# CSE473: Computational Intelligence

# Neural Architectures for Multiclass Models

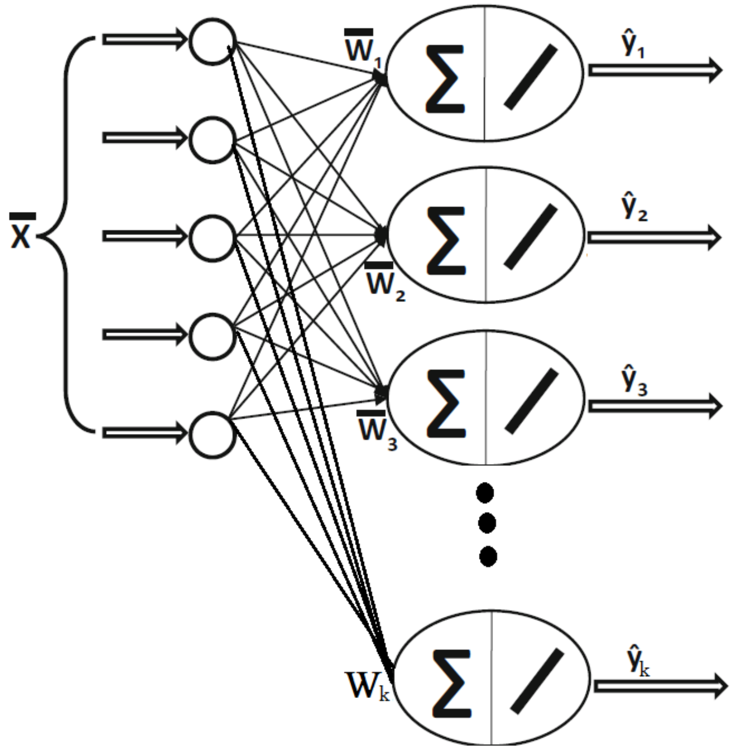
by:
Hossam Abd El Munim
Computer & Systems Engineering Dept.,
Ain Shams University,
1 El-Sarayat Street, Abbassia, Cairo 11517

## Outline

- Shallow neural network for multi-classification
- Problem definition.
- Class score
- Perceptron criterion loss function
- Multi-class SVM (MCSVM)
- Action recognition and MCSVM
- Hierarchical MCSVM
- Multinomial regression (Soft-max classification)
- Initialization of parameters
- Data normalization

# Multiclass Problem Definition

Given a data set, D = {(**X**$_1$, y$_1$), (**X**$_2$, y$_2$)... (**X**$_N$, y$_N$)} of labelled feature vectors where y ϵ {1,2,…,k}. Here, k represents the number of categories. We to estimate the vectors **W**$_1$ , **W**$_2$ , ..., **W**$_k$ that minimize a certain objective/loss function. **What is it?**



## Score and Classification Decision

For a feature vector input $\mathbf{X}_i$, it must have the maximum score of class $y_i$ for a correct classification and zero loss.

| Class# | 1 | 2 | … | $y_i$ | … | k |
|---|---|---|---|---|---|---|
| Score | $\mathbf{W}_1.\mathbf{X}_i$ | $\mathbf{W}_2.\mathbf{X}_i$ | … | $\mathbf{W}_{yi}.\mathbf{X}_i$ | … | $\mathbf{W}_k.\mathbf{X}_i$ |

# Multiclass Perceptron: Loss Function

Given a data set, D = {($\mathbf{X}_1$, $y_1$), ($\mathbf{X}_2$, $y_2$)... ($\mathbf{X}_N$, $y_N$)} of labelled feature vectors where y $\epsilon$ {1,2,…,k}. Here, represents the number of categories. We to estimate the vectors $\mathbf{W}_1$ , $\mathbf{W}_{2\,,\,\ldots,}$ $\mathbf{W}_k$ that minimize a certain objective/loss function. What is it?

$$L = \sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{L_i}$$

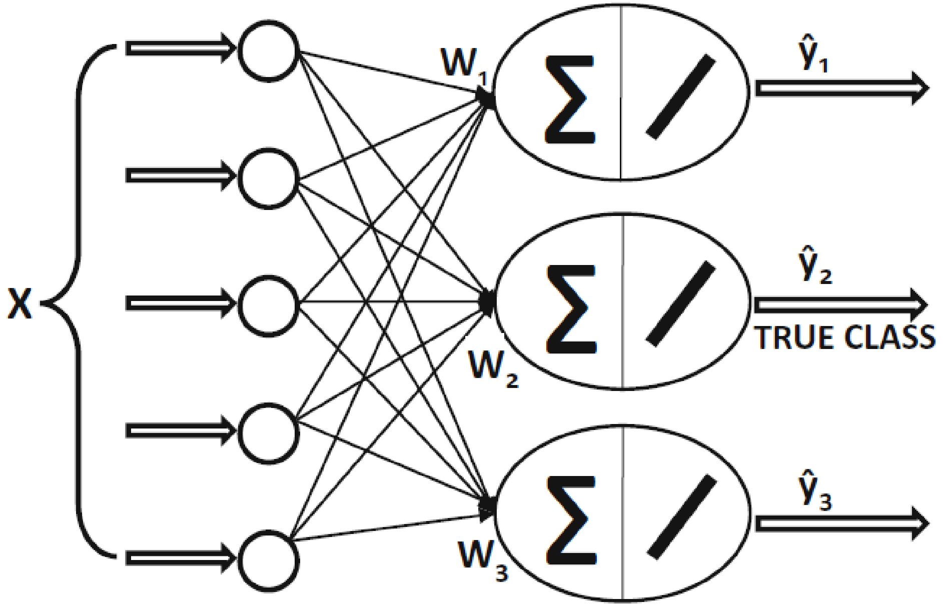
$$\mathrm{d_r} = \max(\mathbf{W}_{\mathrm{r}}.\mathbf{X}_{\mathrm{i}} - \mathbf{W}_{\mathrm{y_i}}.\mathbf{X}_{\mathrm{i}}\,, 0)$$

