Unit 3. Tracking, Recognition, Detection

- **[Tracking]**
  - How doe LK alignment and KLT Tracker work ?
  - How can we derive the formulation to update warping parameters from the objective function to mini mise?
  - How does Mean-shift tracking work?
- **[ML Basics and Recognition]**
  - What is the similarity or difference between image classification, object detection, semantic segmenta tion, and instance segmentation?
  - How can we devise the statistical learning framework for training and testing?
  - What kind of classifiers used for recognition? How do they work? How good/bad are they?
- **[Detection]**
  - How does an HoG descriptor combined with Linear SVM work for the pedestrian detection?
  - How does an integral image work for Viola-Jones face detector? How can we perform boosting?

3–1: Tracking

Image Alignment

2D image transformation

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p})$$

2D image coordinate

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Parameters of the transformation

$$\boldsymbol{p} = \{p_1, \ldots, p_N\}$$

Warped image

$$I(\boldsymbol{x}') = I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$$

Pixel value at a coordinate

Translation

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} x + p_1 \\ y + p_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & p_1 \\ 0 & 1 & p_2 \end{bmatrix}}_{\text{transform}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\text{coordinate}}$$

Affine

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} p_1 x + p_2 y + p_3 \\ p_4 x + p_5 y + p_6 \end{bmatrix} = \underbrace{\begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix}}_{\text{affine transform}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\text{coordinate}}$$

can be written in matrix form when linear
affine warp matrix can also be 3x3 when last row is [0 0 1]

LK alignment

Strategy:

$$\sum_{\boldsymbol{x}} [I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p} + \boxed{\Delta\boldsymbol{p}})) - T(\boldsymbol{x})]^2$$

Assume known approximate solution
Solve for increment

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(\boldsymbol{x})\right]^2$$

Taylor series approximation
Linearize

Solution:

$$\Delta\boldsymbol{p} = H^{-1} \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]^{\top} [T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]$$

Solution to least square s approximation

$$H = \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]^{\top} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]$$

Hessian

Called Gauss-Newton gradient decent non-l

—known approximate solution, solve for increment
—Linear approximation

Alignment
Warp —> compute error image —> gradient —> Jacobian —> Hessian —> Δp —> update parameters

KLT-tracker

KLT Algorithm

1. Find corners satisfying $\min(\lambda_1, \lambda_2) > \lambda$
2. For each corner compute displacement to next frame using the Lucas-Kanade method
3. Store displacement of each corner, update corner position
4. (optional) Add more corner points every M frames using 1
5. Repeat 2 to 3 (4)
6. Returns long trajectories for each corner point

Mean-shift Tracking

Mean-shift Tracking (for images)

```
Initialize $\boldsymbol{x}$
While $v(\boldsymbol{x}) > \epsilon$
    1. Compute mean-shift
        $m(\boldsymbol{x}) = \frac{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s) w(\boldsymbol{x}_s) \boldsymbol{x}_s}{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s) w(\boldsymbol{x}_s)}$
        $v(\boldsymbol{x}) = m(\boldsymbol{x}) - \boldsymbol{x}$
    2. Update $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{v}(\boldsymbol{x})$
```

Mean-shift Tracking.

For each frame:

1. **Initialize location** $\boldsymbol{y}_0$
   Compute $\boldsymbol{q}$
   Compute $\boldsymbol{p}(\boldsymbol{y}_0)$
2. **Derive weights** w_n
3. **Shift to new candidate location (mean shift)** $\boldsymbol{y}_1$
4. **Compute** $\boldsymbol{p}(\boldsymbol{y}_1)$
5. **If** $\|\boldsymbol{y}_0 - \boldsymbol{y}_1\| < \epsilon$ **return**
   Otherwise $\boldsymbol{y}_0 \leftarrow \boldsymbol{y}_1$ **and go back to 2**

Question 3 (a):

a) This question is about **Tracking.**

[4 marks]

Kanade-Lukas-Tomasi (KLT) Tracker solves the objective function below:

$$\min_{\boldsymbol{p}} \sum_{\boldsymbol{x}} [I(\boldsymbol{W}(\boldsymbol{x};\boldsymbol{p})) - T(\boldsymbol{x})]^2$$

where $\boldsymbol{p}$ is warp parameters, I is image intensity, $\boldsymbol{W}$ is a warping function, T is template image intensity, and $\boldsymbol{x}$ is pixel coordinate. The figure below shows the procedure of KLT Tracker. Fill out the blanks with the proper notations.

