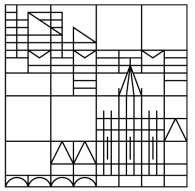
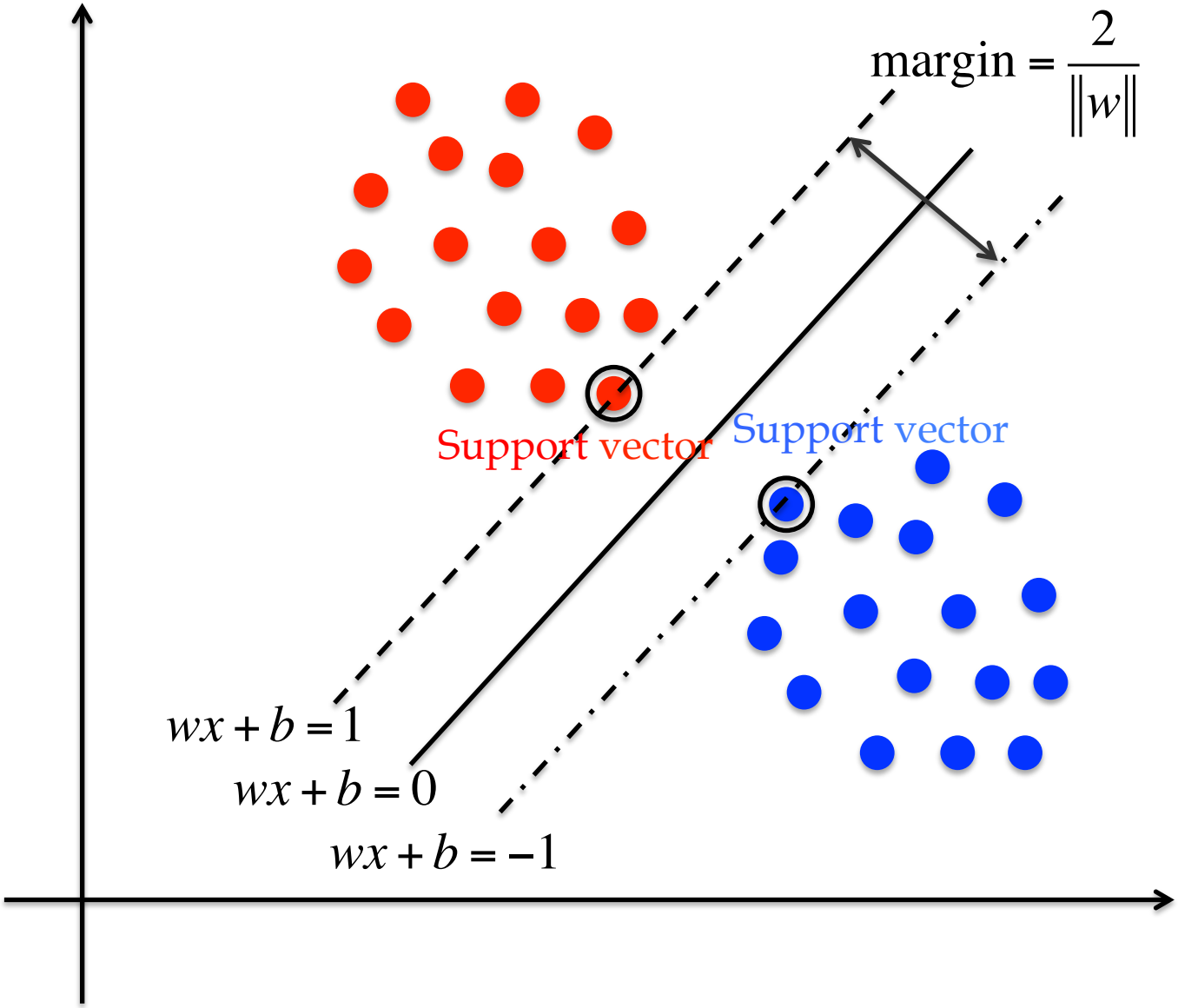
Universität Konstanz



Lecture 5 Support Vector Machine (SVM) – Part 2

Dr. Hanhe Lin
Dept. of Computer and Information Science
University of Konstanz

SVM


margin = $\frac{2}{\|w\|}$
Support vector
Support vector
$wx + b = 1$
$wx + b = 0$
$wx + b = -1$

Primal form

Primal form of SVM optimization problem is:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi^{(i)}$$

$$\text{s.t. } y^{(i)}(wx^{(i)} + b) \geq 1 - \xi^{(i)}, \xi^{(i)} \geq 0 \quad \forall i = 1, \ldots, m$$

Dual form

- Solve by a bunch of algebra and calculus, the dual form of SVM is:

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha^{(i)} \alpha^{(j)} y^{(i)} y^{(j)} x^{(i)} x^{(j)} - \sum_{i=1}^{m} \alpha^{(i)}$$

$$\text{s.t. } 0 \le \alpha^{(i)} \le C, \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} = 0 \ \forall i = 1, \ldots, m$$

- The decision can also be rewritten as:

$$w = \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} x^{(i)}$$

$$f(x) = wx + b \Rightarrow f(x) = \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} x^{(i)} x + b$$

Kernel SVM

Classifier: $f(x) = \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} \Phi(x^{(i)}) \Phi(x) + b$

Optimization:

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha^{(i)} \alpha^{(j)} y^{(i)} y^{(j)} \Phi(x^{(i)}) \Phi(x^{(j)}) - \sum_{i=1}^{m} \alpha^{(i)}$$

$$\text{s.t. } 0 \le \alpha^{(i)} \le C, \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} = 0 \;\; \forall i = 1, \ldots, m$$

From logistic regression to SVM

- Given training examples $(x^{(i)}, y^{(i)})$, SVM aims to find an optimal hyperplane so that:

$$f(x^{(i)}) = wx^{(i)} + b \begin{cases} \geq 1 \text{ if } y^{(i)} = 1 \\ \leq -1 \text{ if } y^{(i)} = -1 \end{cases}$$

- Which is equivalent to minimizing the following loss function

$$L(w,b) = \frac{1}{m}\sum_{i=1}^{m} \max(0, 1 - y^{(i)} \underbrace{(wx^{(i)} + b)}_{f(x^{(i)})}) + \frac{\lambda}{2}\|w\|^2$$

- Hinge loss: $\max(0, 1 - y^{(i)} f(x^{(i)}))$

$y^{(i)} f(x^{(i)}) \geq 1$ No contribution to loss

$y^{(i)} f(x^{(i)}) < 1$ Contributes to loss

Gradient computing

$$\frac{\partial}{\partial w} L(w,b) = \frac{1}{m} \sum_{i=1}^{m} \begin{cases} -y^{(i)} x^{(i)} & \text{if } 1\text{-}y^{(i)} f(x^{(i)}) > 0 \\ 0 & \text{else} \end{cases} + \lambda w$$

$$\frac{\partial}{\partial b} L(w,b) = \frac{1}{m} \sum_{i=1}^{m} \begin{cases} -y^{(i)} & \text{if } 1\text{-}y^{(i)} f(x^{(i)}) > 0 \\ 0 & \text{else} \end{cases}$$

Multi-class SVM – notation

Let weight $W = \begin{bmatrix} w^{(1)} & w^{(2)} & \dots & w^{(K)} \end{bmatrix} \in \Re^{n \times K}$, bias $b = \begin{bmatrix} b^{(1)} \\ b^{(2)} \\ \vdots \\ b^{(K)} \end{bmatrix} \in \Re^{K \times 1}$ be the parameters of multi-class SVM

Hypothesis: $h_{W,b}(x) = Wx + b$

The decision function of multi-class SVM is defined:

$$f(x) = \underset{k}{\operatorname{argmax}}(w^{(k)}x + b^{(k)}) \quad k = 1, 2, \dots, K$$

Multi-class SVM – loss function

$$L(W,b) = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1, j\neq y^{(i)}}^{K} \max(0, (w^{(j)}x^{(i)} + b^{(j)}) - (w^{(y^{(i)})}x^{(i)} + b^{(y^{(i)})}) + \underbrace{\Delta}_{\text{margin}})$$

Example: assume we have three classes that predict the scores $f(x^{(1)}) = \begin{bmatrix} 5 \\ -13 \\ 12 \end{bmatrix}$

and $f(x^{(2)}) = \begin{bmatrix} -7 \\ 6 \\ 3 \end{bmatrix}$ w.r.t. $y^{(1)} = 1$ and $y^{(2)} = 2$. Also assume $\Delta = 10$, then we have:

$$L(W,b) = \underbrace{\max(0,-13-5+10) + \max(0,12-5+10)}_{x^{(1)}} + \underbrace{\max(0,-7-6+10) + \max(0,3-6+10)}_{x^{(2)}}$$

$$= 0 + 17 + 0 + 7$$

$$= 24$$

Multi-class SVM – loss function

Add regularization term:

$$L(W,b) = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1, j\neq y^{(i)}}^{K} \max(0, (w^{(j)}x^{(i)} + b^{(j)}) - (w^{(y^{(i)})}x^{(i)} + b^{(y^{(i)})}) + \underbrace{\Delta}_{\text{margin}}) + \frac{\lambda}{2}\sum_{l=1}^{K}\left\|w^{(l)}\right\|^2$$

Partial derivative:

