient directions is compiled;
  - The descriptor is the concatenation of these histograms;
- For improved accuracy, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this measure to normalize all cells within the block. This normalization results in better invariance to changes in illumination and shadowing.

Why is HOG?

- Because HOG operates on local cells, it is invariant to geometric and photometric transformations since such changes would only appear in larger spatial regions, except for object orientation.
- Moreover, as Dalal and Triggs discovered, coarse spatial sampling, fine orientation sampling, and strong local photometric normalization permits the individual body movement of pedestrians to be ignored so long as they maintain a roughly upright position. The HOG descriptor is thus particularly suited for human detection in images.

Outline of HOG Algorithms

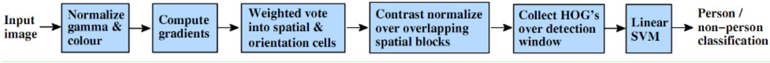
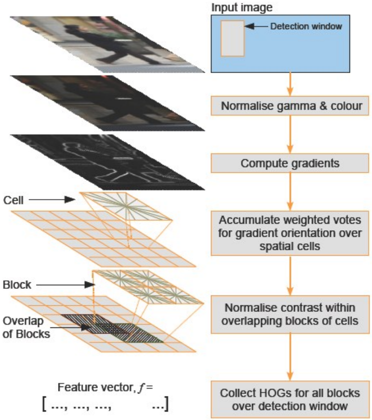
- 

Figure: Overview of the Method

- 

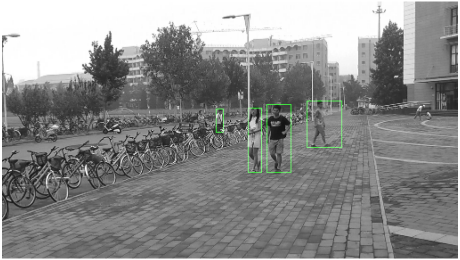
Pedestrian Detection

Pedestrian Detection Processing

1) Get ROI (Region of Interest), or regions may include human
   1.1) Slide Window
   1.2) Cluster

2) Describe every regions with feature
   2.1) Histogram of oriented gradients (HOG)
   2.2) Haar
   2.3) R-CNN

3) Classify features to determine the regions include human or not
   3.1) Support Vector Machine (SVM)
   3.2) Adaboost
   3.3) Dense (Softmax)



HOG Pedestrian Detection

Algorithm 1 HOG Pedestrian Detection

INPUT: $Image[OriHeight][OriWidth]$
Gamma Transformation: $Image[i][j] = GammaTable[Image[i][j]]$
Get the *Gradient* and *Angle* image with Sobel operator
$BlockSet = [Partial\ Image\ with\ BlockX, BlockY, StrideX, StrideY]$
for BlockImg in BlockSet **do**
    $CellSet = [Partial\ BlockImg\ with\ CellX, CellY]$
    for Cell in CellSet **do**
        Statistic Histogram of oriented gradients: $Hisrogram[nbins]$
    end for
    Normalize the Hisrogram
    $EigenVector = Hog\ descriptor\ in\ the\ Block$
    if $LinearSVM(EigenVector) == 0$ **then**
        No human in the *Block*
    else
        $WarningBlockSet.push\_back(Block)$
    end if
end for
OUTPUT: WarningBlockSet

Pretreatment: Gamma Transformation

Gamma Transformation

$$Img_{output} = A \cdot Img_{Input}^{\gamma}$$

Algorithm 2 Gamma Transform (Normal)

```
INPUT: Image[height][width]
for i = 1 : height do
    for j = 1 : width do
        GammaImg[i][j] = A · Image[i][j]^γ
    end for
end for
OUTPUT: GammaImg
```

Pretreatment: Gradient and Angle

Gradient and Angle

$$SobelX = [-1, 0, 1] \qquad SobelY = [-1, 0, 1]^T$$

Convolution

$$GradientX = Image * SobelX$$

$$GradientY = Image * SobelY$$

So we have

$$Gradient = \sqrt{GradientX^2 + GradientY^2}$$

$$Angle = \arctan\left[\frac{GradientY}{GradientX}\right]$$

Get ROI: Slide Window Method

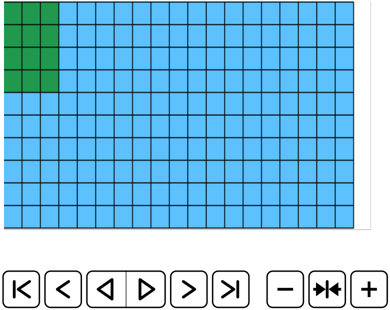
Slide Window in 1-dim vector Just like convolution