(4 marks)

1. Warp image [$I(\boldsymbol{W}(\boldsymbol{x};\boldsymbol{p}))$]
2. Compute error image [$I(\boldsymbol{W}(\boldsymbol{x};\boldsymbol{p})) - T(\boldsymbol{x})$]
3. Compute gradient [$I(\boldsymbol{x}')$ ($\boldsymbol{x}'$: coordinates of the warped image)]
4. Evaluate Jacobian [ ]
5. Compute Hessian [ ]
6. Compute $\Delta\boldsymbol{p}$ = [ ]
7. Update parameters [ ]

① $\frac{\partial W}{\partial p}$

② $\sum_x (\nabla I \frac{\partial W}{\partial p})^T (\nabla I \frac{\partial W}{\partial p})$

③ $H^{-1} \sum_x (\nabla I \frac{\partial W}{\partial p})^T (T(x) - I(W(x;p)))$

④ $p \leftarrow p + \Delta p$

$\min \sum_x [I(W(x;p)) - T(x)]^2$

$\downarrow$

$\sum_x [I(W(x;p+\Delta p)) - T(x)]^2$

$\sum_x [I(W(x;p)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x)]^2$

$\sum_x [\nabla I \frac{\partial W}{\partial p} \Delta p - (T(x) - I(W(x;p)))]^2$

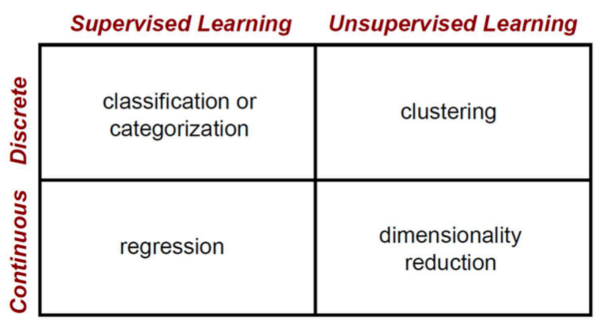
$\| A \quad x \quad - \quad b \|^2 \Rightarrow (A^T A)^{-1} A^T b$

$\Delta p = H^{-1} \sum_x (\nabla I \frac{\partial W}{\partial p})^T (T(x) - I(W(x;p)))$

$H = \sum_x [\nabla I \frac{\partial W}{\partial p}]^T [\nabla I \frac{\partial W}{\partial p}]$.

3–3: classification

Machine learning problems



	Supervised Learning	Unsupervised Learning
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

Generalization: how well does a learned model generalize from the data it was trained on to a new test set?

Object recognition: identifying objects in digital photographs (4)
– Image classification
– Object detection
– Semantic segmentation
– Instance segmentation

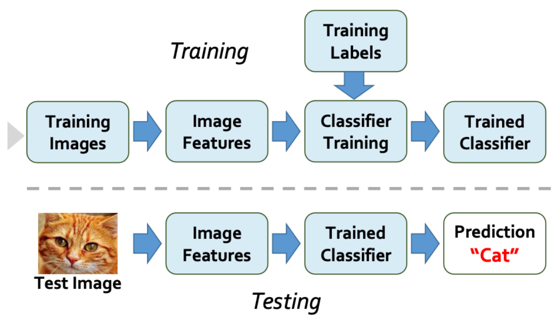
Image classification
Identifying what is in the image and the associated level of confidence, can be binary–label or multi–label classification

Object detection
Localizing and classifying one or more objects in an image, it includes object localization and image classification

Semantic segmentation
Assign a label to every pixel, treat multiple objects of the same class as a single entity.

Instance segmentation
Assign a label to every pixel but treat multiple objects of the same class as distinct instance.