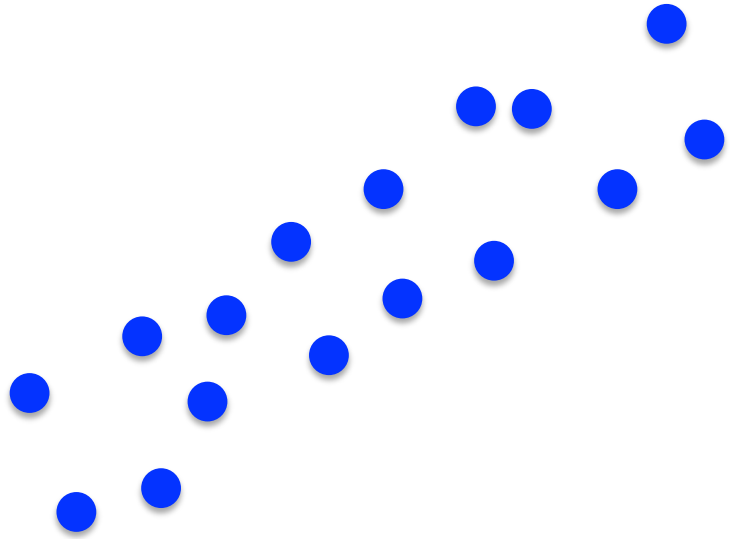
$$\frac{\partial L}{\partial w^{(k)}} = \lambda w^{(k)} + \frac{1}{m}\sum_{i=1}^{m}\begin{cases} -x^{(i)} & \text{if } (w^{(j)}x^{(i)} + b^{(j)}) - (w^{(y^{(i)})}x^{(i)} + b^{(y^{(i)})}) + \Delta > 0 \text{ and } y^{(i)} = k \\ x^{(i)} & \text{if } (w^{(j)}x^{(i)} + b^{(j)}) - (w^{(y^{(i)})}x^{(i)} + b^{(y^{(i)})}) + \Delta > 0 \text{ and } j = k \\ 0 & \text{if } (w^{(j)}x^{(i)} + b^{(j)}) - (w^{(y^{(i)})}x^{(i)} + b^{(y^{(i)})}) + \Delta \leq 0 \end{cases}$$

$$\frac{\partial L}{\partial b^{(k)}} = \frac{1}{m}\sum_{i=1}^{m}\begin{cases} -1 \text{ if } (w^{(j)}x^{(i)} + b^{(j)}) - (w^{(y^{(i)})}x^{(i)} + b^{(y^{(i)})}) + \Delta > 0 \text{ and } y^{(i)} = k \\ 1 \text{ if } (w^{(j)}x^{(i)} + b^{(j)}) - (w^{(y^{(i)})}x^{(i)} + b^{(y^{(i)})}) + \Delta > 0 \text{ and } j = k \\ 0 \text{ if } (w^{(j)}x^{(i)} + b^{(j)}) - (w^{(y^{(i)})}x^{(i)} + b^{(y^{(i)})}) + \Delta \leq 0 \end{cases}$$

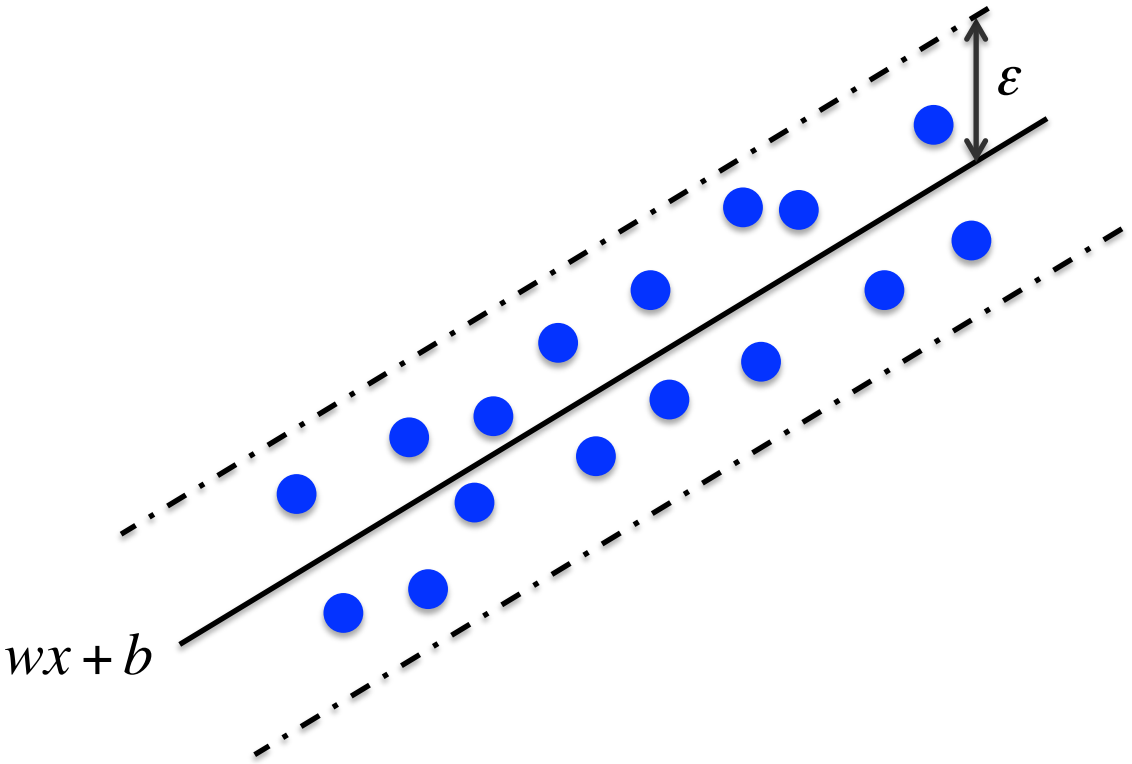
Softmax vs. multi-class SVM

- Softmax classifier provides “probabilities” for each class, while SVM provides “scores” for each class
- In practice, SVM and softmax are usually comparable
- Softmax classifier is more commonly applied in deep learning classification problems

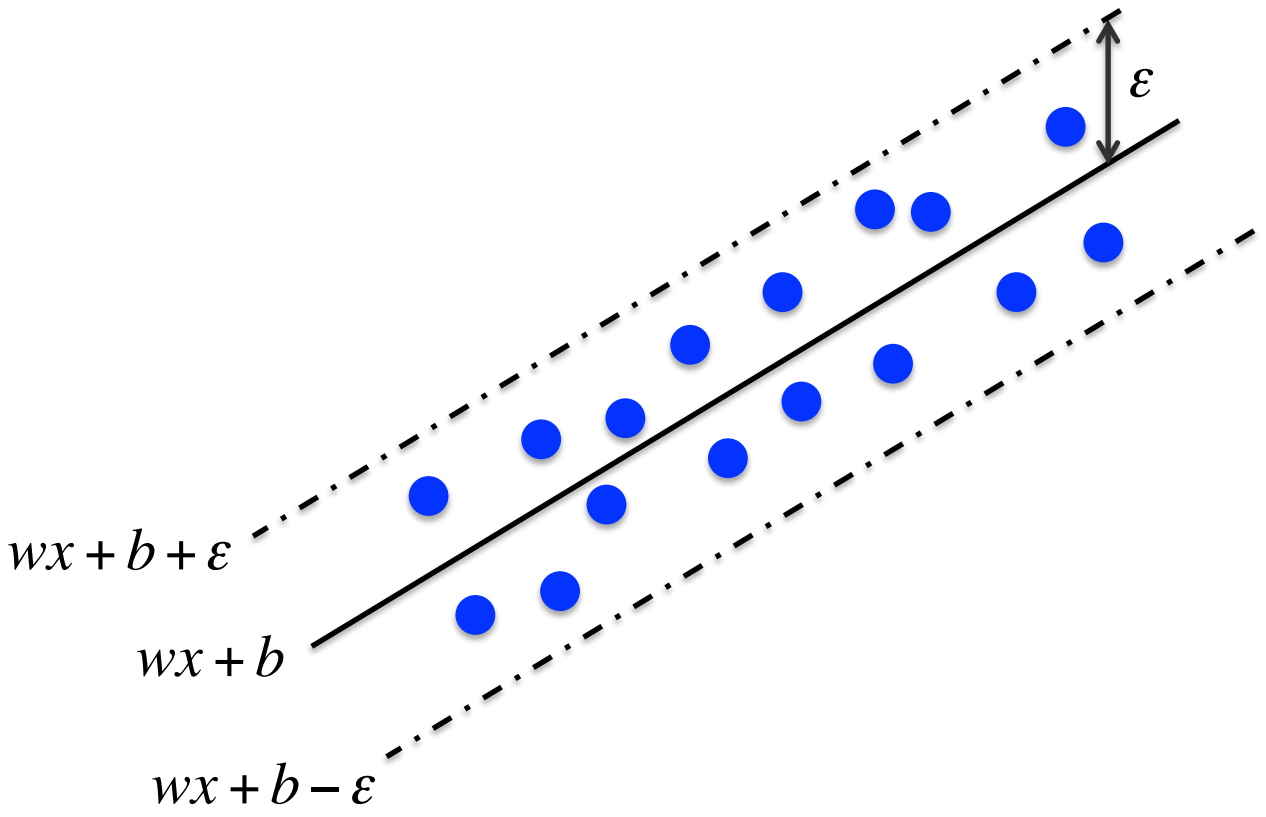
Support Vector Regression (SVR) – hard margin



Support Vector Regression (SVR) – hard margin



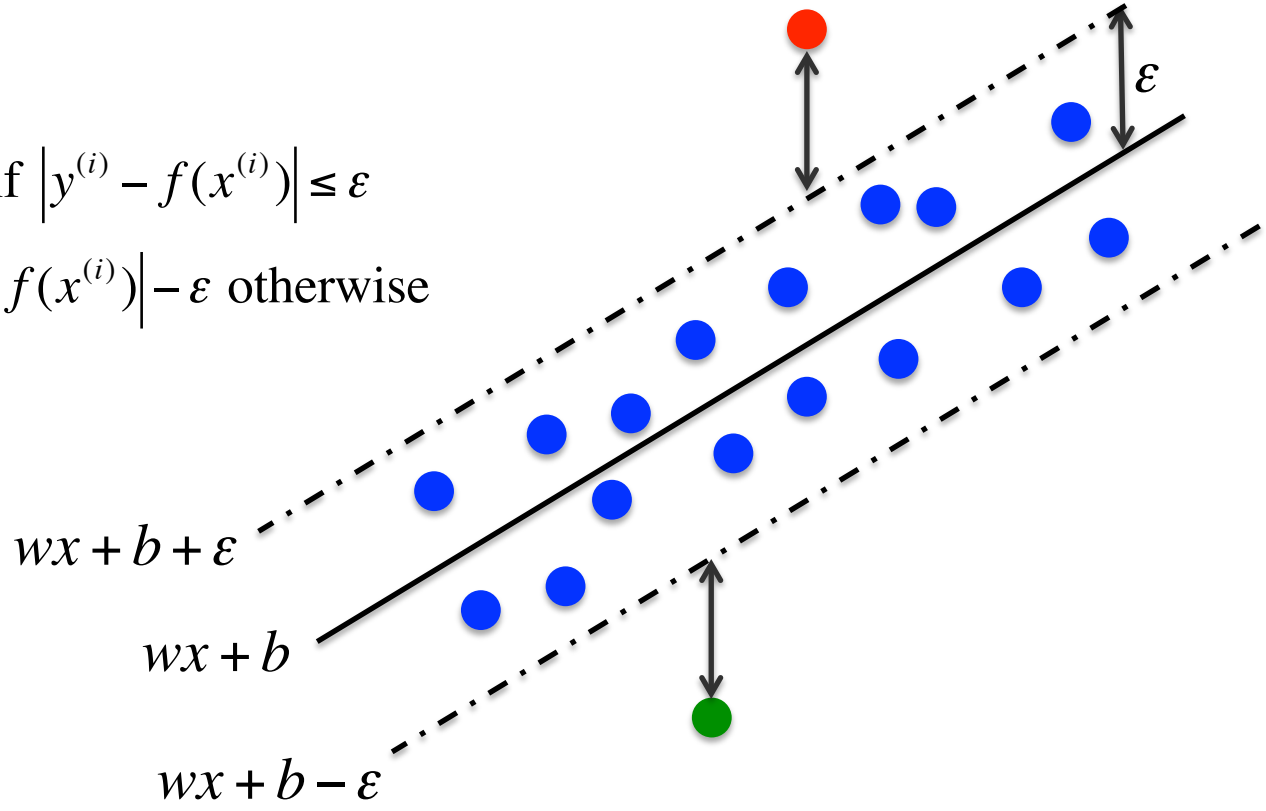
Support Vector Regression (SVR) – hard margin



