Machine Learning (CE 40717)

Fall 2025

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

October 27, 2025

Binary Classification Problem

- Consider a **binary classification** task:
  - Email classification: Spam / Not Spam
  - Online transactions: Fraudulent / Genuine
  - Tumor diagnosis: Malignant / Benign

Define the target variable formally:

$$y \in \{0, 1\}, \quad \begin{cases} 0 & \text{Negative class (e.g., benign tumor)} \\ 1 & \text{Positive class (e.g., malignant tumor)} \end{cases}$$

Linear Regression for Classification

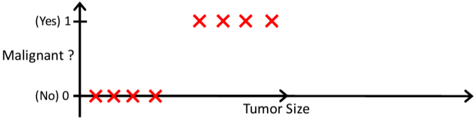
- A natural approach is to use linear regression:

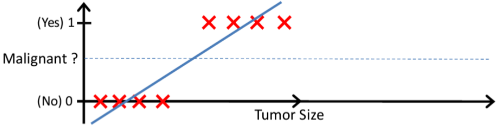
$$h_\theta(x) = \theta_0 + \theta_1 x$$

and define a threshold at 0.5 for prediction:

$$\hat{y} = \begin{cases} 1, & h_\theta(x) \geq 0.5 \\ 0, & h_\theta(x) < 0.5 \end{cases}$$

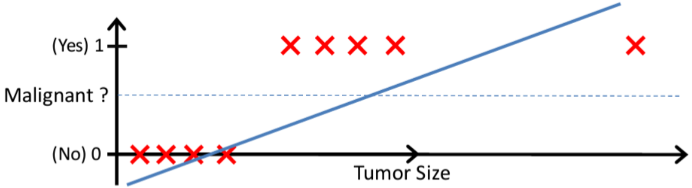
- Here, $\hat{y}$ is the **predicted class label** (i.e., the model's guess for y). It converts the continuous output of $h_\theta(x)$ into a discrete class (0 or 1).





Limitations of Linear Regression for Classification

- The model's output, $h_\theta(x)$, is unbounded and can produce predictions greater than 1 or less than 0.
- Linear regression does not provide probabilistic outputs.
- The decision boundary may be highly sensitive to outliers.



Requirement: $0 \leq h_\theta(x) \leq 1$

Introduction

- Suppose we have a binary classification task (so $K = 2$).
- By observing age, gender, height, weight and BMI we try to distinguish if a person is overweight or not overweight.

Age	Gender	Height (cm)	Weight (kg)	BMI	Overweight
25	Male	175	80	25.3	0
30	Female	160	60	22.5	0
		...			
35	Male	180	90	27.3	1

- We denote the features of a sample with vector x and the label with y.
- In logistic regression we try to find an $\sigma(w^T x)$ which predicts **posterior** probabilities $P(y = 1|x)$.

Introduction (cont.)

- $\sigma(w^T x)$: probability that $y = 1$ given x (parameterized by $\mathbf{w}$)

$$P(y = 1|x, \mathbf{w}) = \sigma(\mathbf{w}^T x)$$
$$P(y = 0|x, \mathbf{w}) = 1 - \sigma(\mathbf{w}^T x)$$

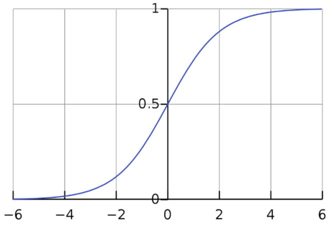
- We need to look for a function which gives us an output in the range [0, 1]. (like a probability).
- Let's denote this function with $\sigma(.)$ and call it the **activation function**.

Introduction (cont.)

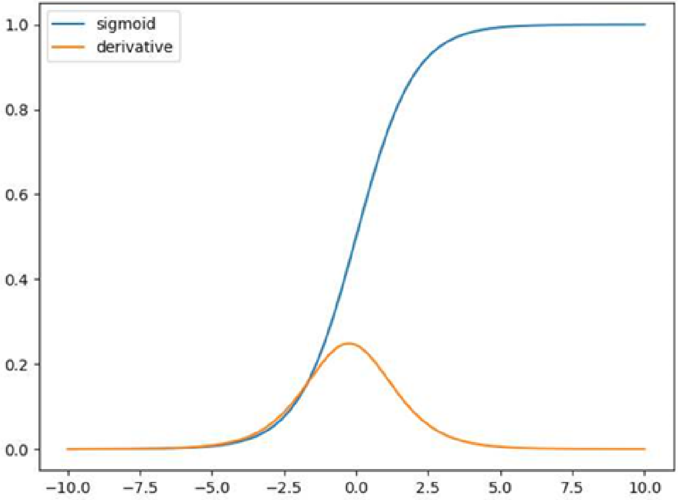
- Sigmoid (logistic) function.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- A good candidate for activation function.
- It gives us a number between 0 and 1 **smoothly**.
- It is also **differentiable**



Sigmoid function & its derivative



Introduction (cont.)