Machine Learning Framework



Bag of features:

1. **Extract local features**
2. Learn **“visual vocabulary”**
3. **Quantize local features** using “visual vocabulary”
4. Represent images by **frequencies of “visual words”**

Trainable Classifiers

- Nearest neighbor (NN)
- Linear classifier
- SVM (support vector machine)

NN classifier
f(x) — label of the training example nearest to x
KNN classifier
For a new point, find the k closest points from training data, then vote for class label with labels of the k points

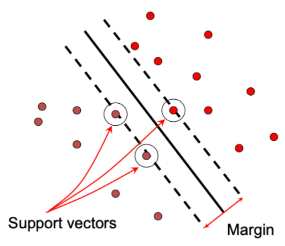
Linear classifier
- find a linear function to separate the classes

NN v.s. Linear

	NN classifier	Linear classifier
Pros	1. Simple to implement 2. No restriction to number of classes 3. Nonparametric method 4. Decision boundaries not need to be linear	1. Fast at test time 2. Low-dimensional parametric representation
Cons	1. Slow at test time 2. Need good distance function	1. Only work for 2 classes 2. How to train the classifier? 3. Cannot deal with non-linearly separable issues

Support vector machines

Idea: find a hyperplane that maximizes the margin between the positive and negative examples



$\mathbf{x}_i$ positive $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support vectors, $\quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and hyperplane: $\quad \dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

Therefore, the margin is $2 / \|\mathbf{w}\|$

Quadratic optimization problem

$$\boxed{\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1}$$

Non-separable data:

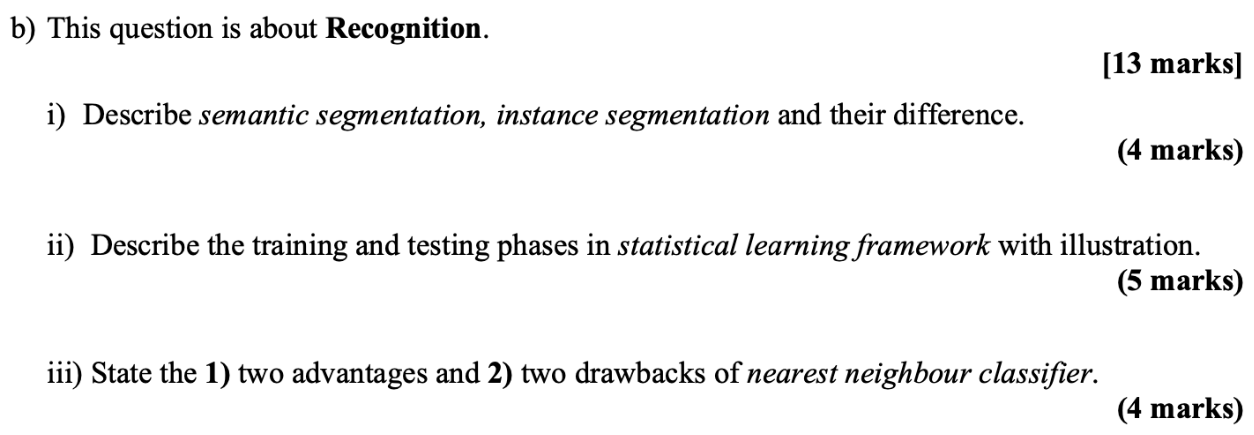
$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(wx_i+b))$$

Non-linear SVMs
Idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable

SVMs: Pros and Cons
Pros
- **non-linear SVM** is powerful, flexible
- Training is convex optimization, can find the **global optimal solution**
- SVMs **work very effectively in practice**, even with a small amount of training data

Cons
- **no direct multi-class** SVM, can only combine two-class SVMs
- **Memory, computation**

Question 3 (b)


b) This question is about **Recognition**.

[13 marks]

i) Describe *semantic segmentation, instance segmentation* and their difference.

(4 marks)

ii) Describe the training and testing phases in *statistical learning framework* with illustration.