SVR – soft margin

ε-insensitive loss function:

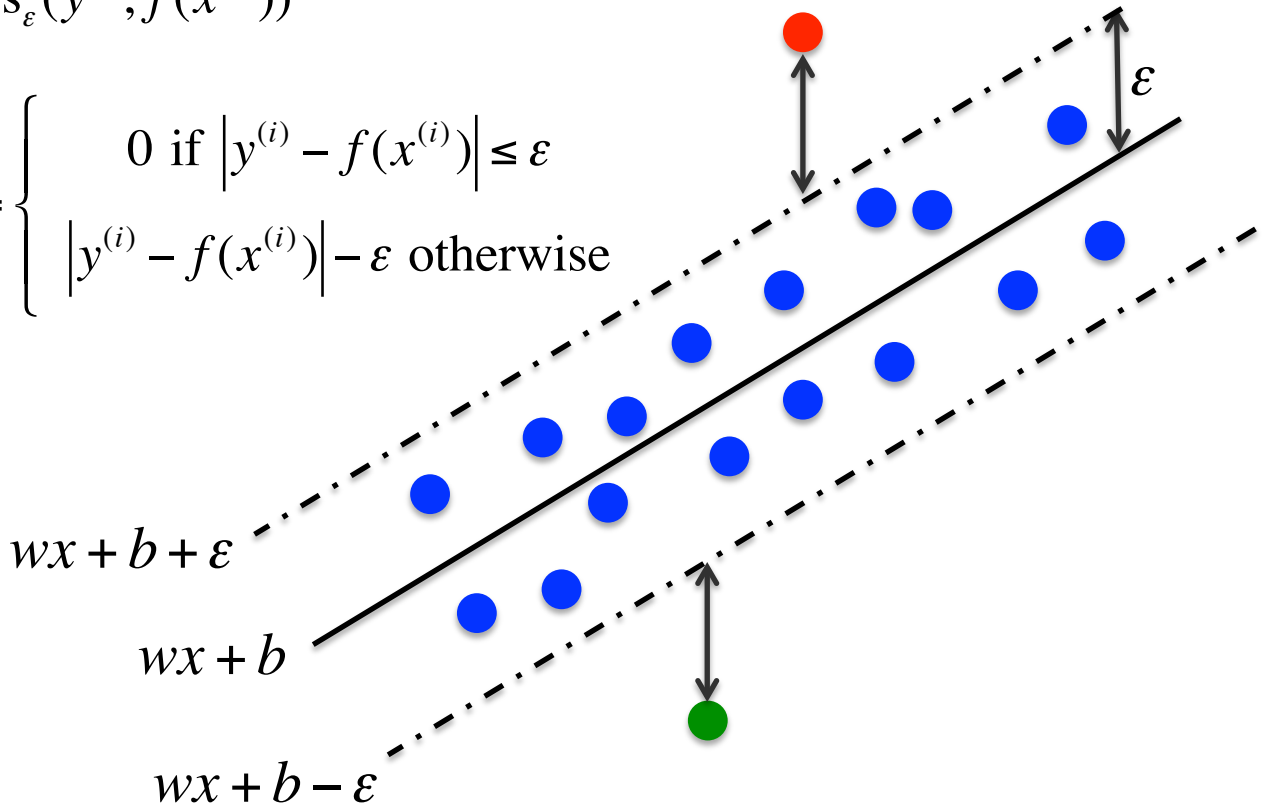
$$\text{Loss}_{\varepsilon}(y^{(i)}, f(x^{(i)})) = \begin{cases} 0 \text{ if } \left|y^{(i)} - f(x^{(i)})\right| \le \varepsilon \\ \left|y^{(i)} - f(x^{(i)})\right| - \varepsilon \text{ otherwise} \end{cases}$$



SVR – loss function

$$L(w,b) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \text{Loss}_\varepsilon(y^{(i)}, f(x^{(i)}))$$

$$\text{where Loss}_\varepsilon(y^{(i)}, f(x^{(i)})) = \begin{cases} 0 \text{ if } \left|y^{(i)} - f(x^{(i)})\right| \le \varepsilon \\ \left|y^{(i)} - f(x^{(i)})\right| - \varepsilon \text{ otherwise} \end{cases}$$



SVR – primal form

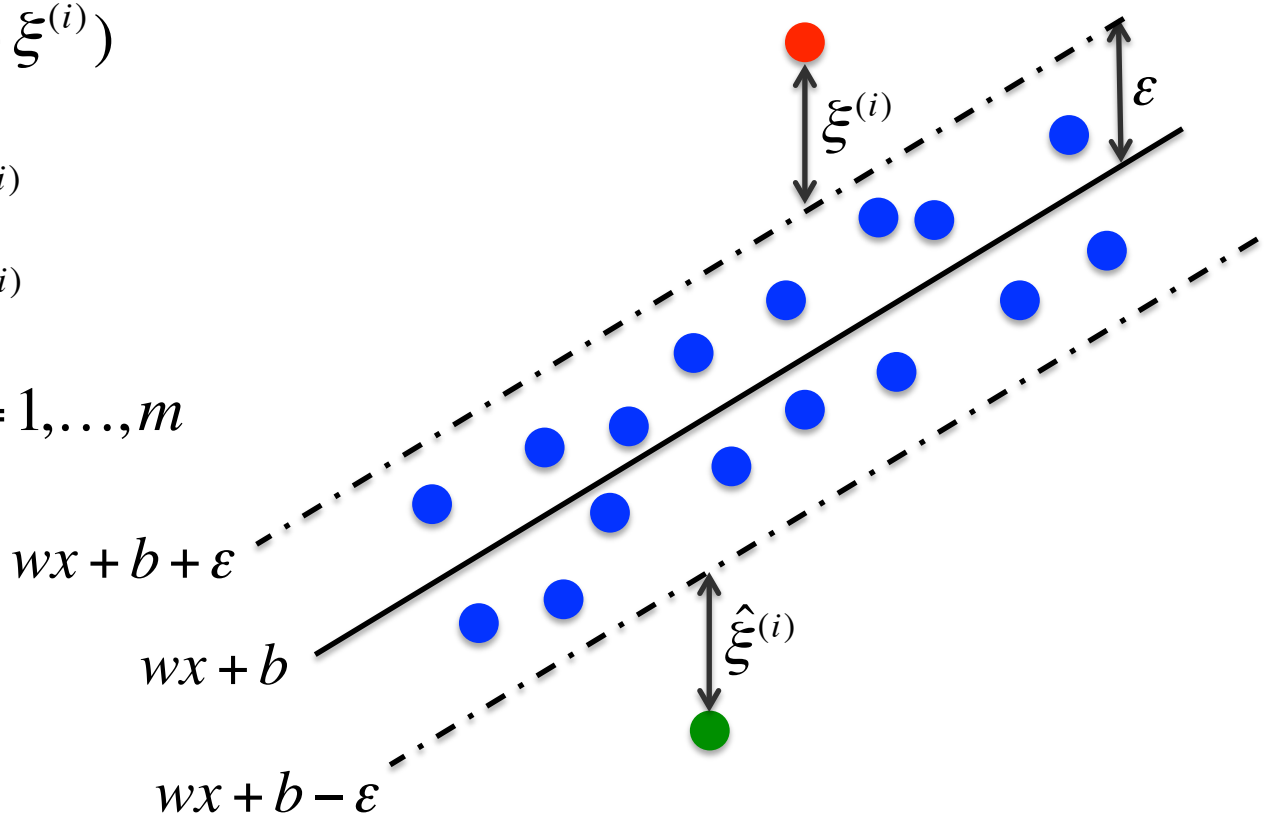
Introducing slack variables $\hat{\xi}^{(i)}, \xi^{(i)} \geq 0$, we have:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}(\hat{\xi}^{(i)} + \xi^{(i)})$$

$$\text{s.t. } y^{(i)} - f(x^{(i)}) \leq \varepsilon + \hat{\xi}^{(i)}$$

$$f(x^{(i)}) - y^{(i)} \leq \varepsilon + \xi^{(i)}$$

$$\hat{\xi}^{(i)} \geq 0, \xi^{(i)} \geq 0, \forall i = 1, \ldots, m$$



SVR vs. SVM

- SVR is the extension of SVM, thus the optimization algorithm for SVM can be applied to SVR directly
- Likewise, “kernel trick” can also be applied to SVR
- Q: how many hyper-parameters should I tune if I use Gaussian kernel SVR?

SVR vs. SVM

- SVR is the extension of SVM, thus the optimization algorithm for SVM can be applied to SVR directly
- Likewise, "kernel trick" can also be applied to SVR
- Q: how many hyper-parameters should I tune if I use Gaussian kernel SVR?

Three parameters, namely, C, σ, and ε.