$$\mathrm{r} \in \{\{1,2,\ldots,\mathrm{k}\} - \{\mathrm{y_i}\}\}$$

$$\mathrm{L_i} = \max(\ldots, \mathrm{d_r}, \ldots)$$

# Multiclass Perceptron: Loss Function Example

Assume k = 3, $y_i$ = 2, then $L_i$ will be:-



$$d_1 = \max(\mathbf{W}_1.\mathbf{X}_i - \mathbf{W}_2.\mathbf{X}_i, 0)$$

$$d_3 = \max(\mathbf{W}_3.\mathbf{X}_i - \mathbf{W}_2.\mathbf{X}_i, 0)$$

$$r \in \{1,3\}$$

$$L_i = \max(d_1, d_3)$$

# Multiclass Perceptron: Learning Rule

- For correctly classified features, there will be no update as the loss is zero. It is derivative will be zero as well.
- For miss-classified features, there will be a derivative computed as follows:-

$$\mathrm{d}_1 = \max(\mathbf{W}_1 . \mathbf{X}_\mathrm{i} - \mathbf{W}_2 . \mathbf{X}_\mathrm{i}, 0)$$

$$\mathrm{d}_3 = \max(\mathbf{W}_3 . \mathbf{X}_\mathrm{i} - \mathbf{W}_2 . \mathbf{X}_\mathrm{i}, 0)$$

$$\mathrm{r} \in \{1,3\}$$

$$\mathrm{L}_\mathrm{i} = \max(\mathrm{d}_1, \mathrm{d}_3)$$

$$\frac{\partial}{\partial \mathbf{W}_\mathrm{r}} \mathrm{L}_\mathrm{i} = \begin{cases} -\mathbf{X}_\mathrm{i} \text{ if } \mathrm{r} = \mathrm{y}_\mathrm{i} \\ \mathbf{X}_\mathrm{i} \text{ if } \mathrm{r} \neq \mathrm{y}_\mathrm{i} \\ 0 \text{ otherwise} \end{cases}$$ is the most miss-classified prediction.

## Multiclass Perceptron: Learning Rule Disadvantages

- The multiclass perceptron only updates the linear separator of a class that is predicted most incorrectly along with the linear separator of the true class.

## Weston-Watkins SVM

- On the other hand, the Weston-Watkins SVM updates the separator of any class that is predicted more favourably than the true class. In both cases, the separator of the observed class is updated by the same aggregate amount as the incorrect classes (but in the opposite direction).
- Not only does the Weston-Watkins SVM update the separator in the case of misclassification, it updates the separators in cases where an incorrect class gets a prediction that is “uncomfortably close” to the true class. This is based on the notion of margin.

## Weston-Watkins SVM Loss Function

$$L = \sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{L_i}$$

$$\mathrm{d_r} = \max(1 + \mathbf{W}_\mathrm{r}.\mathbf{X}_\mathrm{i} - \mathbf{W}_{\mathrm{y_i}}.\mathbf{X}_\mathrm{i}, 0)$$

$$\mathrm{r} \in \{\{1,2,\dots,\mathrm{k}\} - \{\mathrm{y_i}\}\}$$

$$\mathrm{L_i} = \sum_{\mathrm{r}=1,\, \mathrm{r}\neq \mathrm{y_i}}^{\mathrm{k}} \mathrm{d_r}$$

## Weston-Watkins SVM Learning Rule

$$\frac{\partial}{\partial \mathbf{W}_{\mathrm{r}}} \mathrm{L}_{\mathrm{i}} = \begin{cases} -\mathbf{X}_{\mathrm{i}} \left( \sum_{j=1, j \neq r}^{k} \delta_{ji} \right) & \text{if } \mathrm{r} = \mathrm{y}_{\mathrm{i}} \\ \mathbf{X}_{\mathrm{i}} (\delta_{ri}) & \text{if } \mathrm{r} \neq \mathrm{y}_{\mathrm{i}} \end{cases}$$