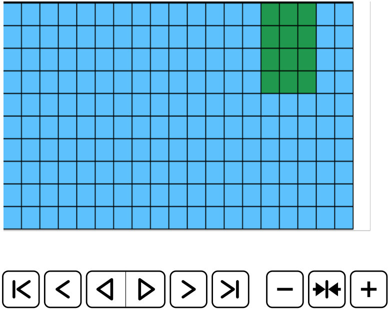
$$a, b, c, d, e, f, g, \ldots$$

Then, the results of slide window are (width $=3$, strideX $=1$)

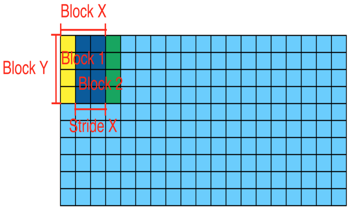
$$(a, b, c), (b, c, d), (c, d, e), \ldots$$

Slide Window in 2-dim image

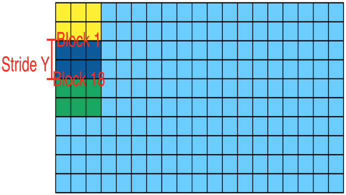




Slide Window in 2-dim image



(a) Slide X



(b) Slide Y

Why we have divided the image into (e.g.8×8) cells?

- One of the important reasons to use a feature descriptor to describe a patch of an image is that it provides a compact representation.
- Also, calculating a histogram over a patch makes this representation more robust to noise.
- HOG was used for pedestrian detection initially. 8×8 cells in a photo of a pedestrian scaled to 64×128(the size of block) are big enough to capture interesting features (e.g. the face, the top of the head etc.)

Why we have divided the image into (e.g.8 $\times$ 8) cells?

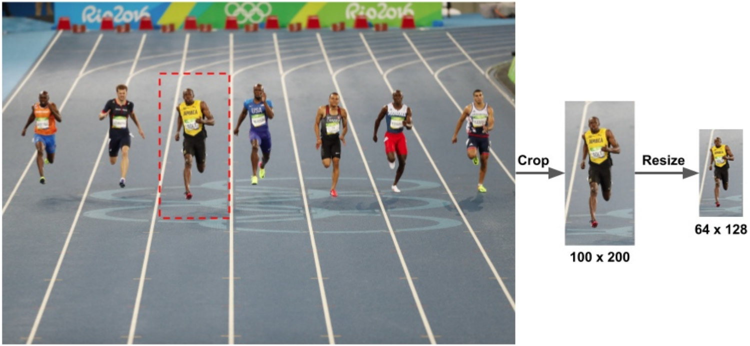
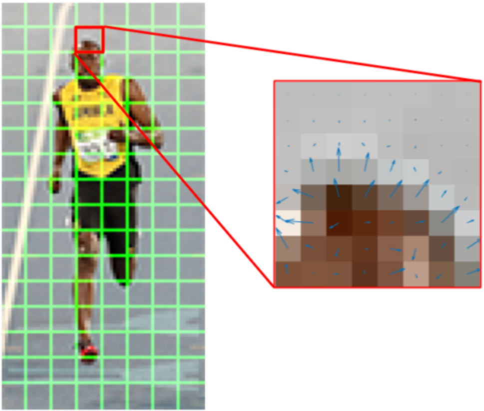


Figure: HOG preprocessing

Why we have divided the image into (e.g.8 $\times$ 8) cells?



2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

Figure: Center: The RGB cell and gradients represented using arrows(*The arrow shows the direction of gradient*($0-180$ *instead of* $0-360$) *and its length shows the magnitude.*). Right: The gradients in the same patch represented as numbers.

Describe ROIs: Histogram of Oriented Gradients(HOG)

Get CellSet

Cell 1	Cell 2	Cell 3
Cell 4	Cell 5	Cell 6
Cell 7	Cell 8	Cell 9
Cell 10	Cell 11	Cell 12

Cell 1	Cell 2	Cell 3
Cell 4	Cell 5	Cell 6
Cell 7	Cell 8	Cell 9
Cell 10	Cell 11	Cell 12

Cell 1	Cell 2	Cell 3
Cell 4	Cell 5	Cell 6
Cell 7	Cell 8	Cell 9
Cell 10	Cell 11	Cell 12