SVM toolbox

- Matlab SVM toolbox
- Libsvm
- SVMlight
- …

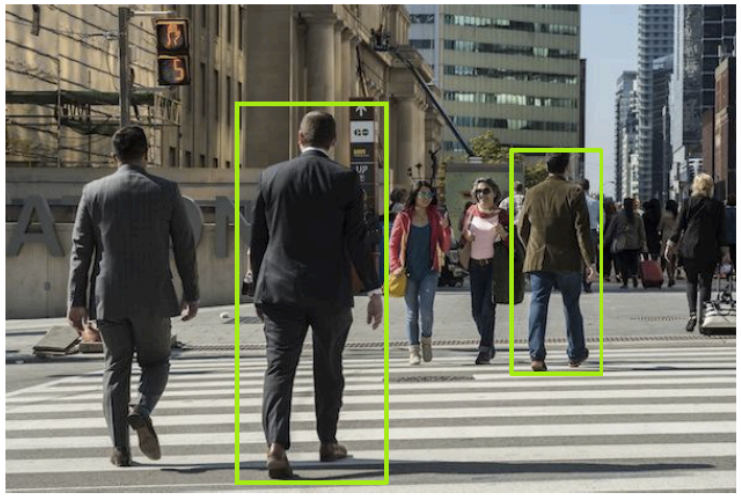
SVM – takeaway

- SVM was originally proposed by Boser, Guyon and Vapnik in 1992 and gained increasing popularity in late 1990s
- Maximizing the margin of a linear separator is a good training criteria
- SVMs learn a max-margin linear classifier
- The SVM optimization problem can be solved with Quadratic Programming (QP) solvers
- Learned decision boundary is defined by its support vectors
- SVM can be applied to complex data types beyond feature vectors (e.g. graphs, sequences, relational data) by designing kernel functions for such data
- SVM is a convex problem, thus we have global optimal solution. However, the computational cost increases along with the number of training examples. Therefore, more efficient optimization algorithms are proposed, e.g. Sequential Minimum Optimization (SMO)
- Tuning SVMs remains a black art: selecting a specific kernel and parameters is usually done in a try-and-see manner

SVM application: pedestrian detection

Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005. (33000+ citations)

Challenges in pedestrian detection



- A wide variation of pedestrian
- Different scales of pedestrians in different images

Pedestrian detection system

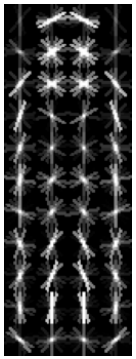
Positive examples

Negative examples

Pedestrian detection with HOG

- Train a pedestrian template using a linear SVM
- At test time, convolve feature map with template
- Find local maxima of response
- For multi-scale detection, repeat over multiple levels of a HOG pyramid

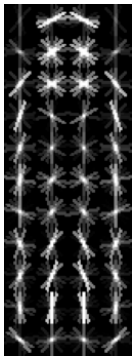
Template



Pedestrian detection with HOG

- Train a pedestrian template using a linear SVM
- At test time, convolve feature map with template
- Find local maxima of response
- For multi-scale detection, repeat over multiple levels of a HOG pyramid

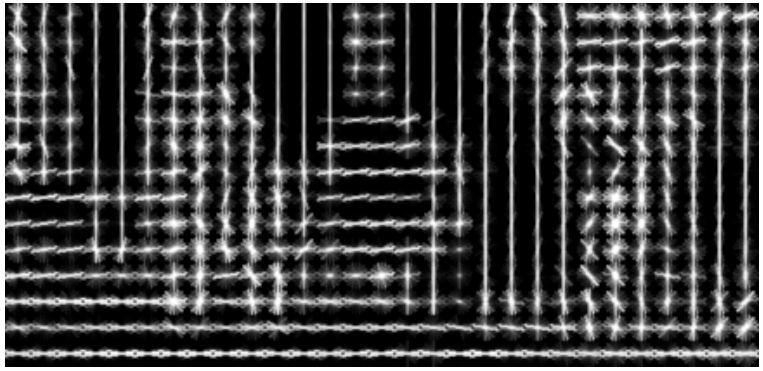
Template



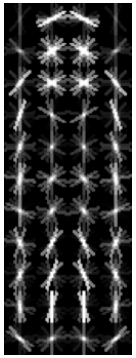
Pedestrian detection with HOG

- Train a pedestrian template using a linear SVM
- At test time, convolve feature map with template
- Find local maxima of response
- For multi-scale detection, repeat over multiple levels of a HOG pyramid

HOG feature map



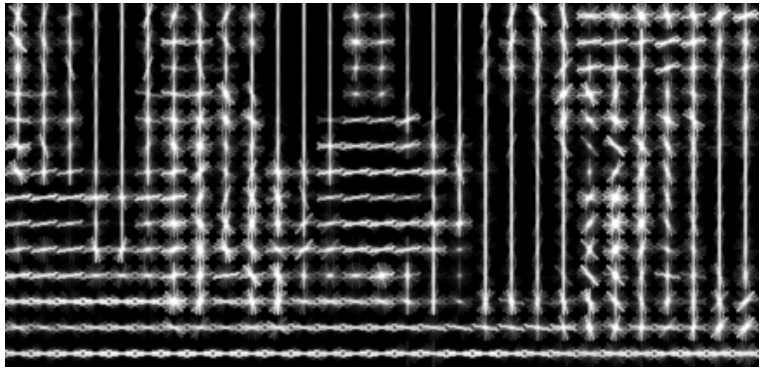
Template



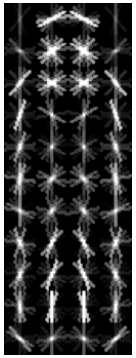
Pedestrian detection with HOG

- Train a pedestrian template using a linear SVM
- At test time, convolve feature map with template
- Find local maxima of response
- For multi-scale detection, repeat over multiple levels of a HOG pyramid

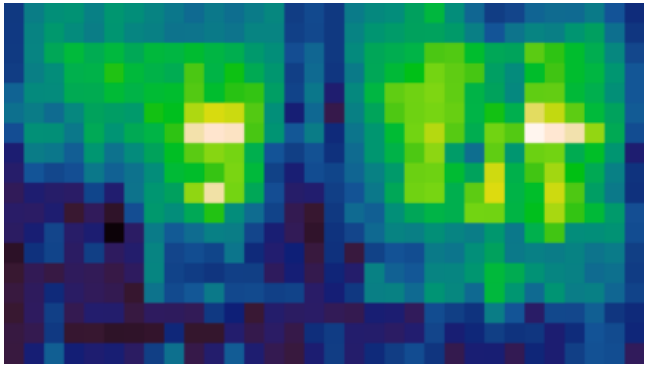
HOG feature map



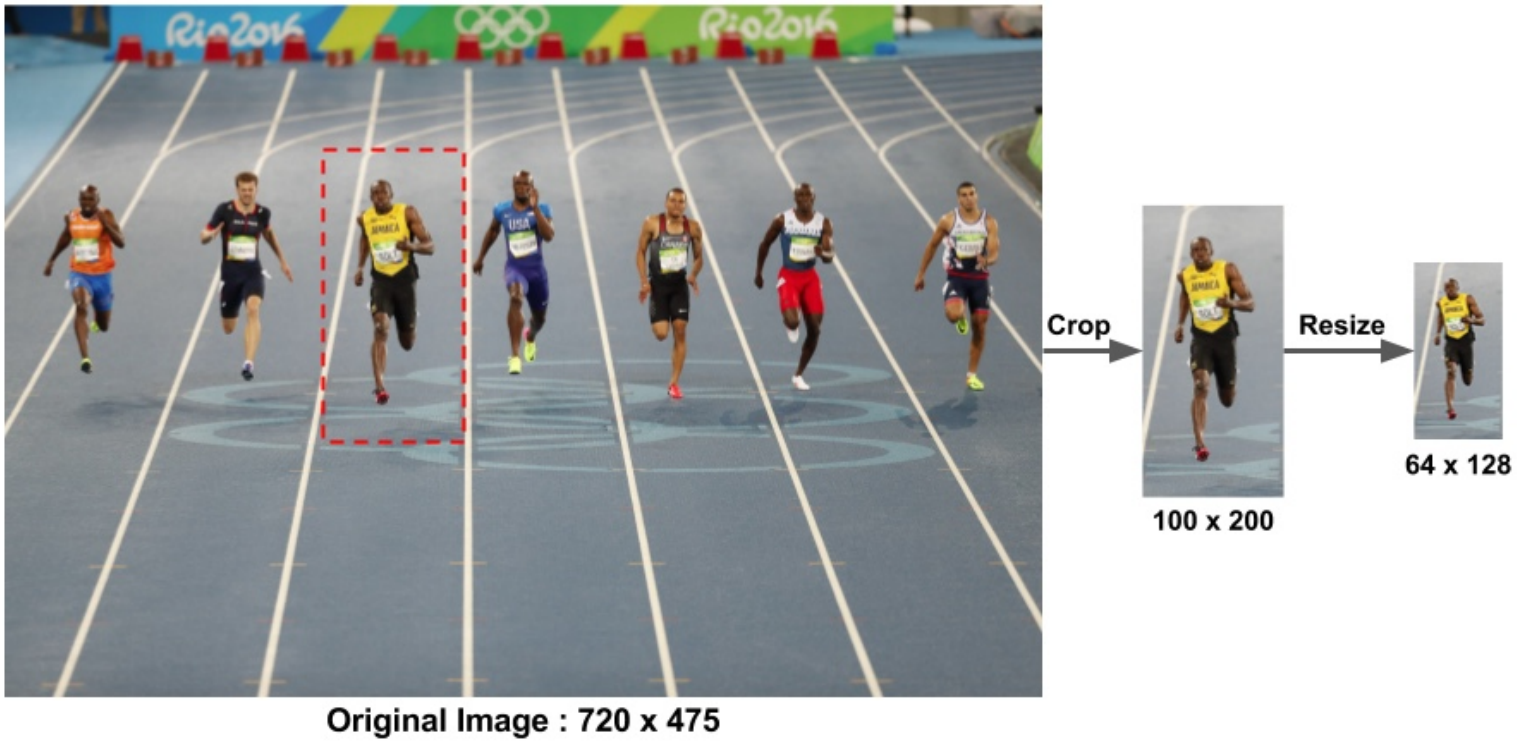
Template



Detector response map



Step 1: collect and rescale data



Suggestions on data collection:

- Diversity
- Normalize as same scale
- As many as possible?

