CS 2750 Machine Learning
Lecture 12

# Evaluation of classifiers
# MLPs

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square



## Evaluation

For any data set we use to test the model we can build a **confusion matrix:**

- Counts of examples with:
- class label $\omega_j$ that are classified with a label $\alpha_i$

| | | target | |
|---|---|---|---|
| | | $\omega = 1$ | $\omega = 0$ |
| **predict** | $\alpha = 1$ | 140 | 17 |
| | $\alpha = 0$ | 20 | 54 |

# Evaluation

For any data set we use to test the model we can build a **confusion matrix:**

| | | target $\omega = 1$ | target $\omega = 0$ |
|---|---|---|---|
| **predict** | $\alpha = 1$ | 140 | 17 |
| | $\alpha = 0$ | 20 | 54 |

agreement

Error: ?



# Evaluation

For any data set we use to test the model we can build a confusion matrix:

| | | model $\omega = 1$ | model $\omega = 0$ |
|---|---|---|---|
| **target** | $\alpha = 1$ | 140 | 17 |
| | $\alpha = 0$ | 20 | 54 |

agreement

**Error:** = 37/231

**Accuracy** = 1 Error = 194/231

# Evaluation

- **Confusion matrix can be built for multi-way classification:**
  - Counts of examples with:
  - class label $\omega_j$ that are classified with a label $\alpha_i$

| | $\omega = 0$ | $\omega = 1$ | $\omega = 2$ |
|---|---|---|---|
| $\alpha = 0$ | 140 | 20 | 22 |
| $\alpha = 1$ | 17 | 54 | 8 |
| $\alpha = 2$ | 12 | 4 | 76 |

agreement



# Evaluation for binary classification

Entries in the confusion matrix for binary classification have names:

**target**

| **predict** | $\omega = 1$ | $\omega = 0$ |
|---|---|---|
| $\alpha = 1$ | *TP* | *FP* |
| $\alpha = 0$ | *FN* | *TN* |

*TP: True positive (hit)*
*FP: False positive (false alarm)*
*TN: True negative (correct rejection)*
*FN: False negative (a miss)*

## Additional statistics

- **Sensitivity (recall)**

$$SENS = \frac{TP}{TP + FN}$$

- **Specificity**

$$SPEC = \frac{TN}{TN + FP}$$

- **Positive predictive value (precision)**

$$PPT = \frac{TP}{TP + FP}$$

- **Negative predictive value**

$$NPV = \frac{TN}{TN + FN}$$



## Binary classification: additional statistics

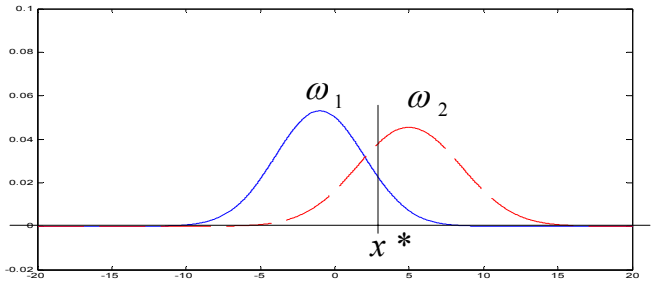
- **Confusion matrix**

| | | target 1 | target 0 | |
|---|---|---|---|---|
| **predict** | 1 | 140 | 10 | $PPV = 140/150$ |
| | 0 | 20 | 180 | $NPV = 180/200$ |
| | | $SENS = 140/160$ | $SPEC = 180/190$ | |

**Row and column quantities:**

- Sensitivity (SENS)
- Specificity (SPEC)
- Positive predictive value (PPV)
- Negative predictive value (NPV)

# Binary decisions: ROC




- **Probabilities:**
  - *SENS* $p(x > x^* \mid \mathbf{x} \in \omega_2)$
  - *SPEC* $p(x < x^* \mid \mathbf{x} \in \omega_1)$



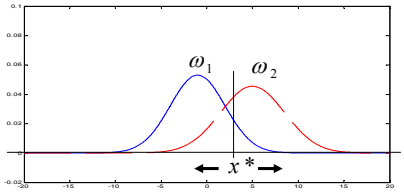
# Receiver Operating Characteristic (ROC)

- **ROC curve plots :**

SN= $p(x > x^* \mid \mathbf{x} \in \omega_2)$

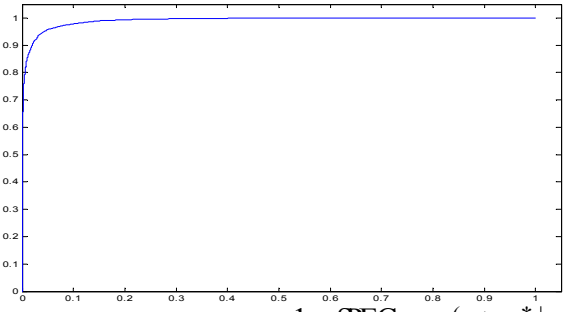
1 SP= $p(x > x^* \mid \mathbf{x} \in \omega_1)$