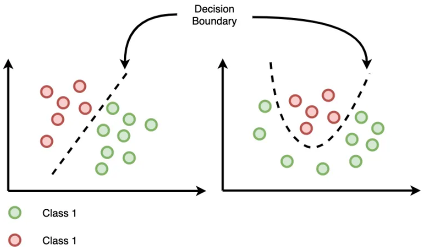
- The sigmoid function takes a number as input but we have:

$$x = [x_0 = 1, x_1, \ldots, x_d]$$
$$w = [w_0, w_1, \ldots, w_d]$$

- So we can use the **dot product** of x and w.
- We have $0 \leq \sigma(\mathbf{w}^T x) \leq 1$. which is the estimated probability of $y = 1$ on input x.
- An Example : A basketball game (Win, Lose)
  - $\sigma(\mathbf{w}^T x) = 0.7$
  - In other terms 70 percent chance of winning the game.

Decision surface

- Decision surface or decision boundary is the region of a problem space in which the output label of a classifier is ambiguous. (could be linear or non-linear)
- In binary classification it is where the probability of a sample belonging to each $y = 0$ and $y = 1$ is equal.



- Decision boundary hyperplane always has **one less dimension** than the feature space.

Decision surface (cont.)

- An example of linear decision boundaries:

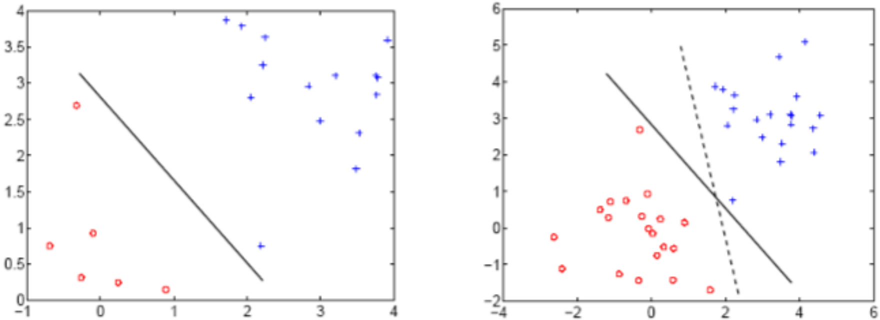


Figure adapted from Eric Xing, Machine Learning, CMU

Decision surface (cont.)

- Back to our logistic regression problem.
- Decision surface $\sigma(\mathbf{w}^T x) =$ **constant**.

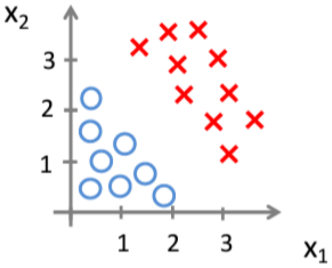
$$\sigma(\mathbf{w}^T x) = \frac{1}{1 + e^{-(\mathbf{w}^T x)}} = 0.5$$

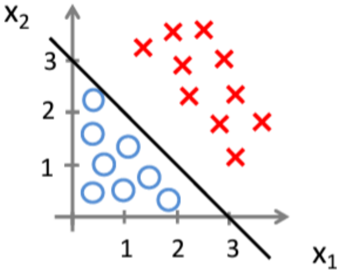
- Decision surfaces are **linear functions** of x
  - if $\sigma(\mathbf{w}^T x) \geq 0.5$ then $\hat{y} = 1$, else $\hat{y} = 0$
  - Equivalently, since $\sigma(z) \geq 0.5$ only when $z \geq 0$, this means:
    - if $\mathbf{w}^T x \geq 0$ then decide $\hat{y} = 1$, else $\hat{y} = 0$

$\hat{y}$ is the predicted label

Decision boundary example

$$\sigma(\mathbf{w}^T x) = \sigma(w_0 + w_1 x_1 + w_2 x_2)$$



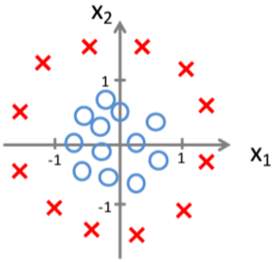


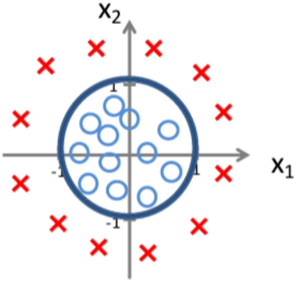
Predict $y = 1$ if $-3 + x_1 + x_2 \geq 0$