3	5	8	10	10	10	12	14
4	6	10	12	15	20	25	30
20	30	72	59	34	12	25	82
60	75	91	81	76	82	19	54
24	30	77	27	32	87	71	46
80	57	23	83	75	32	58	6
15	49	4	47	2	89	9	75
39	81	64	85	91	35	93	42

53	13	39	96	34	71	71	72
34	33	62	81	14	48	20	2
43	45	65	32	92	83	42	16
57	9	14	89	58	4	19	47
12	86	99	71	7	79	57	68
21	58	26	21	51	7	3	54
40	90	52	91	93	38	59	57
14	96	55	19	89	75	10	64

20	100	63	28	66	20	75	83
42	93	1	96	59	26	55	93
15	30	29	83	73	77	8	17
61	35	16	35	76	54	41	32
63	37	91	72	86	22	28	77
52	95	90	27	86	55	14	94
75	30	62	23	51	73	75	99
39	1	38	88	99	61	10	45

Statistic the HOG

3	5	8	10	10	10	12	14
4	6	10	12	15	20	25	30
20	30	72	59	34	12	25	82
60	75	91	81	76	82	19	54
24	30	77	27	32	87	71	46
80	57	23	83	75	32	58	6
15	49	4	47	2	89	9	75
39	81	64	85	91	35	93	42

(a) Cell1

Angle	0-30	30-60	60-90	90-120	120-150	150-180
Statistic	3					
	5					
		8				
		10				
			10			
			10			
						12
		14				
		4				
		6				
	10					
			12			
		15				
			20			
						25
					30	

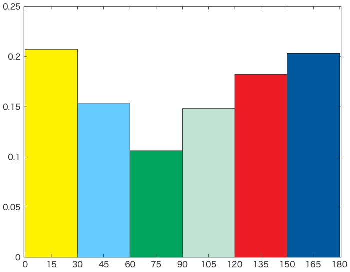
(b) Statistic

Describe ROIs: HOG - Get the histogram

Statistic the HOG

Angle	0-30	30-60	60-90	90-120	120-150	150-180	
Statistic	3	8	10	60	30	12	
	5	10	10	24	30	25	
	10	14	12	80	34	25	
	81	4	20	15	75	19	
	82	6	72	49	76	71	
	54	15	59	4	30	58	
	87	20	12	47	77	9	
	46	82	91	39	27	75	
	83	6		81	32	85	
	75	2			57	91	
	32	89			23	35	
		64				42	
		93					
Gradient	558	413	286	399	491	547	2694
Ave	0.20713	0.1533	0.10616	0.14811	0.18226	0.20304	

(c) After Statistic



(d) Histogram

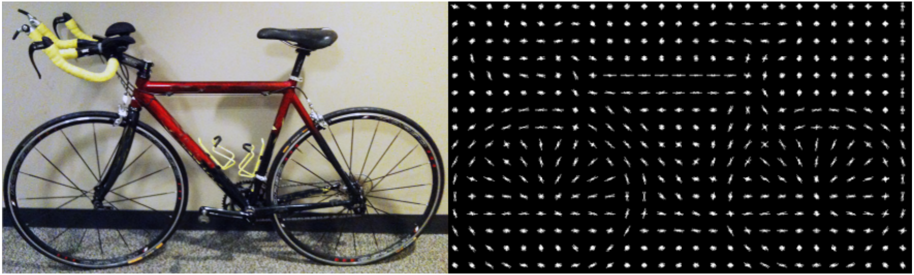
And we can describe this cell as
$[0.20713, 0.15330, 0.10616, 0.14811, 0.18226, 0.20304]$

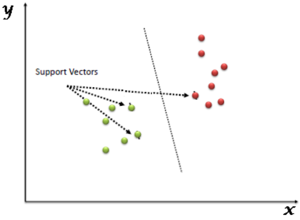
Describe ROIs: HOG - Describe the Block

Cell 1	Cell 2	Cell 3
Cell 4	Cell 5	Cell 6
Cell 7	Cell 8	Cell 9
Cell 10	Cell 11	Cell 12

After statistic all cells, we will get a $6 \times 12 = 72$ length eigen-vector.
That means, we **described** this $(3 \times 4) \times (8 \times 8) = 768$ pixels 2-dimension image as a 72 length 1-dimension vector.

After this describe processing, we will classify every feature to determine if there is a human or not.





- SVM is a supervised machine learning algorithm which can be used for both classification or regression;
- In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyperplane that seperate the two classes very well.

Support vectors are the data points that lie closest to decision surface(hyperplane). They are the data points most difficult to classify and they have direct bearing on the optimum location of the decision surface. SVM is a frontier which best segregates the two classes (hyperplane/ line).