Step 2: gradient computing

-1	0	1

Horizontal

-1
0
1

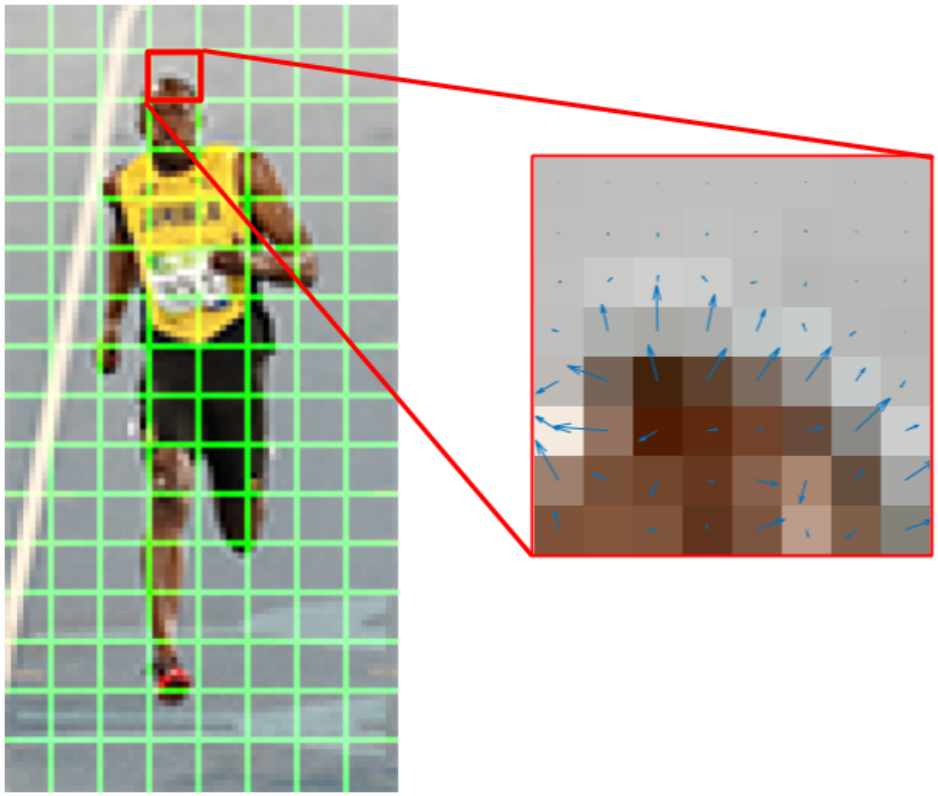
Vertical

Horizontal

Vertical

Magnitude

Step 3: calculate HOG in 8×8 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

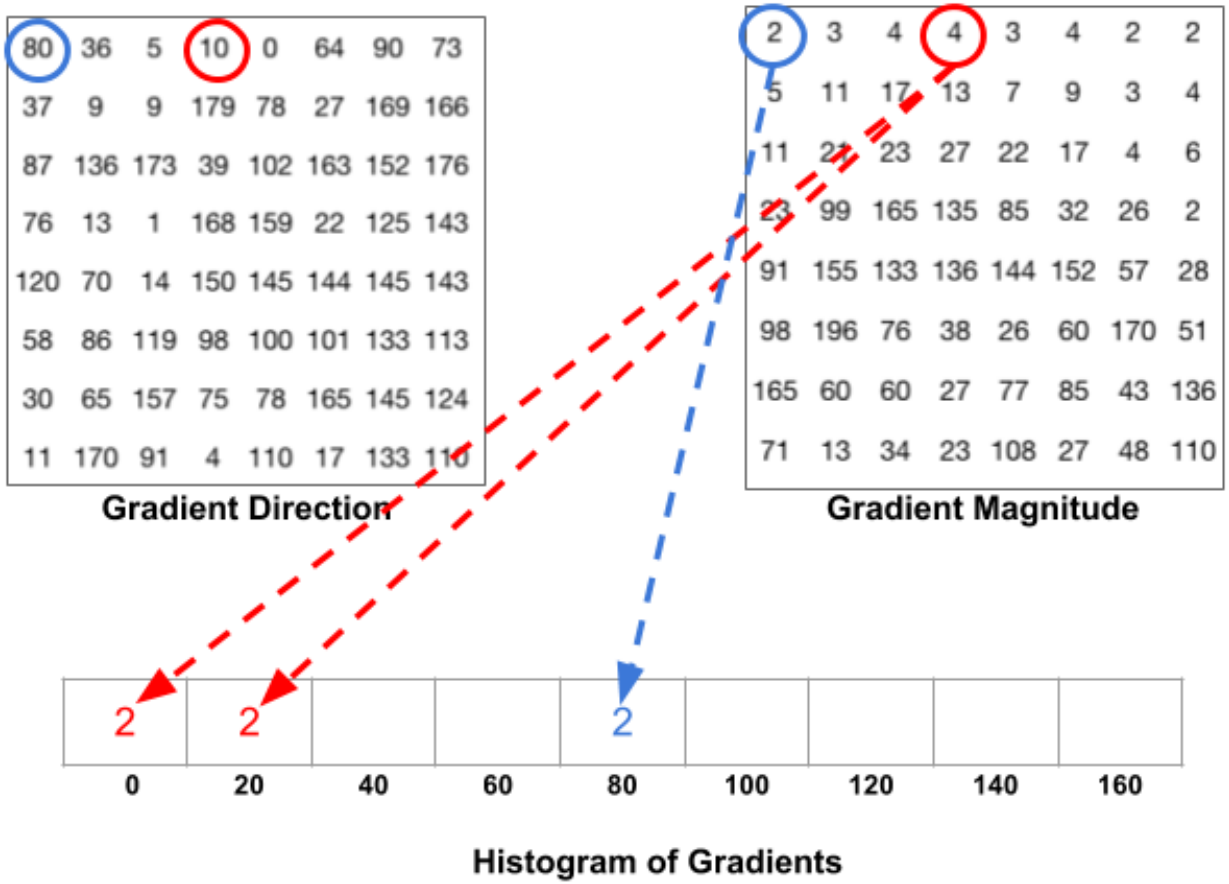
Gradient Direction

In each pixel we have horizontal gradient dx and vertical gradient dy

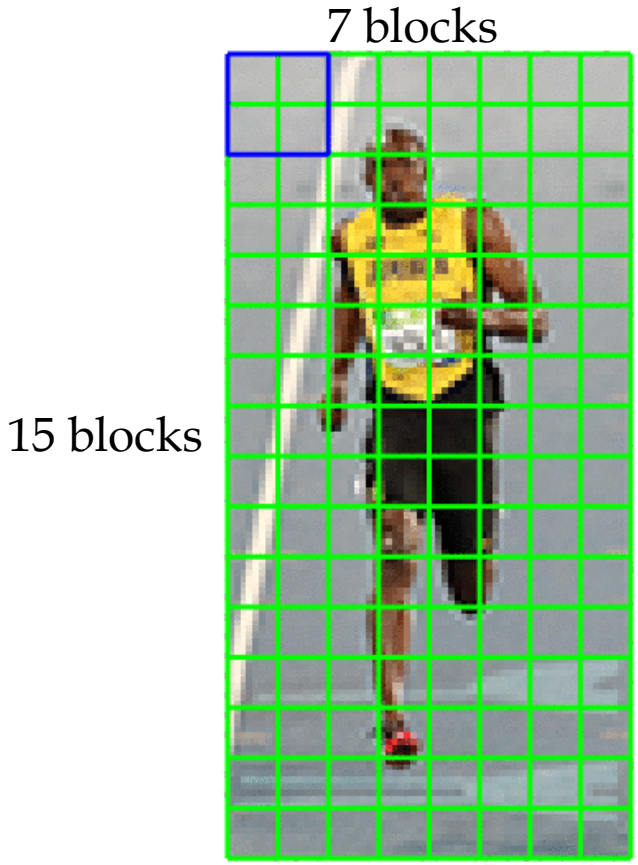
$$\text{magnitude} = \sqrt{dx^2 + dy^2}$$

$$\text{direction} = \arctan(\frac{dy}{dx})$$

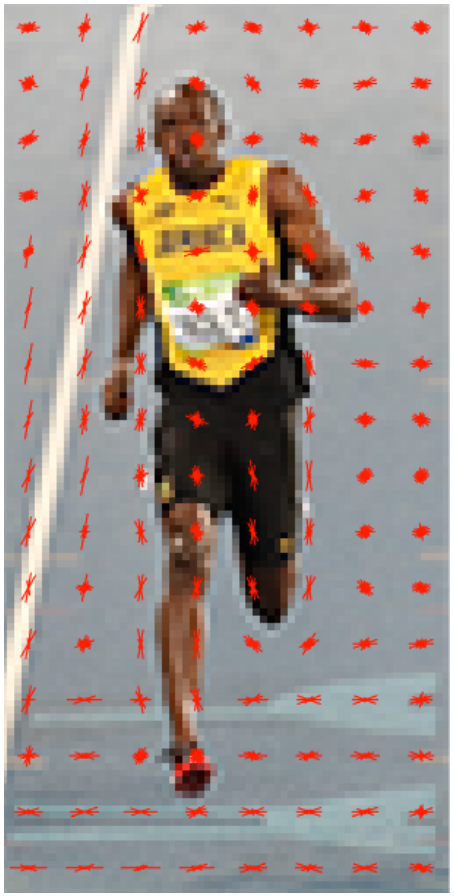
Step 4: weighted vote into spatial and orientation cells



Step 5: 16×16 block normalization

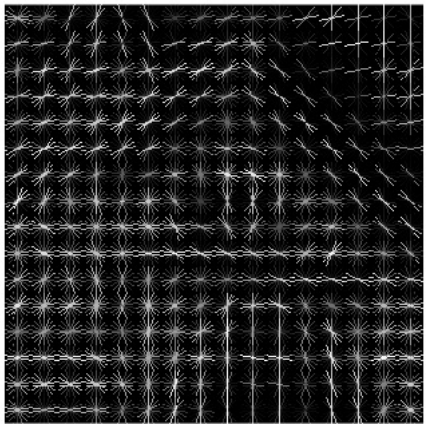


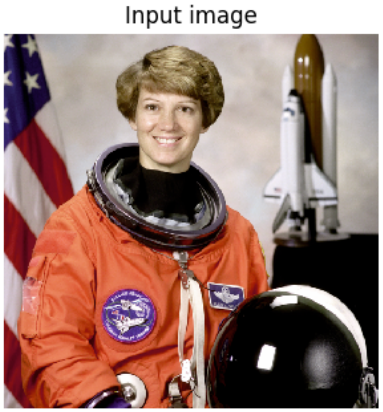
HOG descriptor



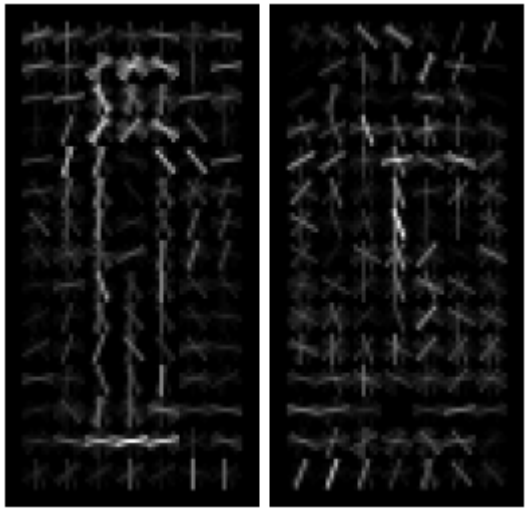
$= 15 \times 7 \times 36 = 3780$ # feature