Non-linear decision boundary example

$$\sigma(\mathbf{w}^T x) = \sigma(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2)$$

We can learn more complex decision boundaries when having higher order terms





$$\text{Predict } y = 1 \text{ if } -1 + x_1^2 + x_2^2 \geq 0$$

Maximum Likelihood Estimation (MLE)

- For n independent samples, the likelihood is:

$$L(\mathbf{w}) = \prod_{i=1}^{n} P(y^{(i)} | x^{(i)}, \mathbf{w})$$

- For binary classification ($y \in \{0, 1\}$):

$$P(y^{(i)} | x^{(i)}, \mathbf{w}) = \sigma(\mathbf{w}^T x^{(i)})^{y^{(i)}} (1 - \sigma(\mathbf{w}^T x^{(i)}))^{1 - y^{(i)}}$$

- This compact form works because $y^{(i)} \in \{0, 1\}$:
  - If $y^{(i)} = 1$, the expression becomes $P(y = 1 | \ldots) = \sigma(\mathbf{w}^T x^{(i)})$.
  - If $y^{(i)} = 0$, the expression becomes $P(y = 0 | \ldots) = 1 - \sigma(\mathbf{w}^T x^{(i)})$.
- Log-likelihood:

$$\ell(\mathbf{w}) = \sum_{i=1}^{n} \left[y^{(i)} \log \sigma(\mathbf{w}^T x^{(i)}) + (1 - y^{(i)}) \log(1 - \sigma(\mathbf{w}^T x^{(i)})) \right]$$

From Likelihood to Cost Function

- Maximizing the likelihood ⇔ minimizing the negative log-likelihood (NLL):

$$J_{\text{MLE}}(\mathbf{w}) = -\ell(\mathbf{w})$$

- Can be written as an integral over the data distribution:

$$J_{\text{MLE}}(\mathbf{w}) = -\int p(x,y)\log P(y|x,\mathbf{w})\,dx\,dy$$

- Empirical estimate (training data):

$$J_{\text{MLE}}(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^{n}\log P(y^{(i)}|x^{(i)},\mathbf{w})$$

- This is exactly the **cross-entropy loss** used in classification.

Maximum A Posteriori Estimation (MAP)

- MAP incorporates prior knowledge about parameters:

$$\hat{\mathbf{w}}_{\text{MAP}} = \arg\max_{\mathbf{w}} P(\mathbf{w}|D)$$

- Using Bayes' rule:

$$P(\mathbf{w}|D) \propto P(D|\mathbf{w})P(\mathbf{w})$$

- Equivalently:

$$\hat{\mathbf{w}}_{\text{MAP}} = \arg\max_{\mathbf{w}} \left[\log P(D|\mathbf{w}) + \log P(\mathbf{w})\right]$$

- Cost function:

$$J_{\text{MAP}}(\mathbf{w}) = -\sum_{i=1}^{n} \log P(y^{(i)}|x^{(i)}, \mathbf{w}) - \log P(\mathbf{w})$$

MAP with Gaussian Prior (L2 Regularization)

- Gaussian prior: $\mathbf{w} \sim \mathcal{N}(0, \tau^2 I)$

$$P(\mathbf{w}) \propto \exp\left(-\frac{\|\mathbf{w}\|^2}{2\tau^2}\right)$$

- MAP cost function:

$$J_{\text{MAP}}(\mathbf{w}) = J_{\text{MLE}}(\mathbf{w}) + \frac{1}{2\tau^2}\|\mathbf{w}\|^2$$

- Let $\lambda = 1/\tau^2$:

$$J_{\text{MAP}}(\mathbf{w}) = J_{\text{MLE}}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

- Equivalent to **L2-regularized logistic regression**.

MAP with Laplace Prior (L1 Regularization)

- Laplace prior: $\mathbf{w} \sim \text{Laplace}(0, b)$

$$P(\mathbf{w}) \propto \exp\left(-\frac{\|\mathbf{w}\|_1}{b}\right)$$

- MAP cost:

$$J_{\text{MAP}}(\mathbf{w}) = J_{\text{MLE}}(\mathbf{w}) + \lambda \|\mathbf{w}\|_1$$

  where $\lambda = 1/b$

- L1 penalty encourages **sparsity** (feature selection).

Regularization Effects (L1 vs L2)

- **L2 (Ridge):** smooths weights, keeps all features but shrinks them.
- **L1 (Lasso):** drives some weights to zero, performs feature selection.
- Both prevent overfitting by penalizing model complexity.