Classifiser : SVM-Support Vector Machine

Data Set

A data set can be expressed as

$$(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_N}, y_N)$$

where $\mathbf{x_i} = (x_{i1}, x_{i2}, \ldots, x_{im})$ is a m-dimension vector.

For example, in data set **Iris**, $\mathbf{x_i} = (x_{i1}, x_{i2}, x_{i3}, x_{i4})$ is the feature a every iris. That means, we can describe a flower as a feature with four real number.

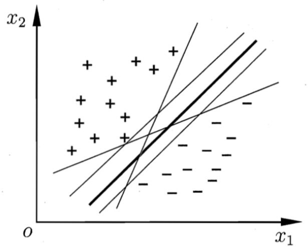
To determine a new flower is Setosa, Versicolor or Virginica, we need deremine the 'line' between different kind of flower.

	Classification	SL	SW	PL	PW
1	Setosa	5.1	3.5	1.4	0.2
2	Setosa	4.9	3	1.4	0.2
...	...	...	...	...	...
51	Versicolor	7	3.2	4.7	1.4
52	Versicolor	6.4	3.2	4.5	1.5
...	...	...	...	...	...
101	Virginica	6.3	3.3	6	2.5
102	Virginica	5.8	2.7	5.1	1.9
...	...	...	...	...	...

Table: Iris

Iris

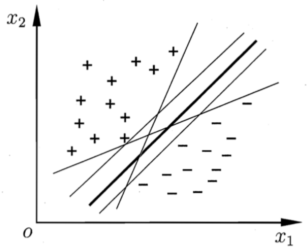
The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: Sepals Length(SL), Sepals Width(SW), Petals Length(PL), Petals Width(PW)



In the pedestrian detection, we just have two classes for all data, one is *with a human* and the other is *Not with a human*. So we can express our data as

$$(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_N}, y_N)$$

where $\mathbf{x_i} = (x_{i1}, x_{i2}, \ldots, x_{im})$ is a m-dimension vector given by HOG descriptor, and $y_i = 1$ if the block include a human and $y_i = -1$ is the block not include human.



For a binary classification, most common discriminate classifier is a *separating line or a hyperplane* in high dimension space. We can express this kind of classifier as

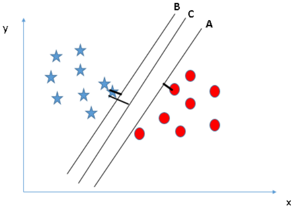
$$\omega^{\mathbf{T}} \cdot \mathbf{x} + b = 0$$

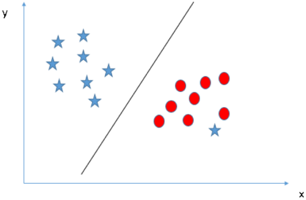
where $\omega^{\mathbf{T}} = (\omega_1, \omega_2, \ldots, \omega_m)^T$ is a normal vector of the hyperplane . And b is the bias to the line.

As the number of this kind of classifiers is infinity, it is necessary to find the **best** one in all of them to describe two class.

How can we identify the right(the best) hyperplane?

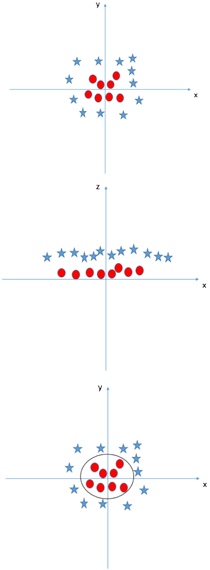
Classifiser : SVM-Support Vector Machine





- Here, maximizing the distances between nearest data point (either class) and hyperplane will help us to decide the right hyperplane. This distance is called as Margin.
- We can see that the margin for hyperplane C is high as compared to both A and B. So, C is the best hyperplane.
- Another reason for selecting the hyperplane with higher margin is robustness. If we select a hyperplane having low margin then there is high chance of miss-classification
- SVM has a feature to ignore outliers and find the hyperplane that has maximum margin. Hence, we can say, SVM is robust to outliers.

Classifiser : SVM-Support Vector Machine



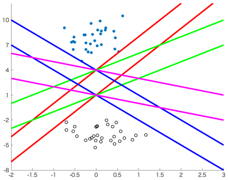
- Till now, we have only looked at the *linear hyperplane.*
- SVM can solve it easily by introducing additional feature(additional dimension). That is to define a new feature $z = x^2 + y^2$.
- To SVM techniques, should we need to add this feature manually to have a hyperplane? Answer is NO!
- SVM has a technique called the *Kernel trick*. These are functions transforms low dimensional space data into a higher dimensional space i.e. it converts not separable problem to separable problem. This actually is the generalization of *similarity* to new kinds of *similarity* measures based on dot products.