HOG visualization

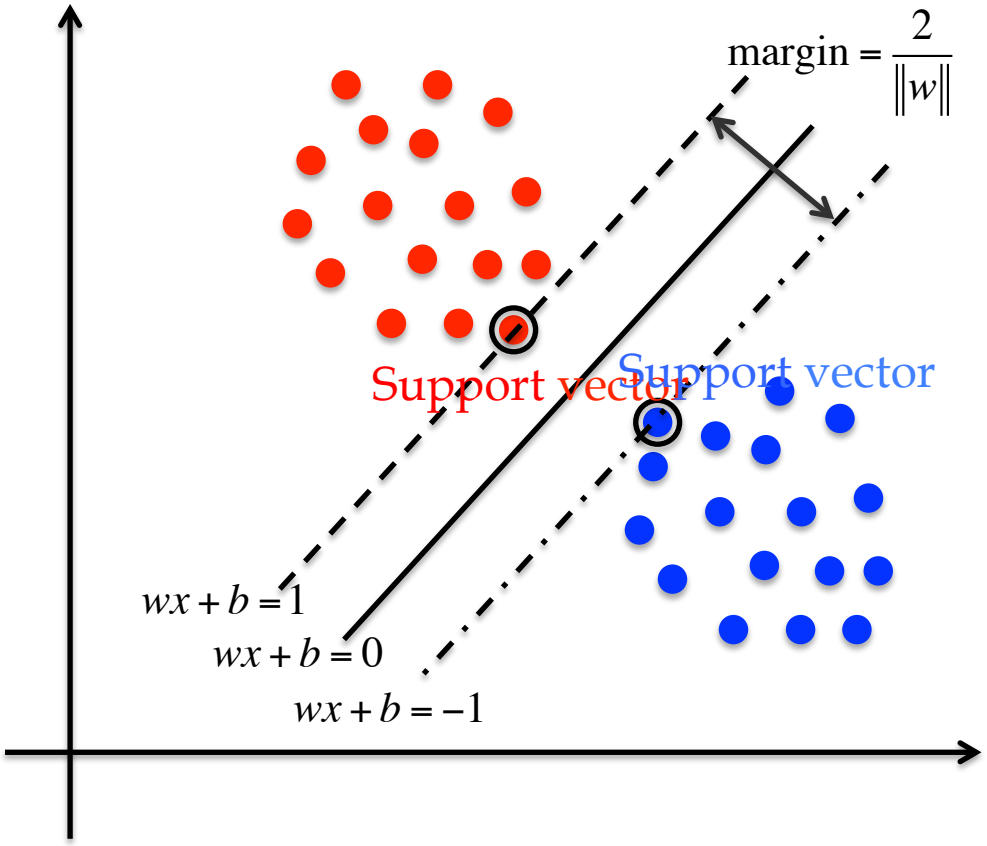




Training a linear SVM using HOG

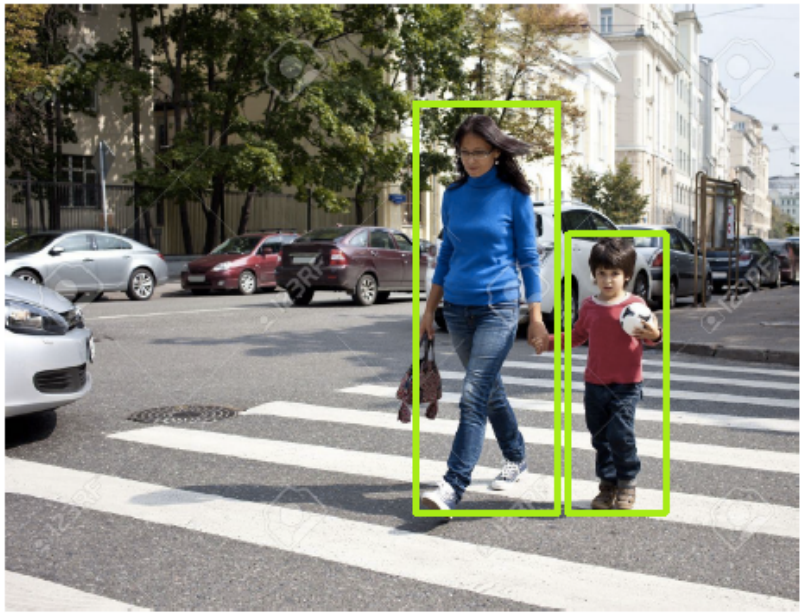


pos w neg w



Detect a pedestrian in an image

- Suppose we have trained a pedestrian detection classifier, how to detect pedestrian in an image?
- Remain challenge: different scale!
- Any idea?



Pyramid

Level 1

Level 2

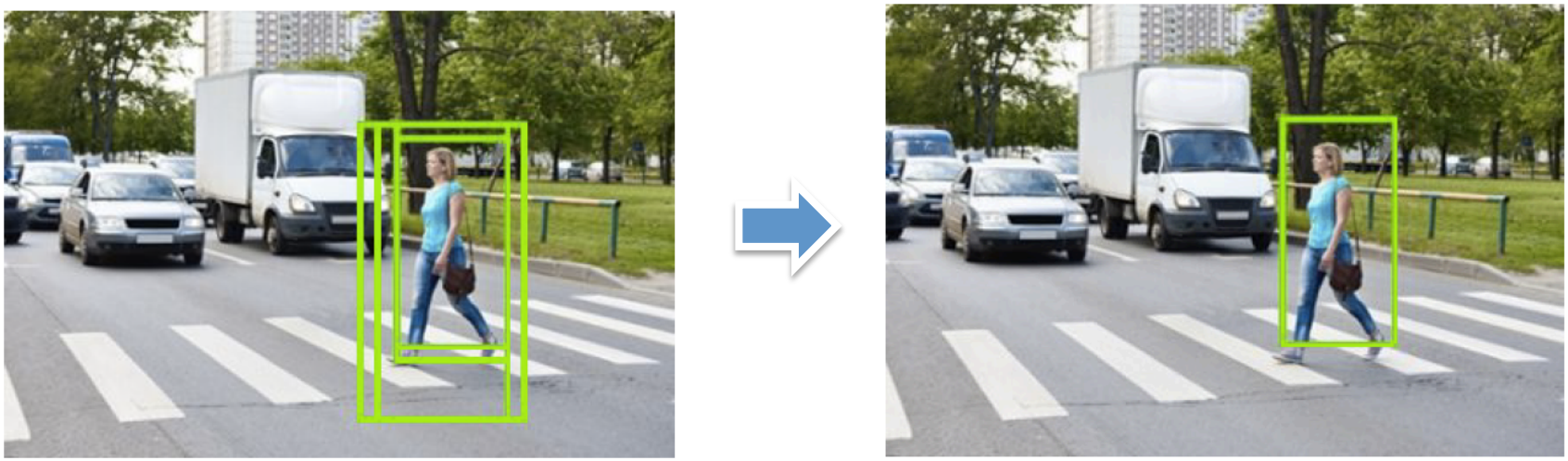
Level 3

Q: pyramid and sliding windows

- How many levels of pyramid?
- Overlap between sliding windows?

Tradeoff between computation and performance

Non-maximum suppression

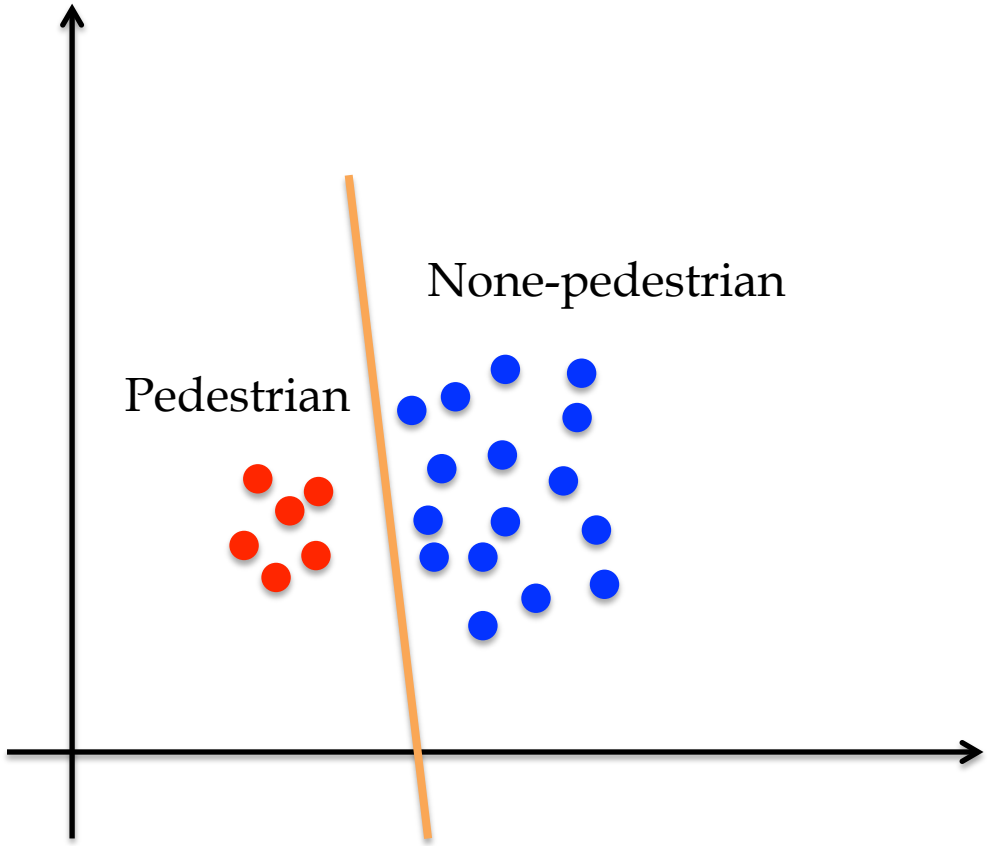


Same detection across different pyramid

Non-maximum suppression

- Step 1: Sort bounding boxes in descending order according to predicted scores.
- Step 2: Greedily select the highest scoring boxes while skipping detections with bounding boxes that at least 50% covered by a bounding box of a previously select detection.