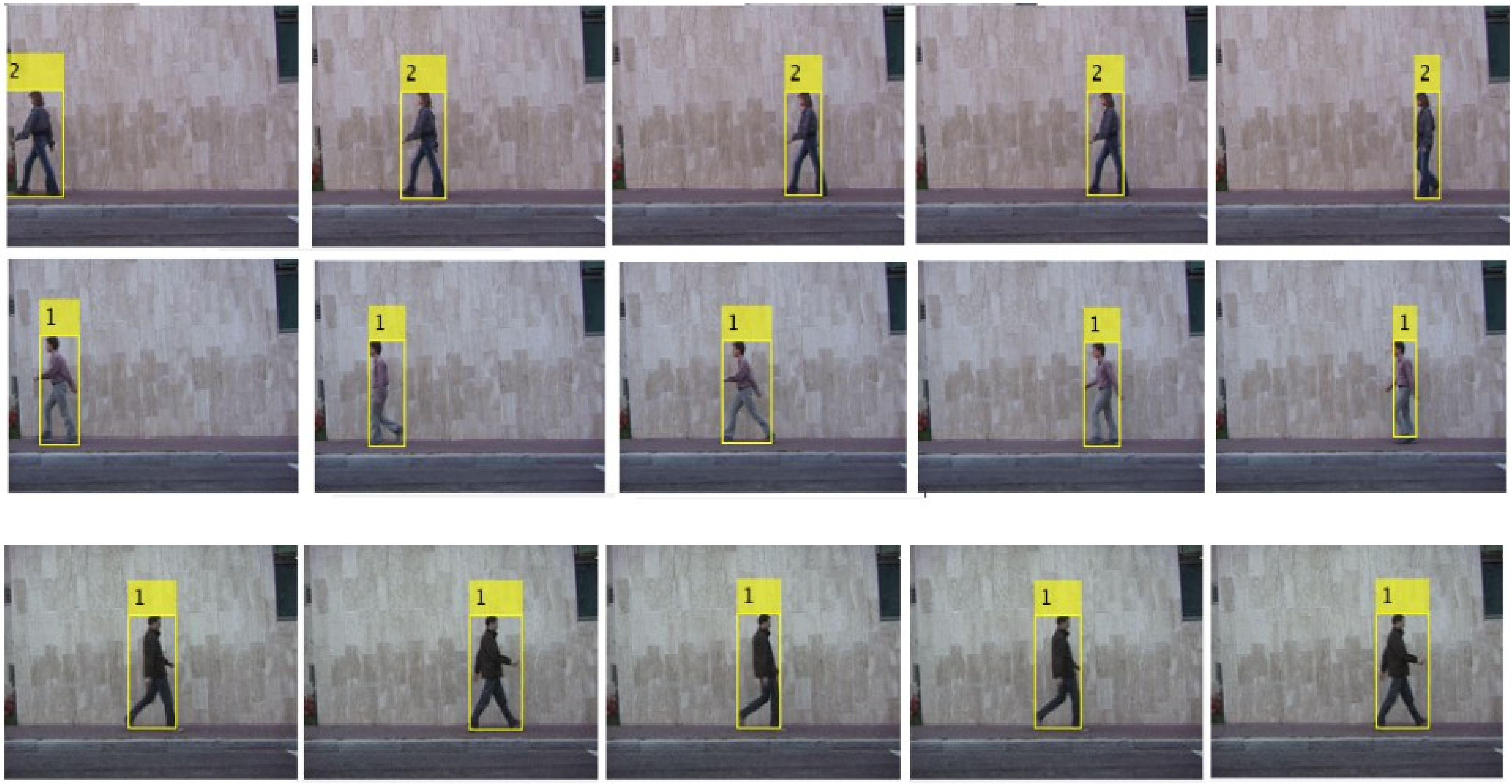
$$\delta_{ji} = \begin{cases} 1 \text{ if } 1 + \boldsymbol{W}_{j} . \boldsymbol{X}_{i} - \boldsymbol{W}_{y_i} . \boldsymbol{X}_{i} > 0 \\ 0 & \text{Otherwise} \end{cases}$$

# Weston-Watkins SVM Learning Rule with Regularization

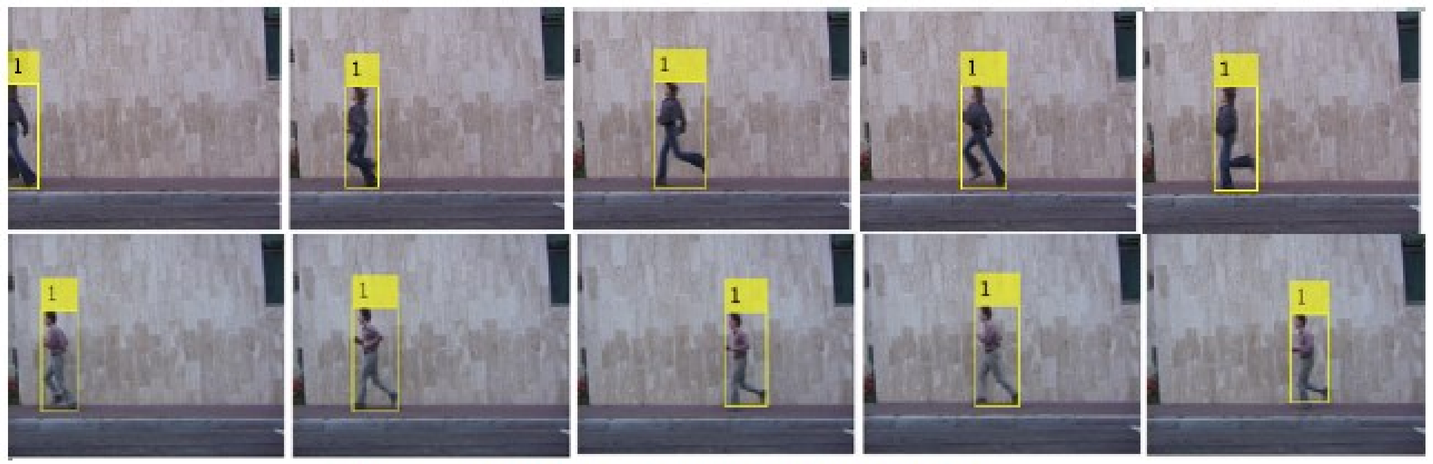
$$\mathrm{L} = \sum_{\mathrm{i}=1}^{\mathrm{N}} \sum_{\mathrm{r}=1,\, \mathrm{r} \neq \mathrm{y_i}}^{\mathrm{k}} \mathrm{d_r} + (\lambda/2) \sum_{\mathrm{r}=1}^{\mathrm{k}} \mathbf{W}_{\mathrm{r}} . \mathbf{W}_{\mathrm{r}}$$