**for different x***



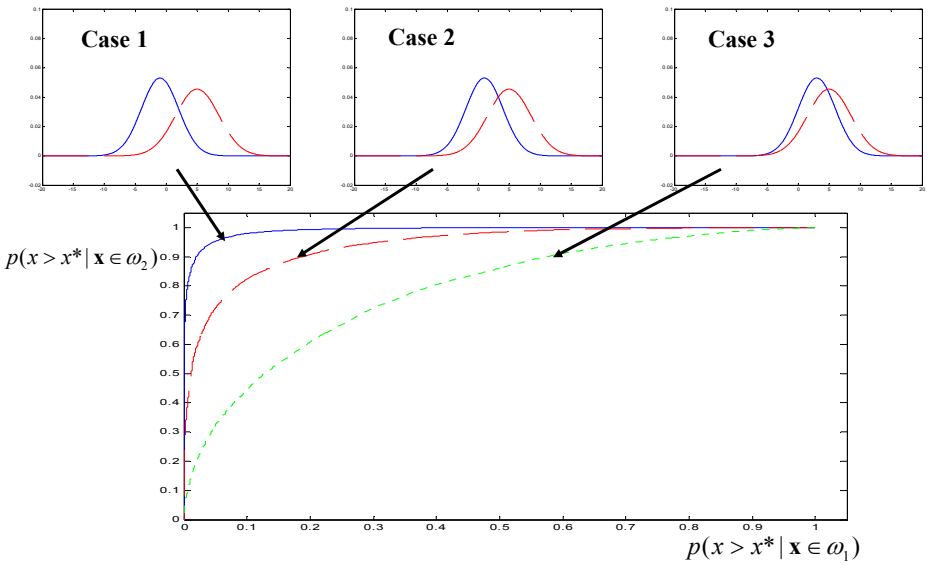

SN

$p(x > x^* \mid \mathbf{x} \in \omega_2)$



1 SPEC $p(x > x^* \mid \mathbf{x} \in \omega_1)$

# ROC curve






# Receiver operating characteristic

- **ROC**
  - shows the discriminability between the two classes under different decision biases
- **Decision bias**
  - can be changed using different loss function

# Zero-one loss function

- **Misclassification error**
  - Based on the zero one loss function
    - Any misclassified example counts as 1
    - Correctly classified example counts as 0

| | $\omega = 0$ | $\omega = 1$ | $\omega = 2$ |
|---|---|---|---|
| $\alpha = 0$ | 140 | 20 | 22 |
| $\alpha = 1$ | 17 | 54 | 8 |
| $\alpha = 2$ | 12 | 4 | 76 |

agreement



# General loss function

- **Error function based on a more general loss function**
  - Different misclassifications have different weight (loss)
  - $\alpha_i$ our choice
  - $\omega_j$ true label
  - $\lambda(\alpha_i \mid \omega_j)$ loss for classification

**Example:**

$\lambda(\alpha_i \mid \omega_j)$

| | $\omega = 0$ | $\omega = 1$ | $\omega = 2$ |
|---|---|---|---|
| $\alpha = 0$ | 0 | 1 | 5 |
| $\alpha = 1$ | 3 | 0 | 2 |
| $\alpha = 2$ | 3 | 1 | 0 |

## Bayesian decision theory

- **More general loss function**
  - Different misclassifications have different weight (loss)

    $\lambda(\alpha_i \mid \omega_j)$
- **Expected loss for the classification choice** $\alpha_i$

$$R(\alpha_i \mid \mathbf{x}) = \sum_j \lambda(\alpha_i \mid \omega_j) P(y = \omega_j \mid \mathbf{x})$$

  - Also called conditional risk
- **Decision rule:** $\alpha(\mathbf{x})$
  - Chooses label (action) according to the input
- **The optimal decision rule**

$$\alpha^*(\mathbf{x}) = \arg\min_{\alpha_i} \sum_j \lambda(\alpha_i \mid \omega_j) P(y = \omega_j \mid \mathbf{x})$$



## Bayesian decision theory

- **The optimal decision rule**

$$\alpha^*(\mathbf{x}) = \arg\min_{\alpha_i} \sum_j \lambda(\alpha_i \mid \omega_j) P(y = \omega_j \mid \mathbf{x})$$

How to modify classifiers to handle different loss?