Classification: SVM - a strict condition

If $\omega^{\mathbf{T}} \cdot \mathbf{x} + b = 0$ is classifer, then we have

$$\omega^{\mathbf{T}}\mathbf{x_i} + b \geq 0 \quad \textit{while } y_i = +1$$

$$\omega^{\mathbf{T}}\mathbf{x_i} + b \leq 0 \quad \textit{while } y_i = -1$$



we can combine these two formula as

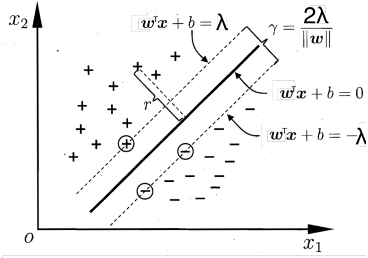
$$y_i(\omega^{\mathbf{T}}\mathbf{x_i} + b) \geq 0 \quad \forall i = 1, 2, \ldots N$$

As we want the **BEST** one for classification, we can impose more stronger constraints on the condition:

$$y_i(\omega^{\mathbf{T}}\mathbf{x_i} + b) \geq \lambda > 0 \quad \forall i = 1, 2, \ldots N$$

However, even we determined the 'intercept' between two 'line', we still have infinity choice classification.

Classification: SVM - Another thought



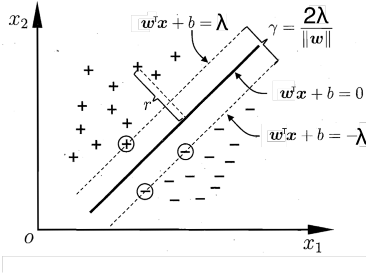
We can express SVM by maximizing the margin around the separating hyperplane

$$\max_{\omega,b} \quad \lambda$$
$$s.t. \quad y_i(\omega^T \mathbf{x_i} + b) \geq \lambda \ \ \forall i = 1, 2, \ldots N$$

Rather than the max of intercept, we can find a group of parallel line which have the largest distance. So we have

$$\max_{\omega,b} \quad \frac{2\lambda}{\|\omega\|}$$
$$s.t. \quad y_i(\omega^T \mathbf{x_i} + b) \geq \lambda \ \ \forall i = 1, 2, \ldots N$$

Classifier: SVM - optimization problem



$$\max_{\omega,b} \quad \frac{2\lambda}{\|\omega\|}$$
$$s.t. \quad y_i(\omega^T \mathbf{x_i} + b) \geq \lambda \ \ \forall i = 1, 2, \ldots N$$

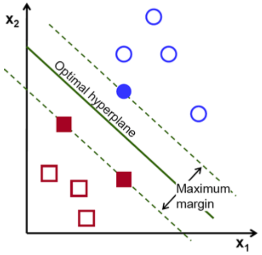
or after transformation

$$\min_{\omega,b} \quad \frac{1}{2}\omega\omega^T$$
$$s.t. \quad y_i(\omega^T \mathbf{x_i} + b) \geq 1 \ \ \forall i = 1, 2, \ldots N$$

This is a 2-dimension programming problem.

Classifier SVM -Support vectors(input data points)

- Which data points should influence optimality?
  - all points?
    - Linear regression;
    - Neural nets
  - Only *difficult points* close to decision boundary.
    - That is the Support vector machines.
- Support vectors are the elements of the training set that would *change the position* of the dividing hyperplane if removed;
- Thus, it is the input data points that just touch the boundary of the margin(street).



Classifier SVM -optimization algorithm

- To maximize the margin, we just to minimize $\|\omega\|$, with the constraints that there are

$$\min_{\omega,b} \quad \frac{1}{2}\omega\omega^T$$
$$s.t. \quad y_i(\omega^T \mathbf{x_i} + b) \geq 1 \quad \forall i = 1, 2, \ldots N$$

- This a convex quadratic optimization problems with only linear constraints.
- We can write the constraint as

$$g_i(w) = 1 - y_i(w^T x_i + b) \leq 0, \tag{1}$$

We have one such constraint for each training example. Note that from the *KKT dual complementarity condition*, we will have dual variable $\alpha_i > 0$ only for the training examples that have functional margin exactly equal to one i.e. the ones corresponding to constraints that hold with equality, $g_i(w) = 0$.