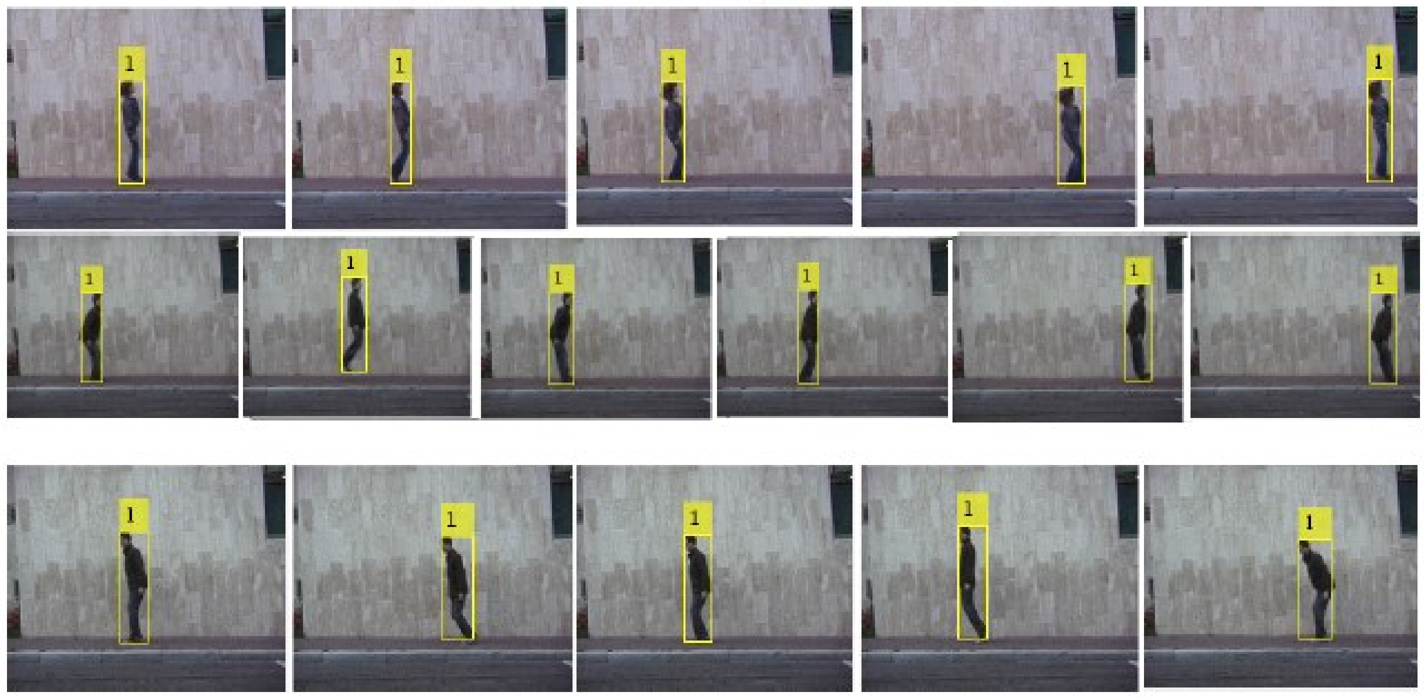
Regularization is considered
essential to the
proper functioning of a support
vector machine.

# Action Recognition (I)



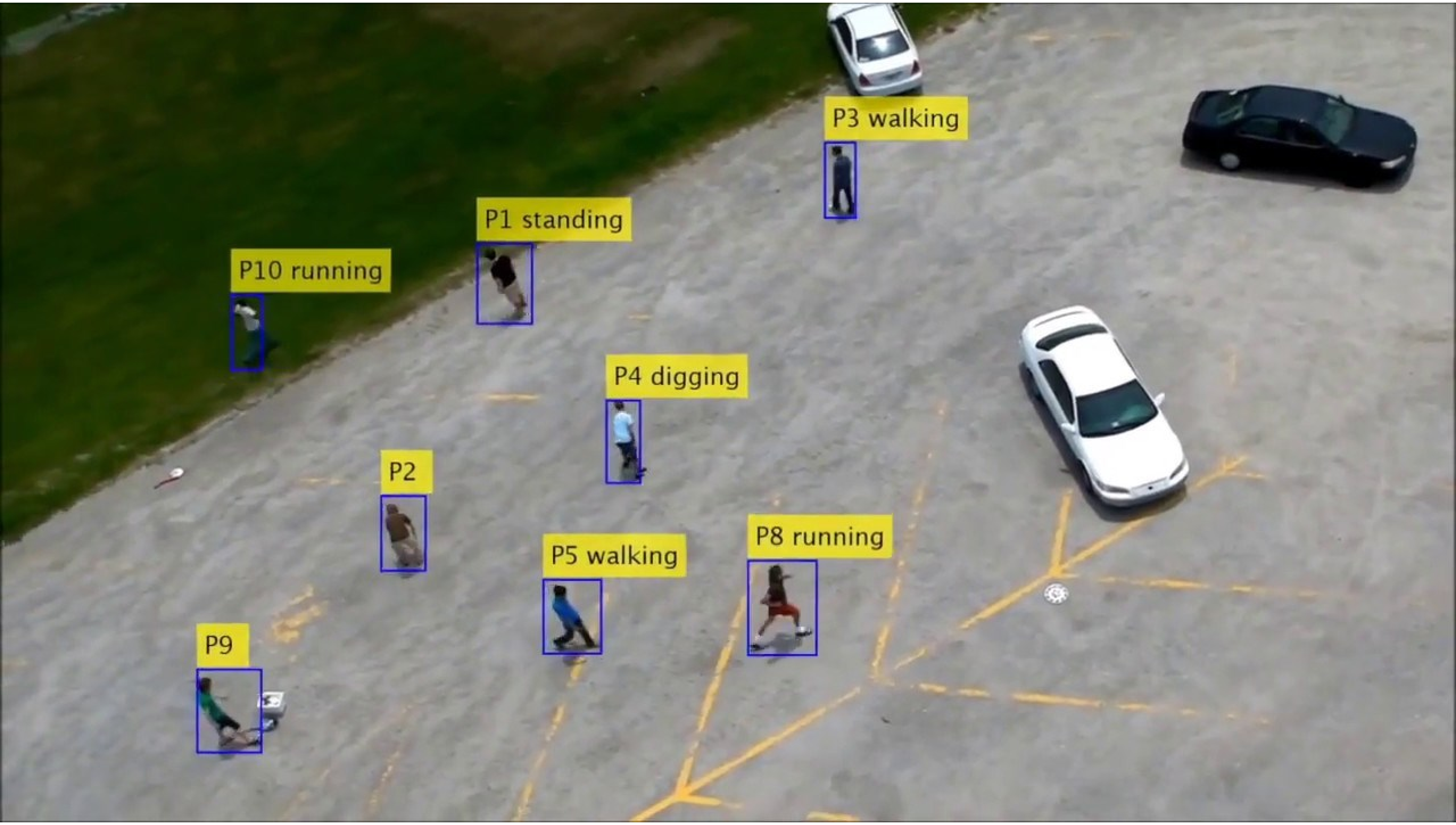
# Action Recognition (II)