- **Discriminative models:**
  - Directly optimize the parameters according to the new loss function
- **Generative models:**
  - Learn probabilities as before
  - Decisions about classes are biased to minimize the empirical loss (as seen above)

# Calculating the loss for data

- **Confusion matrix:**
  - Counts of examples with:
  - class label $\omega_j$ that are classified with a label $\alpha_i$

| | $\omega = 0$ | $\omega = 1$ | $\omega = 2$ |
|---|---|---|---|
| $\alpha = 0$ | 140 | 20 | 22 |
| $\alpha = 1$ | 17 | 54 | 8 |
| $\alpha = 2$ | 12 | 4 | 76 |

agreement

- **Loss** $\frac{1}{N}\sum_i \sum_j \lambda(\alpha_i \mid \omega_j) N(\alpha_j \mid \omega_j)$

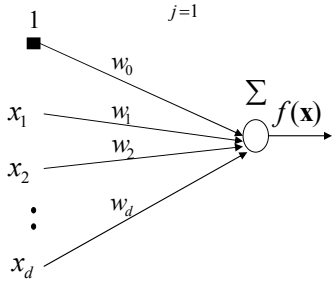


# Multilayer neural networks

# Linear units

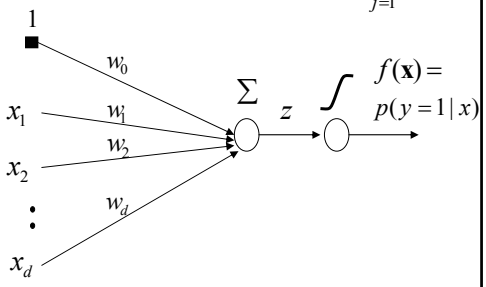
**Linear regression**

$$f(\mathbf{x}) = w_0 + \sum_{j=1}^{d} w_j x_j$$




**Logistic regression**

$$f(\mathbf{x}) = p(y=1|\mathbf{x},\mathbf{w}) = g(w_0 + \sum_{j=1}^{d} w_j x_j)$$




**On-line gradient update:**

$$w_0 \leftarrow w_0 + \alpha(y - f(\mathbf{x}))$$
$$\vdots$$
$$w_j \leftarrow w_j + \alpha(y - f(\mathbf{x}))x_j$$

**The same**

**On-line gradient update:**

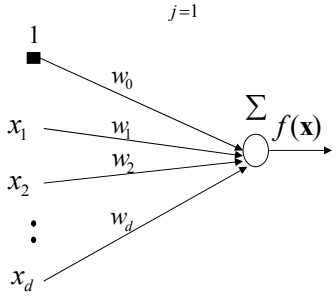
$$w_0 \leftarrow w_0 + \alpha(y - f(\mathbf{x}))$$
$$\vdots$$
$$w_j \leftarrow w_j + \alpha(y - f(\mathbf{x}))x_j$$



# Limitations of basic linear units

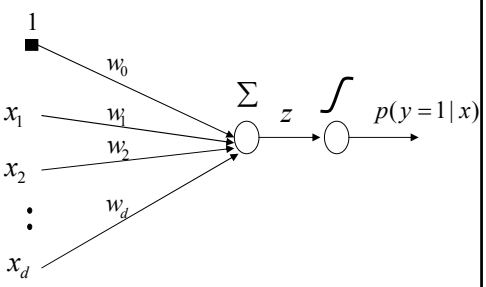
**Linear regression**

$$f(\mathbf{x}) = w_0 + \sum_{j=1}^{d} w_j x_j$$




**Logistic regression**

$$f(\mathbf{x}) = p(y=1|\mathbf{x},\mathbf{w}) = g(w_0 + \sum_{j=1}^{d} w_j x_j)$$