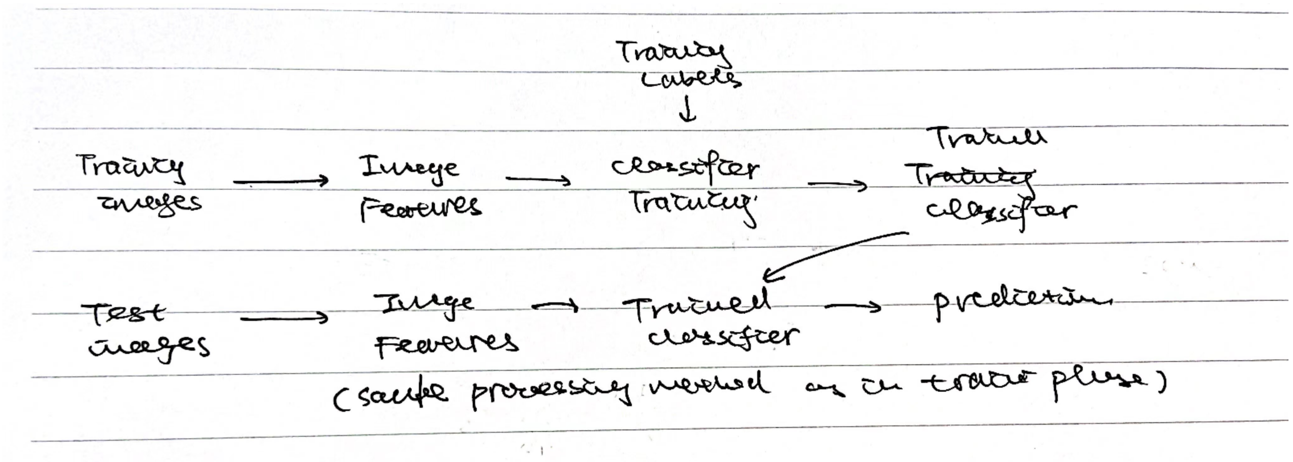
(5 marks)

iii) State the **1)** two advantages and **2)** two drawbacks of *nearest neighbour classifier*.

(4 marks)

(1) Semantic segmentation assigns every pixel a label, it treats multiple objects with the same class as a single entity. Instance segmentation also assigns every pixel as label, however it treats multiple objects with the same class as distinct instances. Normally, instance segmentation is harder than semantic segmentation.

(2)



(3) Advantages: simple to implement, nonparametric methods, can work for multi-class classification, decision boundaries are not linear. Disadvantages: slow at test time, need good distance function.

3–5: Detection

General Process of Object Detection

- Specify object model: what are the object parameters
- Generate hypotheses: Propose an alignment of the model to the image
- Score hypotheses: Mainly gradient-based features, usually based on summary representation, many classifiers
- Resolve Detections: rescore each proposed object based on the whole set

Specifying an object model

1. Statistical template in bounding box
2. Articulated parts model
3. Hybrid template model
4. 3D-ish model

Dala-Triggs pedestrian detector

1. Extract fixed-sized window at each position and scale
2. Compute HOG features within each window
3. Score the window with a linear SVM classifier
4. Perform non-maxima suppression to remove overlapping detections with lower scores

Strengths

- work very well for **non-deformable** objects
- **Fast** detection

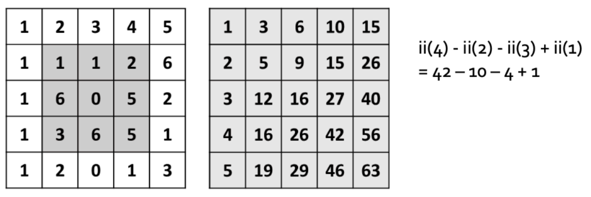
Weaknesses

- not good for **highly deformable objects**
- Not robust to **occlusion**
- Requires **lots of training data**

The Viola/Jones Face Detector

1. Integral images for fast feature evaluation
2. Boosting for feature selection
3. Attentional cascade for fast non-face window rejection

Integral Images



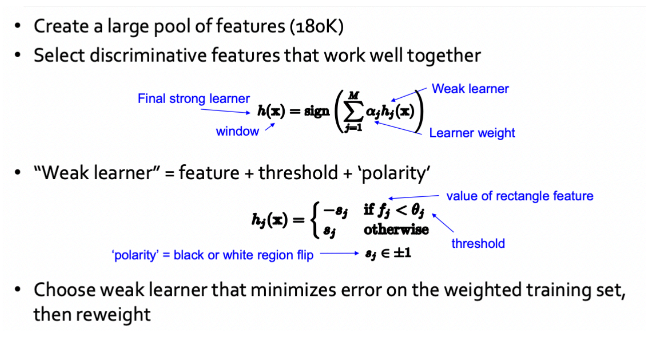
1	2	3	4	5
1	1	1	2	6
1	6	0	5	2
1	3	6	5	1
1	2	0	1	3