Classifier SVM -optimization algorithm

- The Lagrangian to the optimization problem

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

with constraints $\alpha_i \geq 0, \forall i.$.

- Notice that $\max_\alpha L(w, b; \alpha)$ equals to original optimization problem when obeying inequality constraint; otherwise, it will go to ∞, which indicates (2) equals to the original problem.
- Thus, to solve original constrained optimization problems equals to solve the following min max unconstrained optimization problems

$$\min_{\omega, b} \max_{\alpha \geq 0} L(w, b; \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i [y_i(w^T x_i + b) - 1] \qquad (3)$$

Classifier SVM -optimization algorithm

- By the dual and saddle points theory, we have the primal and the dual problems relationship

$$d^* = \max_{\alpha \geq 0} \underbrace{\min_{\omega, b} L(w, b; \alpha)}_{primal} \leq \min_{\omega, b} \underbrace{\max_{\alpha \geq 0} L(w, b; \alpha)}_{dual} = p^* \tag{4}$$

- And, under certain conditions(for example, objective and constrained are convex), we have

$$d^* = p^*$$

- In this case, we can find the dual formulation of the original problem by fist $\min_{\omega,b} L(w, b; \alpha)$, then plug their solutions into $L(w, b; \alpha)$ to get a function with respect to dual variable α.

Classifier SVM -optimization algorithm

- To find the dual form of the problem, we need to

$$\theta(\alpha) = \min_{w,b} L(w, b; \alpha) \tag{5}$$

- Let

$$\nabla_w L(w, b; \alpha) = w - \sum_i^N \alpha_i y_i x_i = 0 \tag{6}$$

This implies that $w = \sum_i^N \alpha_i y_i x_i$

Classifier SVM -optimization algorithm

- Let
$$\nabla_b L(w, b; \alpha) = \sum_i^N \alpha_i y_i = 0 \tag{7}$$
- If we take the value of w and plug that back into the Lagrangian (Equation (2)), and simplify, we get
$$L_d(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N y_i y_j \alpha_i \alpha_j (x_i)^T x_j - b \sum_i^N \alpha_i y_i \tag{8}$$
- But from equation (7), the last term must be zero, so we have
$$L_d(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N y_i y_j \alpha_i \alpha_j (x_i)^T x_j \tag{9}$$

Recall that we got to the equation above by minimizing L with respect to w and b. Putting this together with the constraints $\alpha_i \geq 0$ and the constraint (7) we obtain the following dual optimization problem:

Classifier SVM -optimization algorithm

- $$\max_{\alpha} L_d(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i y_i \underbrace{(x_i)^T x_j}_{<x_i, x_j>} y_j \alpha_j$$
$$s.t. \alpha_i \geq 0, i = 1, ...N \quad and \quad \sum_{i}^{N} \alpha_i y_i = 0 \tag{10}$$
- Hence, we can solve the dual in lieu of solving the primal problem[2].
- We will solve (10) by differentiating it w.r.t. α, and setting it to zero. Most of the α_i will turn out to be zero. Then non-zero $\alpha_i > 0$ will correspond to the *support vectors*.

[2] Andrew Ng's CS229, Lecture notes for Support Vector Machines

Classifier SVM -optimization algorithm

- This is because, the inequality constraints $y_i(\omega^T \mathbf{x_i} + b) \geq 1$ in original optimization problem adding to the objective function using Lagrangian multiplier α indicates that, we always have $\alpha_i = 0$ or $y_i(\omega^T \mathbf{x_i} + b) = 1$ to be hold. This is the result of KKT condition to inequality constraints:

$$\begin{cases} \alpha_i \geq 0 \\ y_i(\omega^T \mathbf{x_i} + b) \geq 1 \\ \alpha_i(y_i(\omega^T \mathbf{x_i} + b) - 1) = 0 \end{cases} \tag{11}$$

- If $\alpha_i = 0$, this data point (y_i, x_i) will NOT be used to make a predication in (12);
- Otherwise $\alpha_i > 0$ means that $y_i(\omega^T \mathbf{x_i} + b) = 1$, these data points is nothing but the support vectors.

Classifier SVM -optimization algorithm

- By the KKT theorem: the solution to (10) equals to the solution to the original problem.

Classifier SVM -optimization algorithm

- By the KKT theorem: the solution to (10) equals to the solution to the original problem.
- But why are we doing this? Why not just solve the original problems?