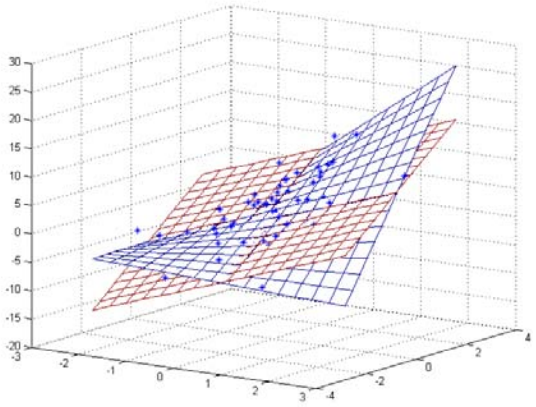

**Function linear in inputs !!**

**Linear decision boundary!!**

# Regression with the quadratic model.

**Limitation:** linear hyper- plane only

- a non- linear surface can be better

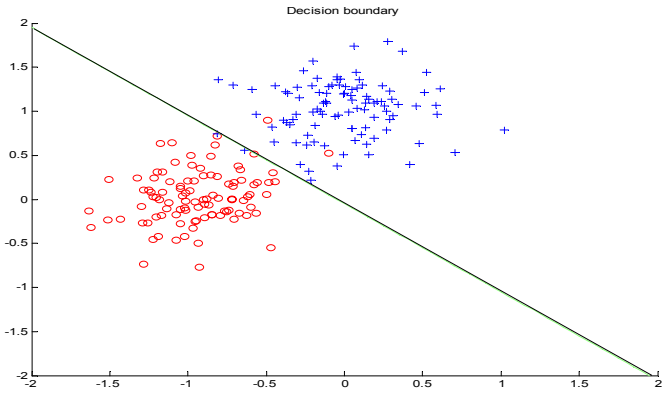




# Classification with the linear model.

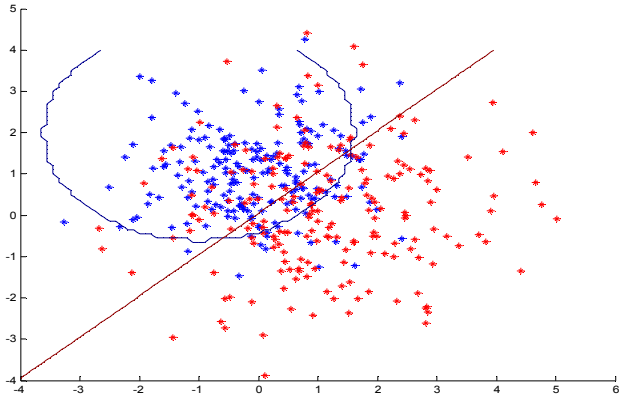
**Logistic regression model defines a linear decision boundary**

- Example: 2 classes (blue and red points)



# Linear decision boundary

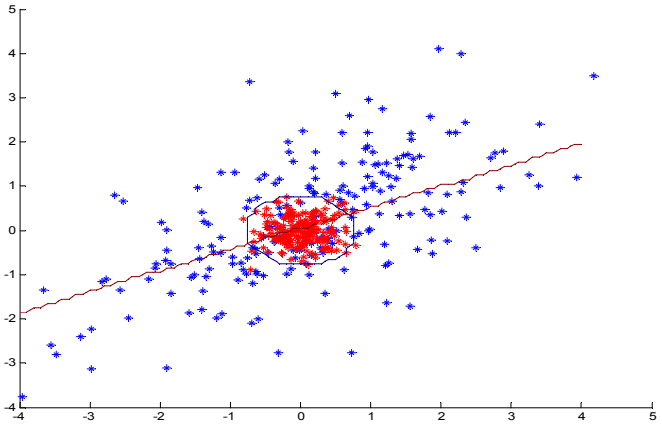
- logistic regression model is not optimal, but not that bad





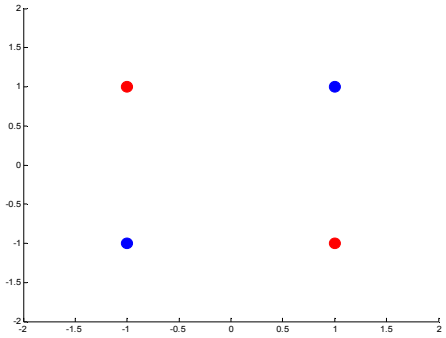

# When logistic regression fails?

- Example in which the logistic regression model fails



# Limitations of linear units.

- Logistic regression does not work for **parity functions**
  - no linear decision boundary exists