1	3	6	10	15
2	5	9	15	26
3	12	16	27	40
4	16	26	42	56
5	19	29	46	63

$ii(4) - ii(2) - ii(3) + ii(1)$
$= 42 - 10 - 4 + 1$

Boosting for feature selection

1. Initially weight each training example equally
2. In each boosting round:
    1. Find the **weak classifier**, trained for each feature, that achieves the **lowest weighted training error**
    2. **Raise the weights** of training examples **misclassified** by current weak classifier
3. Compute final classier as **linear combination of all weak classifier**



- Create a large pool of features (180K)
- Select discriminative features that work well together

$$h(\mathbf{x}) = \text{sign}\left(\sum_{j=1}^{M} \alpha_j h_j(\mathbf{x})\right)$$

Final strong learner
window
Weak learner
Learner weight

- "Weak learner" = feature + threshold + 'polarity'

$$h_j(\mathbf{x}) = \begin{cases} -s_j & \text{if } f_j < \theta_j \\ s_j & \text{otherwise} \end{cases}$$

$$s_j \in \pm 1$$

value of rectangle feature
threshold
'polarity' = black or white region flip

- Choose weak learner that minimizes error on the weighted training set, then reweight

Boosting: Pros and Cons

Advantages of boosting

– integrates classifier training with **feature selection**

– Complexity of training is **linear**

– **Flexibility** in the choice of weak learners

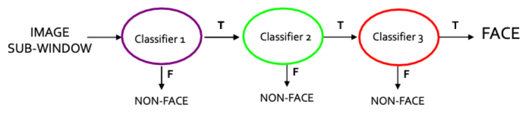
– Testing is fast

Disadvantages

– needs **many training examples**

– Training is slow

Attentional cascade for fast non–face window rejection



Train the cascade

1. Set target detection and false positive rates for each stage
2. Keeps adding features to the current stage until its target rates have been met
3. If the overall false positive rate is not low enough, add another stage
4. Use false positive from current stage as the **negative training examples** for the next stage

Question 3 (c):

c) This question is about **detection**.

[8 marks]

i) In Viola/Jones face detector, integral images are used for fast feature evaluation during testing. Find **1)** the integral image of the figure below and compute **2)** the sum of pixels in the grey region <u>based on the integral image</u>. (Show your calculations)

(4 marks)

5	4	3	8	3
3	1	1	2	6
1	6	0	5	7
1	3	6	5	9
1	2	2	1	3

ii) Viola/Jones face detector employs boosting for feature selection. Describe the process, with illustration, of *boosting round 1*.

(4 marks)

(1) ① $\overset{1}{\boxed{5}}$ 9 12 $\overset{2}{\boxed{20}}$ 23

8 13 17 27 36

9 20 24 39 55

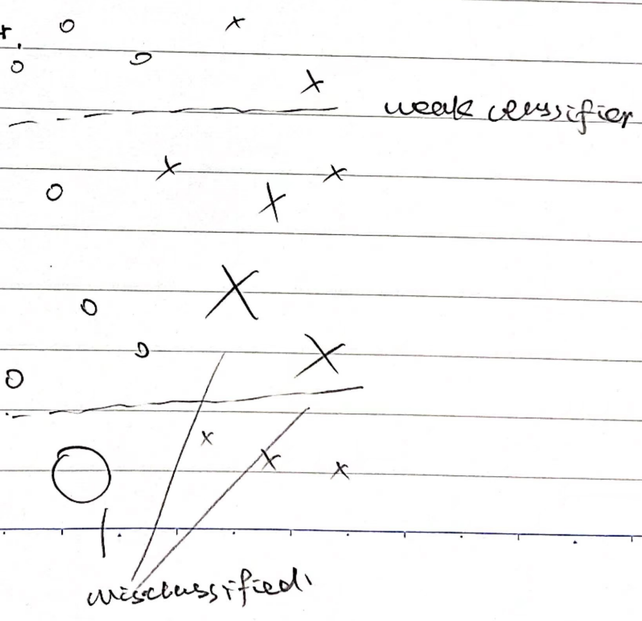
$\overset{3}{\boxed{10}}$ 24 34 $\overset{4}{\boxed{54}}$ 79

11 27 39 60 88

(2) ② ii(4) + ii(1) − ii(2) − ii(3) = 54 + 5 − 30 = 29.

(2) ① Find the weak classifier trained for each feature that achieves the lowest weighted training error

② raise the weight of examples misclassified by current weak classifier



weak classifier
misclassified