Classifier SVM -optimization algorithm

- By the KKT theorem: the solution to (10) equals to the solution to the original problem.
- But why are we doing this? Why not just solve the original problems?
- Because this will let us solve the problem by computing the just the *inner products* of x_i, x_j(which will be very important when we want to solve non-linearly separable classification problems.)

Classifier SVM -optimization algorithm

- By the KKT theorem: the solution to (10) equals to the solution to the original problem.
- But why are we doing this? Why not just solve the original problems?
- Because this will let us solve the problem by computing the just the *inner products* of x_i, x_j(which will be very important when we want to solve non-linearly separable classification problems.)
- Remember that our task is to make a prediction at a new input data x. That is we need to calculate $\omega^T x + b$ and make the prediction. But, using the relationships between primal variables ω and dual variables α, we have

$$\omega^T x + b = \sum_{i}^{N} \alpha_i y_i < x_i, x > +b \tag{12}$$

Classifier SVM -optimization algorithm

- Notice that intercept b is also relevant to dual α trough its relationship to ω.

Classifier SVM -optimization algorithm

- Notice that intercept b is also relevant to dual α trough its relationship to ω.
- Hence, if we've found the α_i by maximize (10), then to make a prediction, we just to calculate a quantity that depends only on the *inner product* between input data x and the x_i(which is known). Moreover, we saw that most of $\alpha_i = 0$ except for the support vector.

Classifier SVM -optimization algorithm

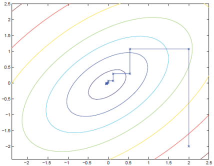
- Notice that intercept b is also relevant to dual α trough its relationship to ω.
- Hence, if we've found the α_i by maximize (10), then to make a prediction, we just to calculate a quantity that depends only on the *inner product* between input data x and the x_i(which is known). Moreover, we saw that most of $\alpha_i = 0$ except for the support vector.
- Thus, many of the terms in the sum in (12) will be zero, and we really need to find only the inner products between x and the support vectors (of which there is often only a small number) in order calculate (12) and make our prediction.

Classifier SVM -optimization algorithm

- Notice that intercept b is also relevant to dual α trough its relationship to ω.
- Hence, if we've found the α_i by maximize (10), then to make a prediction, we just to calculate a quantity that depends only on the *inner product* between input data x and the x_i(which is known). Moreover, we saw that most of $\alpha_i = 0$ except for the support vector.
- Thus, many of the terms in the sum in (12) will be zero, and we really need to find only the inner products between x and the support vectors (of which there is often only a small number) in order calculate (12) and make our prediction.
- Least but not last, *inner product* between input data x and the x_i is used to measure the *similarity* between x and x_i. And, for the non-linear SVM, the Kernel trick can be used to define the similarity in the transformed space(often is a high-dimensional space).

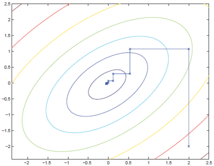
Algorithms to solve the dual problems (10)–Coordinate ascent

- Firstly, the dual problem (10) is a differentiable(but constrained) quadratic problem;



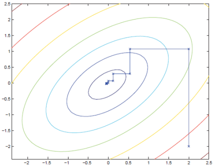
Algorithms to solve the dual problems (10)–Coordinate ascent

- Firstly, the dual problem (10) is a differentiable(but constrained) quadratic problem;
- Coordinate ascent method which will be changing only one α_i variable a time, while keeping others fixed.



Algorithms to solve the dual problems (10)–Coordinate ascent

- Firstly, the dual problem (10) is a differentiable(but constrained) quadratic problem;
- Coordinate ascent method which will be changing only one α_i variable a time, while keeping others fixed.
- If for each α_i, the coordinate ascent can be a fairly efficient algorithm, then solving (10) can be performed efficiently.



Algorithms to solve the dual problems (10)–Coordinate ascent

- Now, the dual optimization problem that we want to solve is

$$
\begin{aligned}
&\max_{\alpha} L_d(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i y_i \underbrace{(x_i)^T x_j}_{<x_i, x_j>} y_j \alpha_j \\
&s.t. \alpha_i \geq 0, i = 1, \ldots N \quad \sum_{i}^{N} \alpha_i y_i = 0
\end{aligned}
\tag{13}
$$

Algorithms to solve the dual problems (10)–Coordinate ascent

- Now, the dual optimization problem that we want to solve is

$$\max_{\alpha} L_d(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i y_i \underbrace{(x_i)^T x_j}_{<x_i, x_j>} y_j \alpha_j$$
$$s.t. \alpha_i \geq 0, i = 1, \ldots N \quad \sum_{i}^{N} \alpha_i y_i = 0 \tag{13}$$

- Can we make any progress just taking only one coordinate α_i and fixed α_j with $j \neq i$? The answer is no! because the constraint $\sum_i^N \alpha_i y_i = 0$ ensures that $\alpha_i y_i = -\sum_{j \neq i} \alpha_j y_j$, that is $\alpha_i = -y_i \sum_{j \neq i} \alpha_j y_j$.