Samples of Running activity for tracked persons

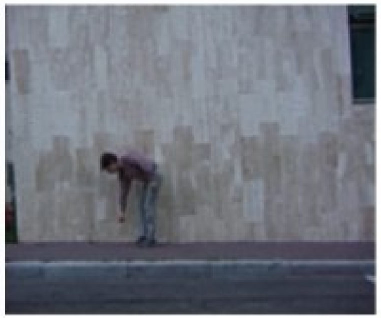


Samples of Jumping activity for tracked persons

# Action Recognition (III)



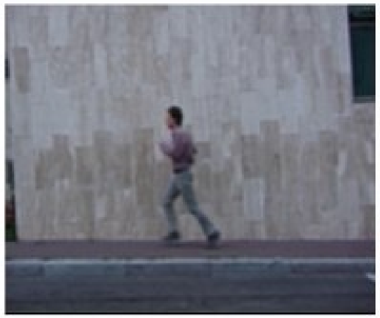
# Action Recognition: Hierarchical MCSVM


**(a)** Bend

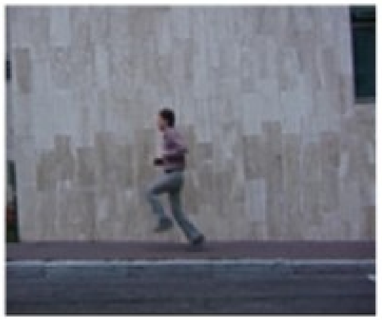
**(b)** Jack

**(c)** Jump

**(d)** Pjump


**(e)** Run

**(f)** Side

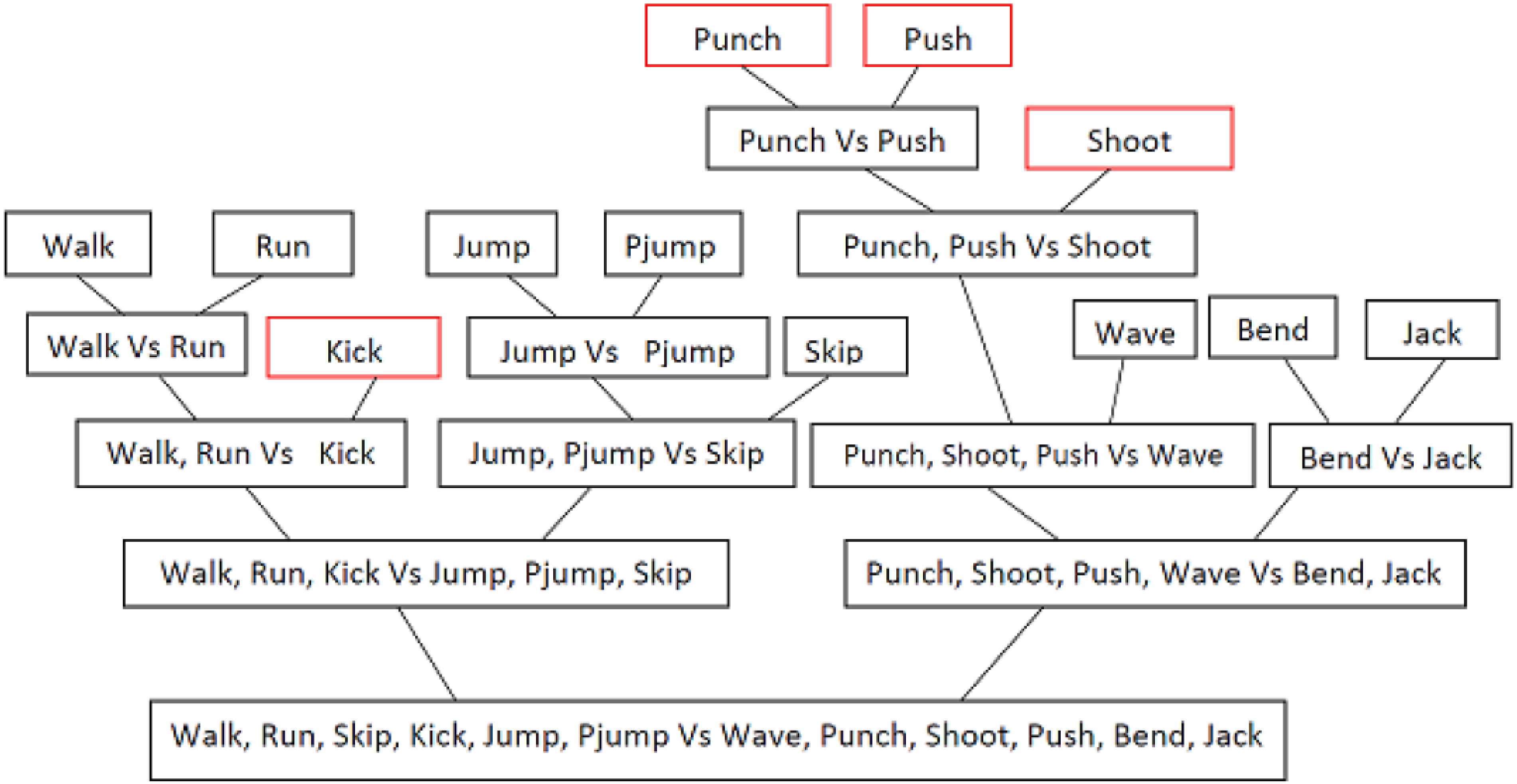

**(g)** Skip

**(h)** Walk

**(i)** Wave1

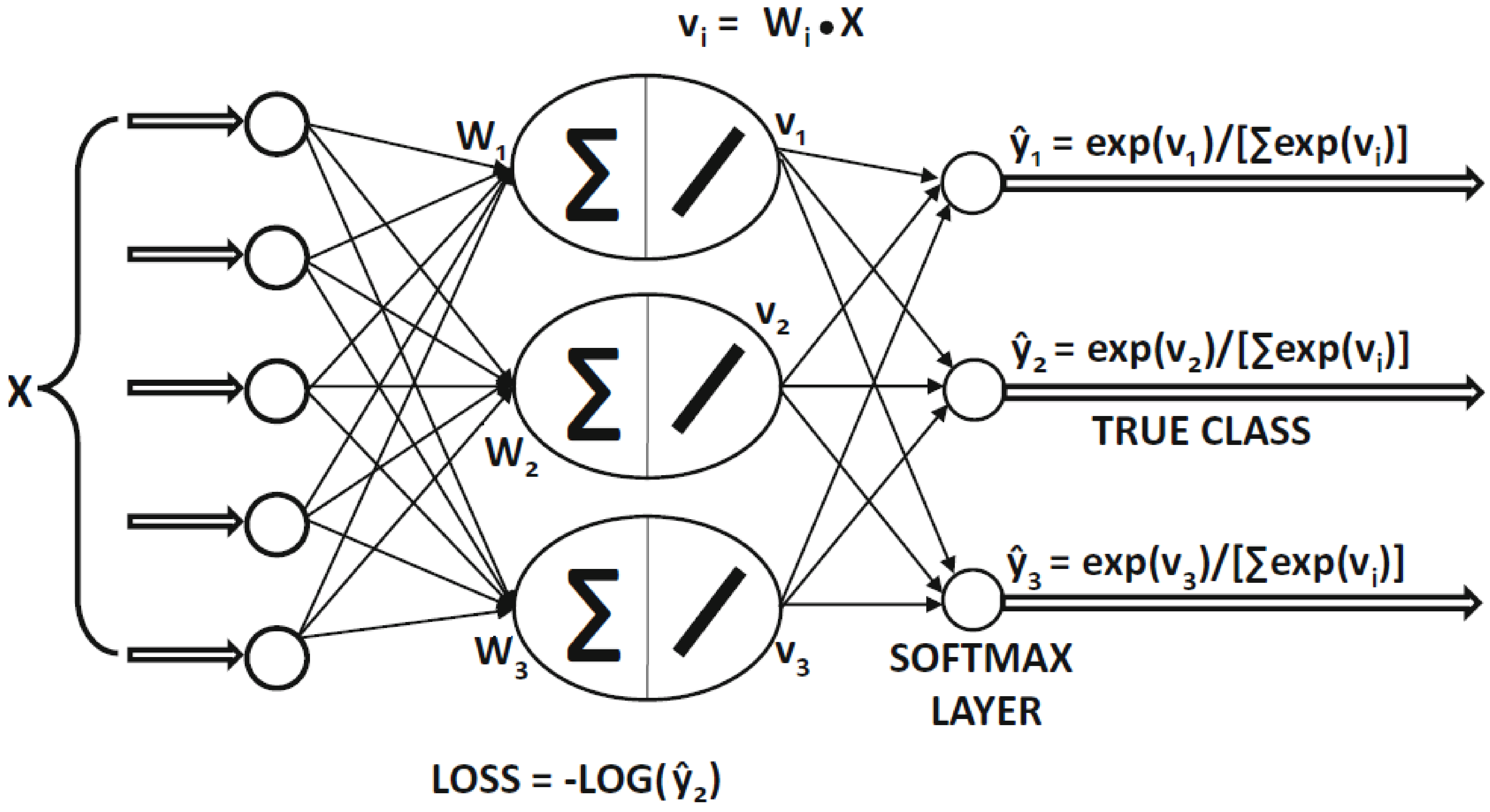
**(j)** Wave2

# Action Recognition: Hierarchical MCSVM


Punch
Push
Punch Vs Push
Shoot
Walk
Run
Jump
Pjump
Punch, Push Vs Shoot
Walk Vs Run
Kick
Jump Vs Pjump
Skip
Wave
Bend
Jack
Walk, Run Vs Kick
Jump, Pjump Vs Skip
Punch, Shoot, Push Vs Wave
Bend Vs Jack
Walk, Run, Kick Vs Jump, Pjump, Skip
Punch, Shoot, Push, Wave Vs Bend, Jack
Walk, Run, Skip, Kick, Jump, Pjump Vs Wave, Punch, Shoot, Push, Bend, Jack

Inverted binary tree hierarchical multiclass SVM.