**Solution:** a model of a non-linear decision boundary



# Extensions of simple linear units

- use **feature (basis) functions** to model **nonlinearities**

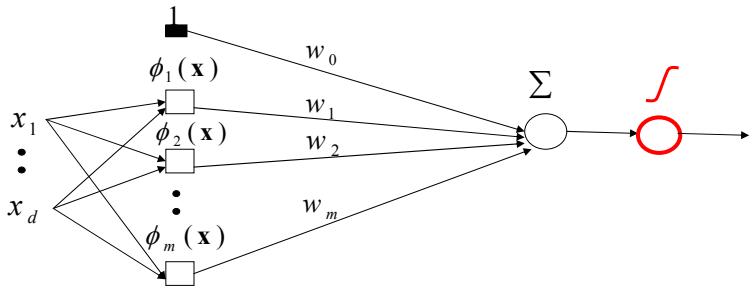
**Linear regression**

$$f(\mathbf{x}) = w_0 + \sum_{j=1}^{m} w_j \phi_j(\mathbf{x})$$

**Logistic regression**

$$f(\mathbf{x}) = g(w_0 + \sum_{j=1}^{m} w_j \phi_j(\mathbf{x}))$$

$\phi_j(\mathbf{x})$ - an arbitrary function of $\mathbf{x}$



# Learning with extended linear units

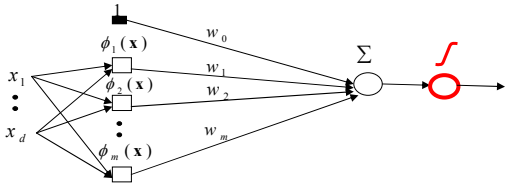
**Feature (basis) functions** model **nonlinearities**

**Linear regression**

$$f(\mathbf{x}) = w_0 + \sum_{j=1}^{m} w_j \phi_j(\mathbf{x})$$

**Logistic regression**

$$f(\mathbf{x}) = g(w_0 + \sum_{j=1}^{m} w_j \phi_j(\mathbf{x}))$$




**Important property:**

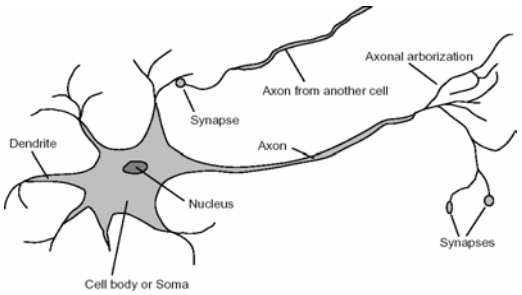
- The same problem as learning of the weights for linear units , the input has changed– but the weights are linear in the new input

**Problem:** too many weights to learn



# Multi-layered neural networks

- Alternative way to introduce nonlinearities to regression/classification models
- **Idea:** Cascade several simple neural models with logistic units. Much like neuron connections.

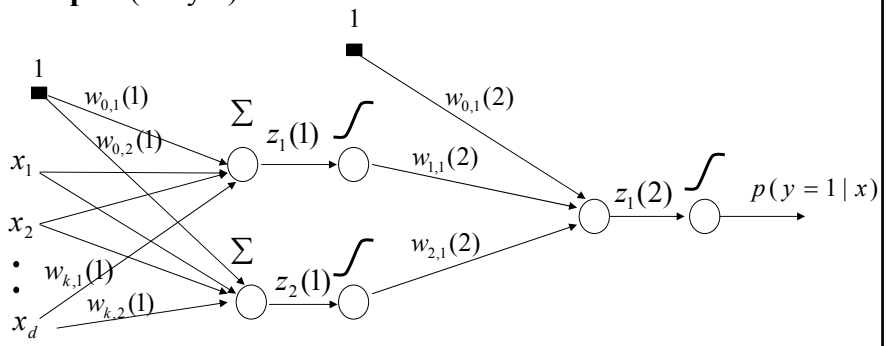


# Multilayer neural network

Also called a **multilayer perceptron (MLP)**

Cascades multiple logistic regression units

**Example:** (2 layer) classifier with non-linear decision boundaries



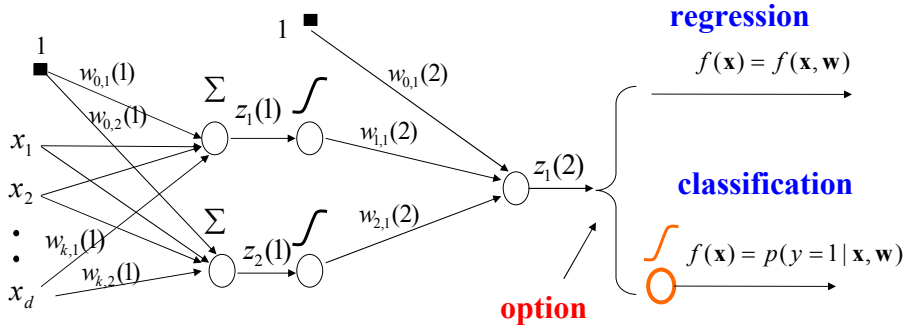

**Input layer** **Hidden layer** **Output layer**



# Multilayer neural network

- Models **non-linearities through logistic regression units**
- Can be applied to both **regression and binary classification problems**

**Input layer** **Hidden layer** **Output layer**



# Multilayer neural network

- **Non-linearities are modeled using multiple hidden logistic regression units (organized in layers)**
- The output layer determines whether it is a **regression or a binary classification problem**