Algorithms to solve the dual problems (10)–Coordinate ascent

- Now, the dual optimization problem that we want to solve is

$$\max_{\alpha} L_d(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i y_i \underbrace{(x_i)^T x_j}_{<x_i, x_j>} y_j \alpha_j$$
$$s.t. \alpha_i \geq 0, i = 1, ...N \quad \sum_{i}^{N} \alpha_i y_i = 0 \tag{13}$$

- Can we make any progress just taking only one coordinate α_i and fixed α_j with $j \neq i$? The answer is no! because the constraint $\sum_i^N \alpha_i y_i = 0$ ensures that $\alpha_i y_i = -\sum_{j \neq i} \alpha_j y_j$, that is $\alpha_i = -y_i \sum_{j \neq i} \alpha_j y_j$.
- This indicates that α_i is exactly determined by the other α_j, and if we were to hold α_j fixed. Then, we can't make any change to α_i without violating the constraint in the optimization problem.

Algorithms to solve the dual problems (13)–Coordinate ascent

Algorithm 3 Coordinate Ascent

Initialize $\alpha^{(0)} \in \mathbb{R}^N$
for $t = 0 : maxIter$ do **do**
    Sample two coordinate i, j randomly from $1, ...N$.
    Optimize $L_d(\alpha)$ w.r.t that coordinate
    $(u_i^*, u_j^*) = argmax_{u_i, u_j)} L_d(\alpha_1^t, ...\alpha_{i-1}^t, u_i, \alpha_{i+1}^t, ...\alpha_{j-1}^t, u_j, \alpha_{j+1}^t, ...\alpha_N^t)$
    $(\alpha_i^{t+1}, \alpha_j^{t+1}) = (u_i^*, u_j^*)$
    $\alpha_k^{t+1} = \alpha_k^t$, for all $k \neq i, j$.
end for
OUTPUT: $\alpha^{maxIter}$

- The $argmax_{(u_i, u_j)}$ step is very effective. It can be transformed into solve univariate quadratic programming under the constraints that $u_i y_i + u_j y_j = c$ and $u_i, u_j \geq 0$. .

Summary: Pedestrian Detection with HOG

INPUT: A image

Pretreatment

Gamma Transformation → Convolution for gradient and angle(Sobel)

Get ROI

Slide Window

Describe a Block

HOG Description: Part a image(Get cell) →
Statistic Histogram of oriented gradients →
Build the feature of cell →
Build the feature of image

Classification

SVM: Training(Before using) → Using for classification

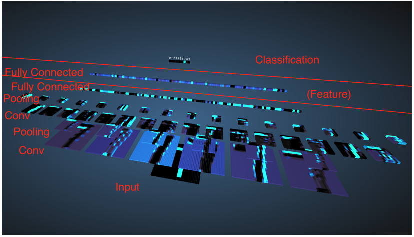
OUTPUT: Blocks Include Human

Other method: Deep Learning(CNN) Method

Convolution Neural Network(CNN) in pedestrian detection

As CNN can also be expressed as feature - classification method. It can also be used in pedestriain detection. Different from Haar or HOG, the method of getting feature is not confirmed. It will be determine during the training processing.

Recent years, based on regions of CNN, R-CNN have wildly used in pedestrian detection problem. The most famous of them is fast R-CNN



More introduction and source code of fast R-CNN:
`https://github.com/rbgirshick/fast-rcnn`

The sample CNN of Handwritten digit recognition:
`http://scs.ryerson.ca/%7Eaharley/vis/conv/`

Some problem for discussing

1) About ROI method: Obviously, slide window method is too slow. So how can we get the ROIs faster?

2) About ROI method: If only part of the selected region is a person, how to choose a larger region to fully include pedestrian?

3) About Convolution: How to establsh Fast Fourier Transformation(FFT) algorithm?

4) About SVM: As Training of SVM(Find the support vector) is a NP hard problem, how can we make this processing faster?

5) About SVM: If a group of data cannot be part with a line, how to find a classifier for classification?

6) How to build a pedestrin detection system with Haar and adaboost?

7) How to build a pedestrin detection system with fast R-CNN?