# Multinomial Logistic Regression (Softmax Classifier)



$v_i = W_i \bullet X$
X
$W_1$
$W_2$
$W_3$
Σ
$v_1$
$v_2$
$v_3$
$\hat{y}_1 = \exp(v_1)/[\sum\exp(v_i)]$
$\hat{y}_2 = \exp(v_2)/[\sum\exp(v_i)]$
TRUE CLASS
$\hat{y}_3 = \exp(v_3)/[\sum\exp(v_i)]$
SOFTMAX LAYER
LOSS = $-\mathrm{LOG}(\hat{y}_2)$

## Multinomial Logistic Regression (Loss Function)

$$v_r = \mathbf{W}_\mathrm{r} . \mathbf{X}_\mathrm{i}$$

$$\widehat{y_r} = \frac{exp(v_r)}{\sum_{j=1}^{k} exp(v_j)}$$

$$\mathrm{L_i} = -\log(\widehat{\mathrm{y}_{\mathrm{y_i}}})$$

$$\mathrm{L_i} = -v_{\mathrm{y_i}} + \log(\sum_{j=1}^{k} exp(v_j))$$

# Multinomial Logistic Regression (Learning Rule)

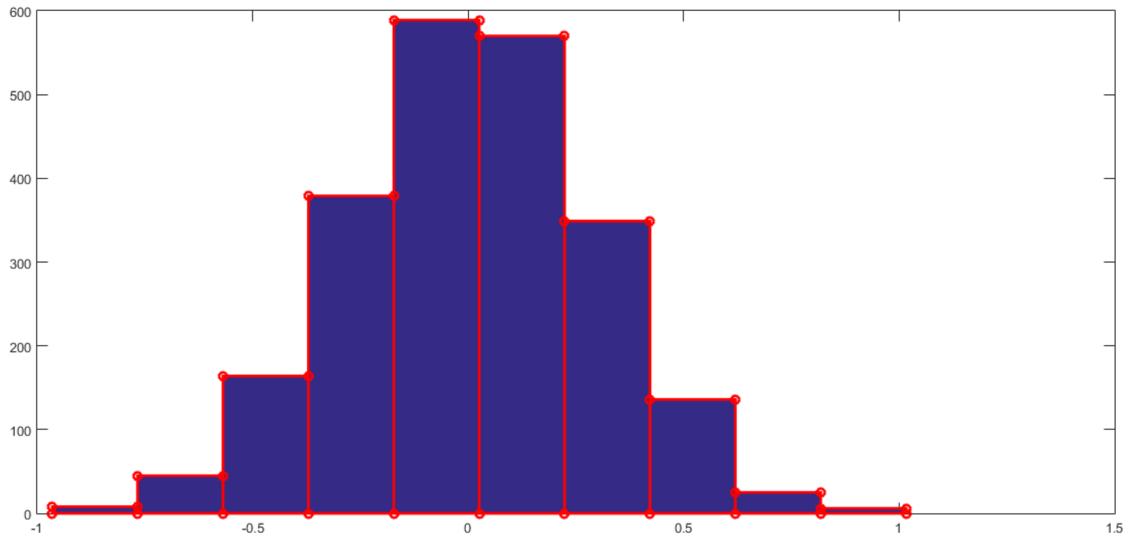
$$\frac{\partial \mathrm{L_i}}{\partial \mathbf{W}_r} = (\partial \mathrm{L_i} / \partial v_r)(\partial v_\mathrm{r} / \partial \mathbf{W}_r)$$

$$\partial \mathrm{L_i} / \partial v_r = \begin{cases} -(1 - \widehat{y_r}) \text{ if } \mathrm{y_i} = \mathrm{r} \\ \widehat{y_r} \qquad\qquad \text{if } \mathrm{y_i} \neq r \end{cases}$$

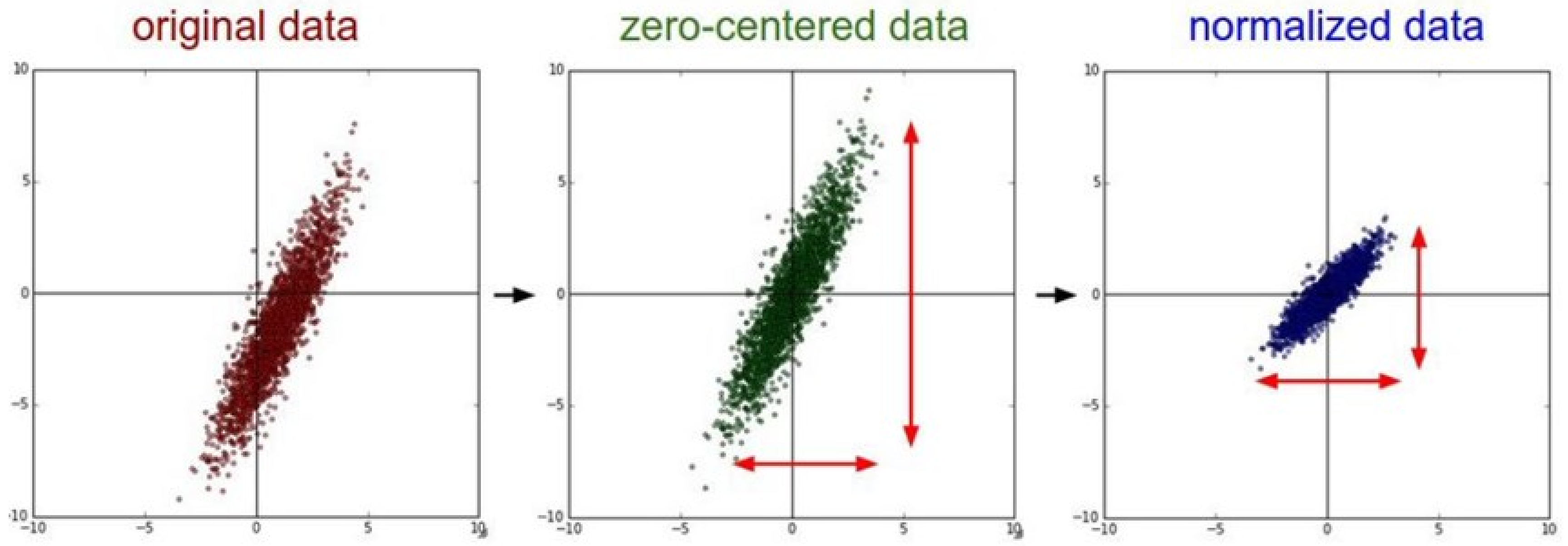
$$\frac{\partial v_\mathrm{r}}{\partial \mathbf{W}_r} = \mathbf{X}_i$$