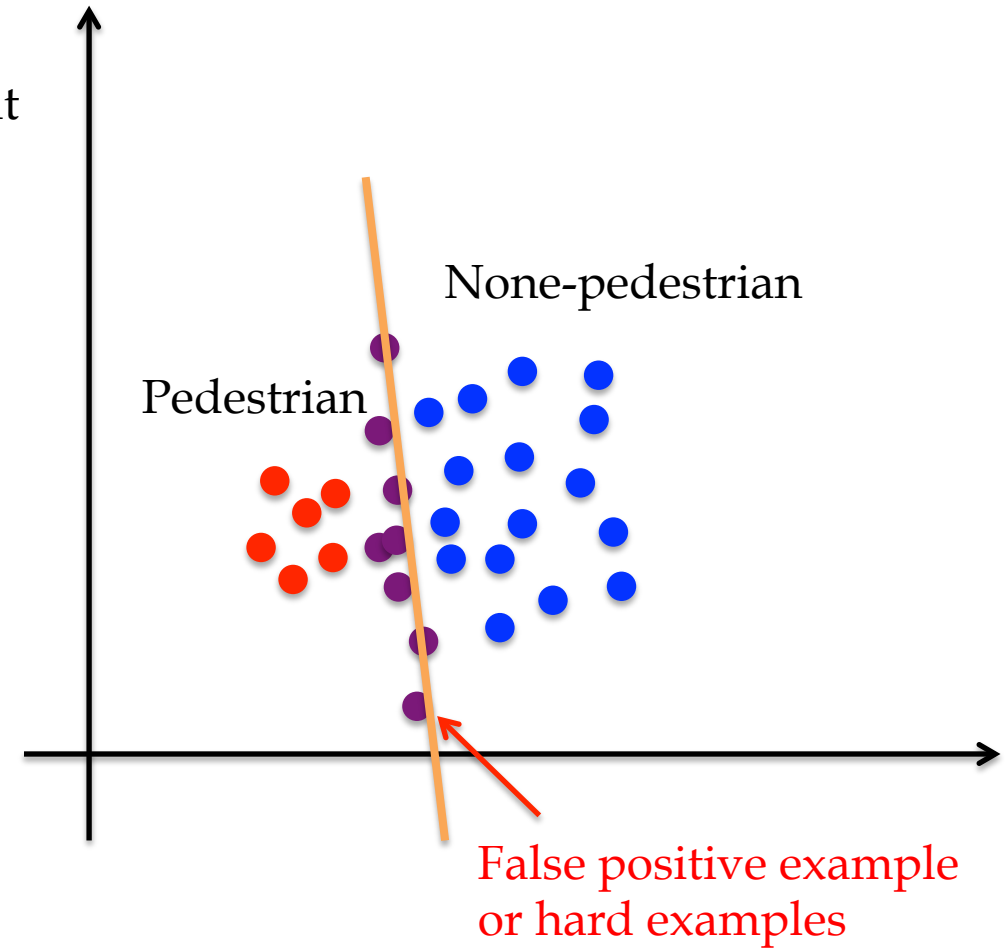
A more robust classifier



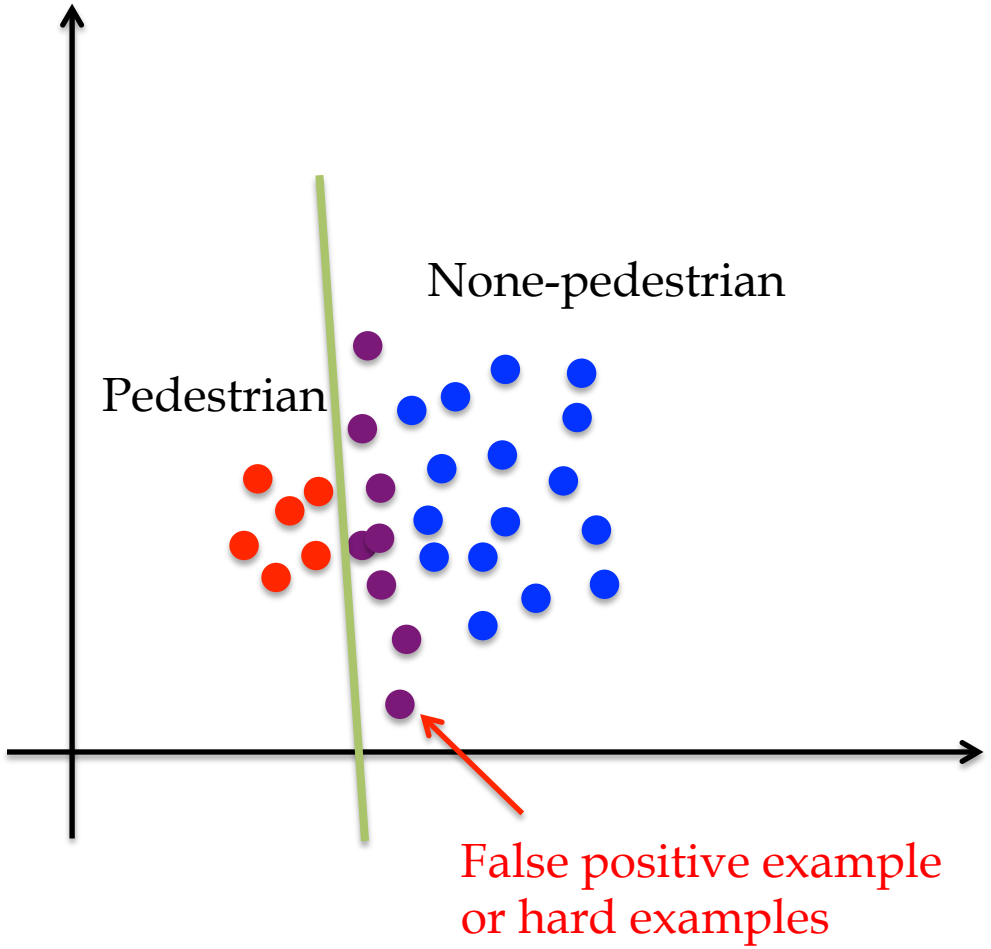
The classifier may not that accurate

A more robust classifier

Test on non-pedestrian images and select those that are predicted as pedestrians.



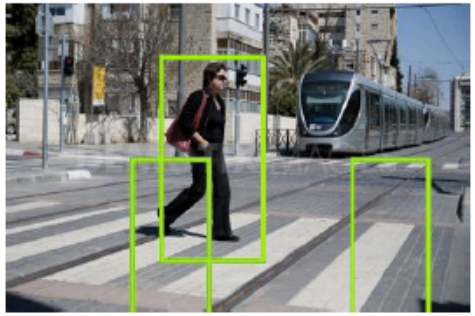
A more robust classifier

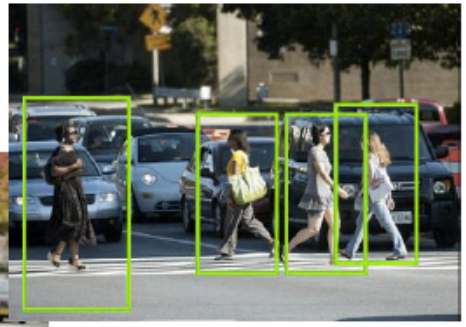
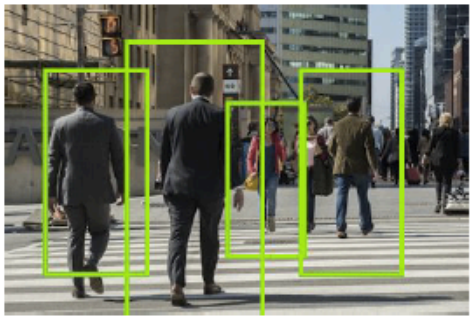
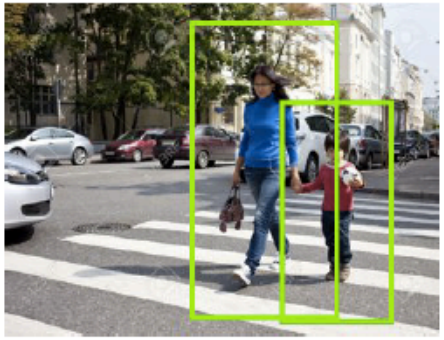
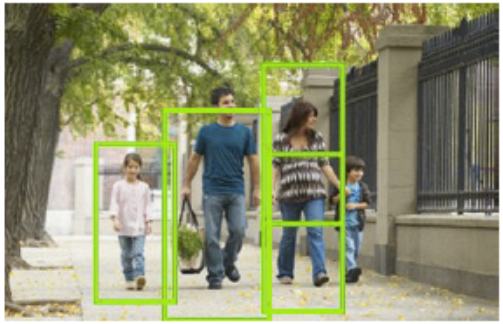
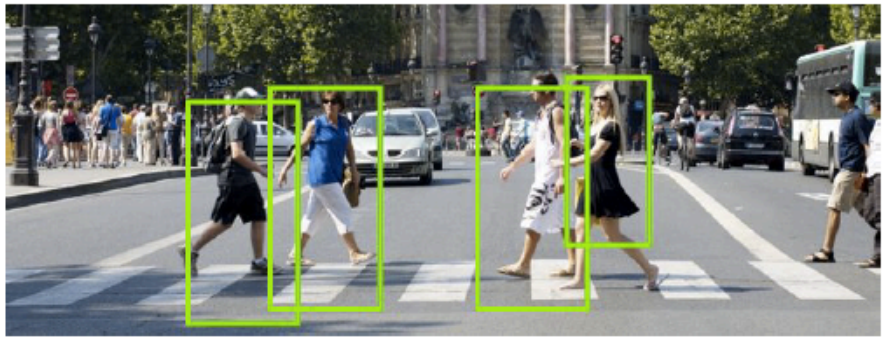
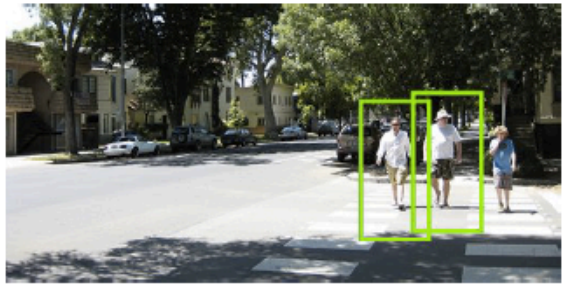


Dataset

- Available dataset from HOG paper (http://pascal.inrialpes.fr/data/human/)
- There are more than 2,000 positive examples, and more than 400 pedestrian-free images.
- Each example is cropped to 64×128 patches.