## A Hint about Initialization

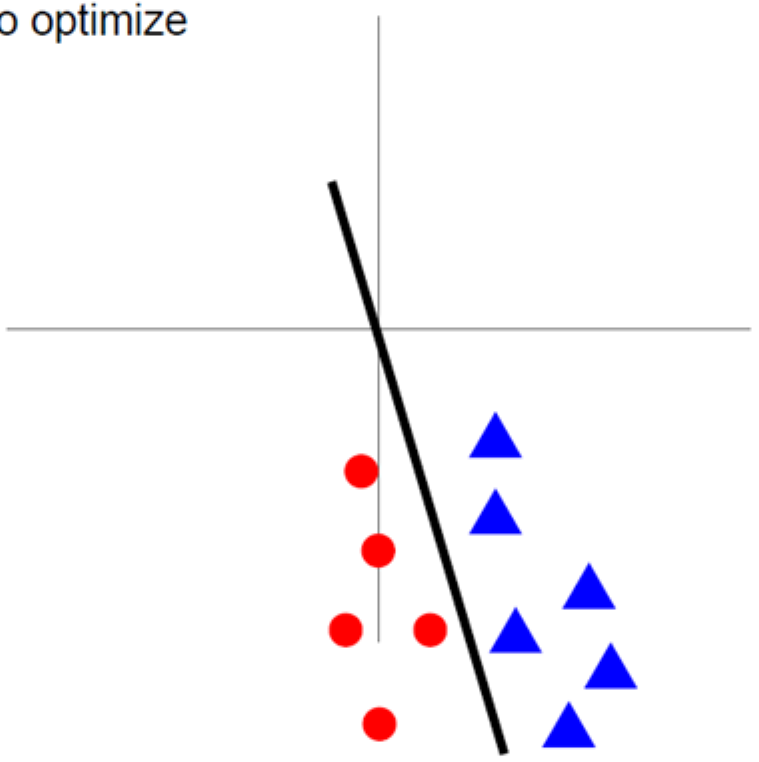
- If weights are very close to zero, you will get zeros functions and hence zero derivatives. There will be no learning.
- Very large values are not recommended to avoid numerical instability. If your are using tanh/sigmoid functions, you get saturated outputs with zero derivatives as well.
- Initialize with small values around zero, to get a nice distribution. Optimization/Learning will go smoothly.



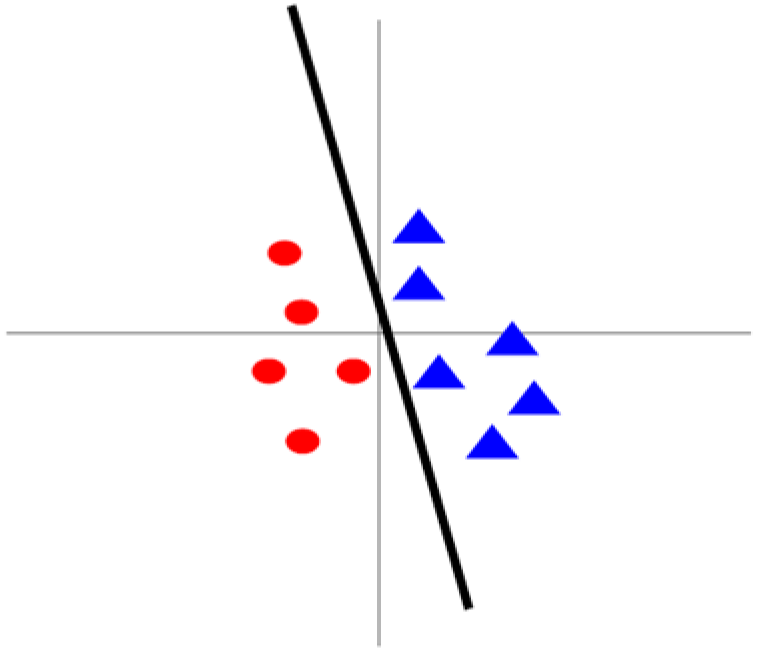
## Training Data Preprocessing



**Before normalization**: classification loss very sensitive to changes in weight matrix; hard to optimize



**After normalization**: less sensitive to small changes in weights; easier to optimize



# Batch Normalization

**Input:** $x : N \times D$

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

**Intermediates:** $\mu, \sigma : D$
$\hat{x} : N \times D$

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$$

**Output:** $y : N \times D$

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$