Experimental results



Overview: Dalal-Triggs pedestrian detector

- Training stage:
  - Given positive (pedestrian) and negative (non-pedestrian) examples (64×128), compute HOG features and train a linear SVM classifier
- Test stage:
  - Given a test image, extract fixed-sized (64×128 pixel) window at each position and scale
  - Compute HOG (histogram of gradient) features within each window
  - Score the window with the trained linear SVM classifier
  - Perform non-maximum suppression to remove overlapping detections with lower scores

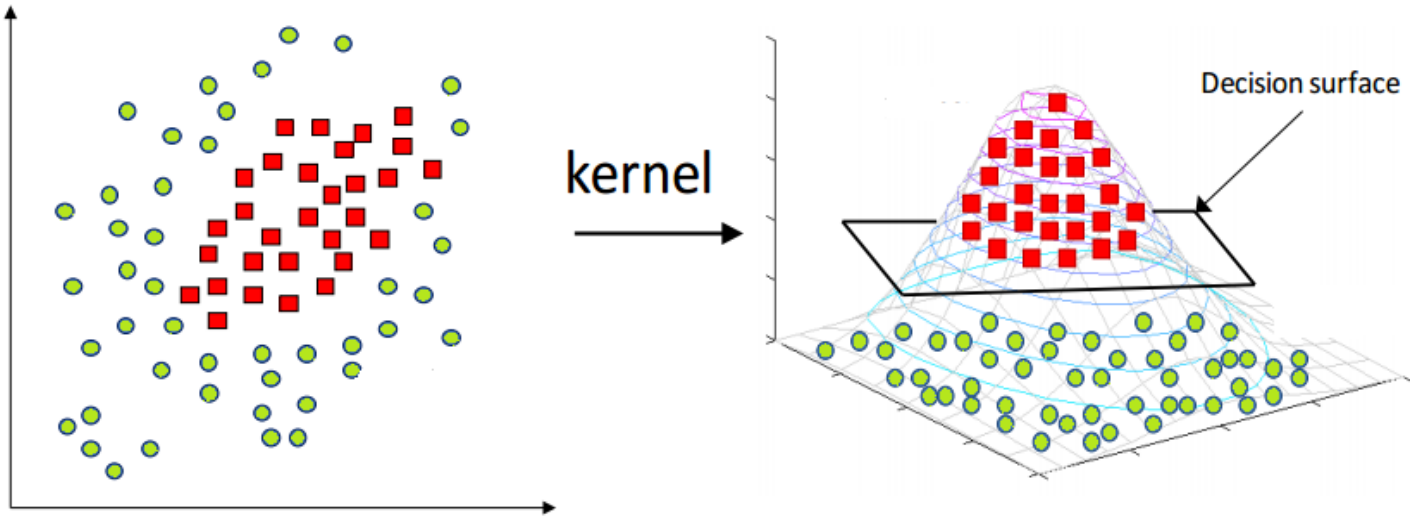
Neural networks – Part 0
Introduction

Non-linear data representation

- Apply the linear model not to data x itself but to a transformed input $\Phi(x)$, where Φ is a non-linear transformation. In other words, mapping the linear data to a non-linear feature space
- Three options to choose the mapping Φ
  - use a very generic Φ
  - manually engineer Φ
  - deep learning strategy

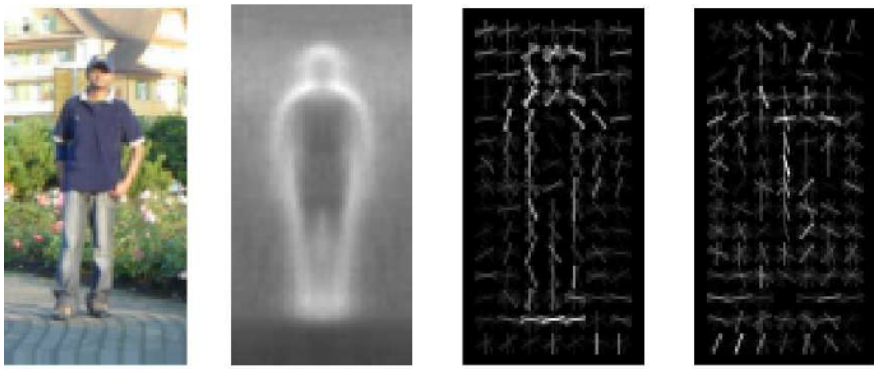
Option 1: use a very generic Φ

- Mapping data from *n*-dimensional to *N*-dimensional space (*n* << *N*)
- Pros: can always learn a linear classifier
- Cons: poor generalization to the test set
- Example: kernel trick in SVM



Option 2: manually engineer Φ

- Requires human effort for each separate task with practitioners specializing in different domains, such as computer vision or speech recognition
- Pros: good performance
- Cons: time-consuming to design and difficult to transfer between domains
- Example: Histograms of oriented gradients (HOG)



Option 3: deep learning strategy

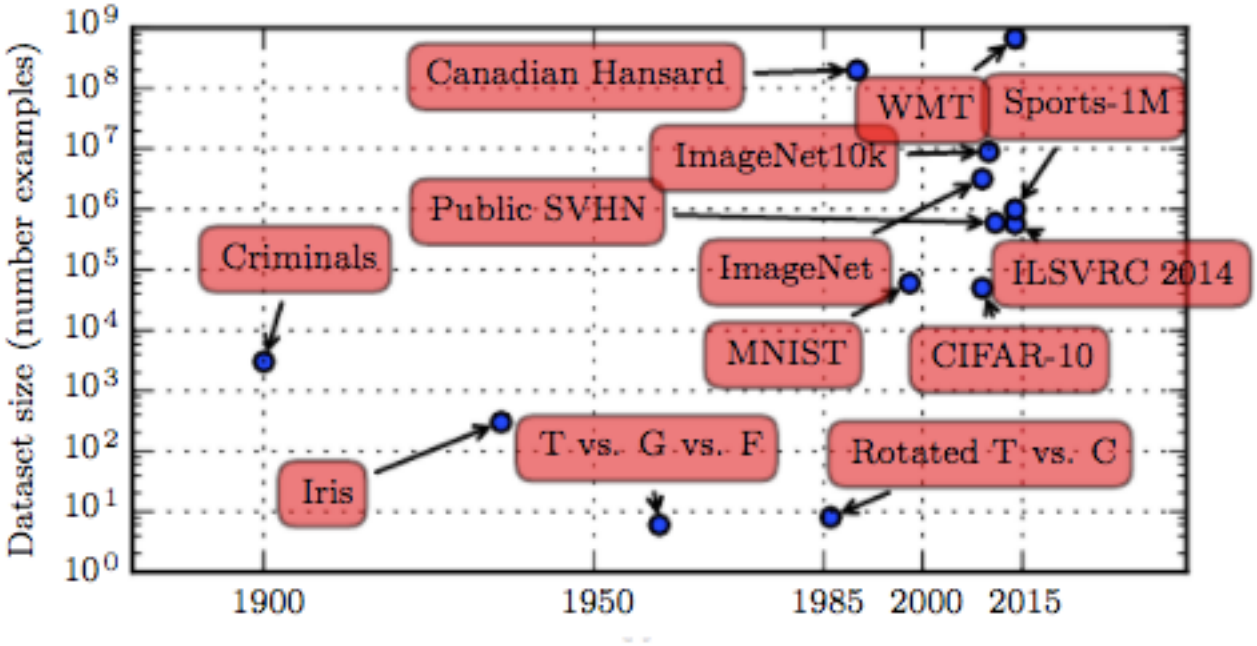
Given a model $y = f(x;\theta,w,b) = \Phi(x;\theta)w + b$, deep learning learns both θ, w, and b simultaneously

- Pros:
  - Outstanding performance
  - Easy to transfer between domains
- Cons:
  - Time-consuming
  - Non-convex optimization

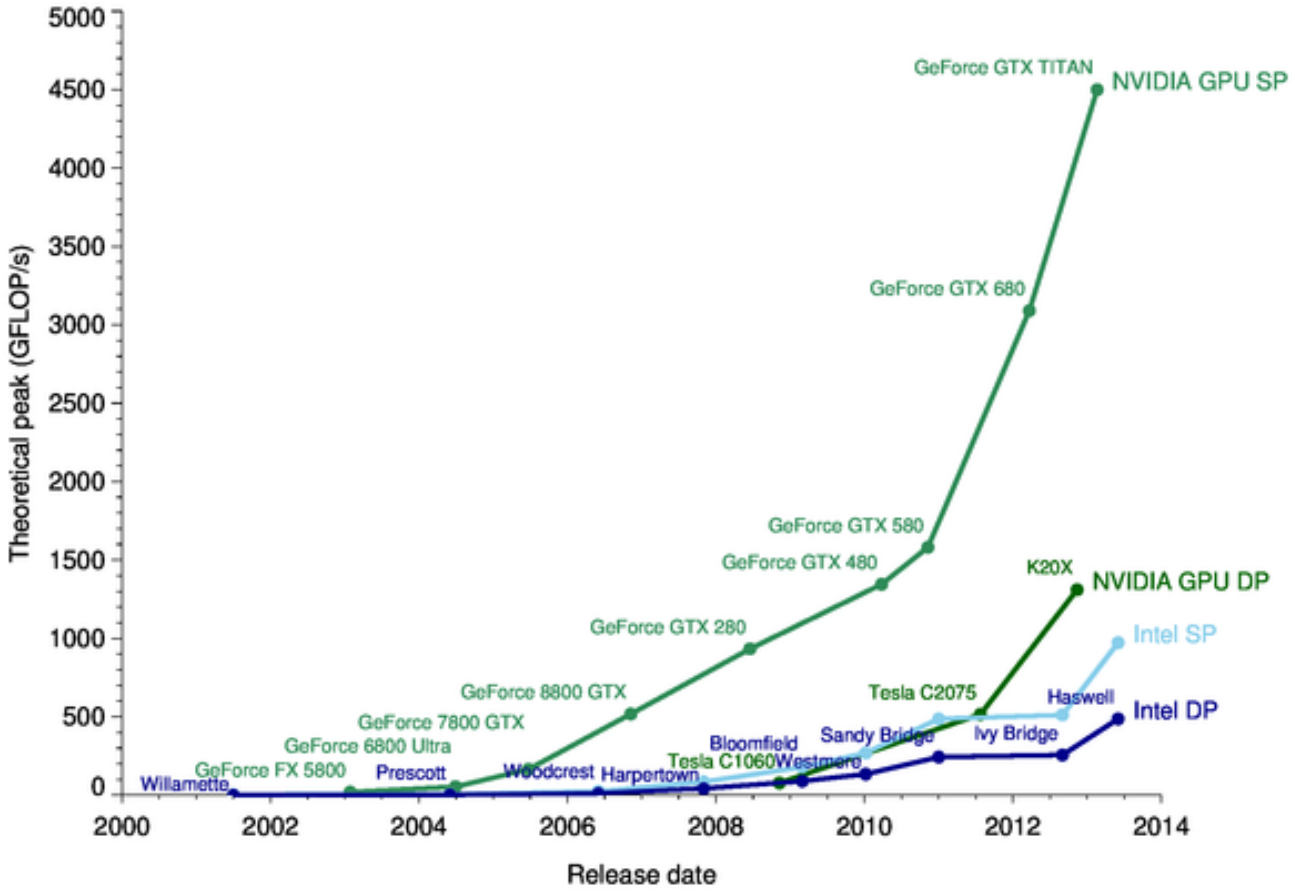
Why is deep learning taking off?

- Neural network has a very long history in machine learning, it has attracted more attentions recently due to:
  - Increasing dataset size
  - Increasing computational power
  - Increasing performance

Increasing dataset size



Increasing computational power



Increasing performance