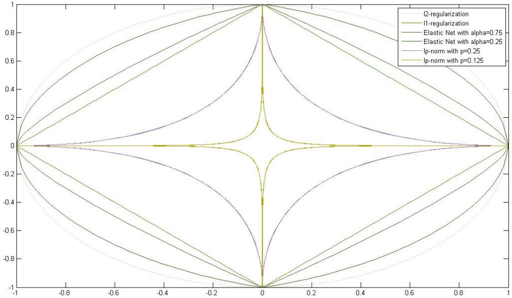


Image adapted from CS 4780, Cornell University

Regularizers: Types and Properties

Type	Form	Properties / Advantages	Disadvantages / Effect
L2	$r(\mathbf{w}) = \|\mathbf{w}\|_2^2$	Strictly convex, differentiable	Dense solutions (uses all features)
L1	$r(\mathbf{w}) = \|\mathbf{w}\|_1$	Convex, encourages sparsity	Not differentiable at 0
Lp	$r(\mathbf{w}) = \|\mathbf{w}\|_p = (\sum_i \lvert v_i \rvert^p)^{1/p}$, $0 < p \leq 1$	Very sparse solutions, initialization dependent	Non-convex, not differentiable

Note: Choice of regularizer affects sparsity and smoothness of learned weights.

MLE vs MAP Comparison

	MLE	**MAP**
Objective	Maximize likelihood $P(D\|\mathbf{w})$	Maximize posterior $P(D\|\mathbf{w})P(\mathbf{w})$
Prior	None (uniform)	Explicit prior $P(\mathbf{w})$
Cost function	$-\log P(D\|\mathbf{w})$	$-\log P(D\|\mathbf{w}) - \log P(\mathbf{w})$
Regularization	None	Arises from prior
Overfitting control	No	Yes

Gradient descent

- Remember from previous slides:

$$J(w) = \sum_{i=1}^{n} -y^{(i)} \log(\sigma(\mathbf{w}^T x^{(i)})) - (1 - y^{(i)}) \log(1 - \sigma(\mathbf{w}^T x^{(i)}))$$

- Update rule for **gradient descent**:

$$w^{t+1} = w^t - \eta \nabla_w J(w^t)$$

- With $J(w)$ definition for logistic regression we get:

$$\nabla_w J(w) = \sum_{i=1}^{n} (\sigma(\mathbf{w}^T x^{(i)}) - y^{(i)}) x^{(i)}$$

Gradient descent

- Compare the gradient of logistic regression with the gradient of SSE in linear regression :

$$\nabla_w J(w) = \sum_{i=1}^{n} (\sigma(\mathbf{w}^T x^{(i)}) - y^{(i)}) x^{(i)}$$

$$\nabla_w J(w) = \sum_{i=1}^{n} (\mathbf{w}^T x^{(i)} - y^{(i)}) x^{(i)}$$

Loss function

- Loss function is a single overall measure of loss incurred for taking our decisions (over entire dataset).
- This is the **cross-entropy** (or log) loss for a single sample:

$$Loss(y, \sigma(\mathbf{w}^T x)) = -y \log(\sigma(\mathbf{w}^T x)) - (1-y)\log(1-\sigma(\mathbf{w}^T x))$$

- Since in binary classification $y \in \{0, 1\}$, the loss simplifies:

$$Loss(y, \sigma(\mathbf{w}^T x)) = \begin{cases} -\log(\sigma(\mathbf{w}^T x)) & \textbf{if } y = 1 \\ -\log(1-\sigma(\mathbf{w}^T x)) & \textbf{if } y = 0 \end{cases}$$

Loss function (cont.)

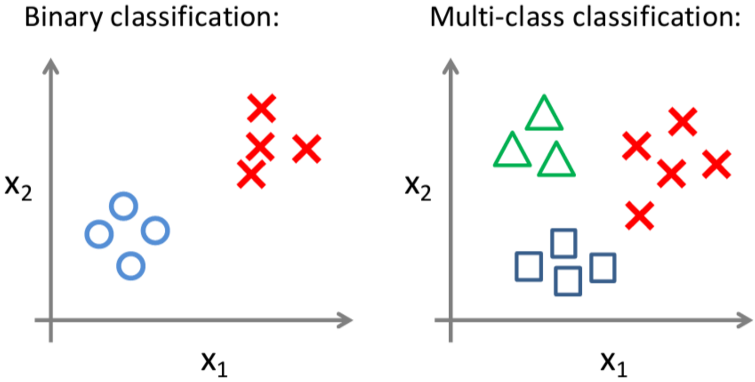
- This is different from the **zero-one loss**, which simply counts misclassifications:

$$\text{Loss}_{0-1}(y, \hat{y}) = \begin{cases} 1 & \textbf{if } y \neq \hat{y} \\ 0 & \textbf{if } y = \hat{y} \end{cases}$$

  ($\hat{y}$ is the predicted label and y is the true label)
- We use cross-entropy (logistic loss) instead of zero-one loss because the zero-one loss function is **non-differentiable** and **non-convex**.
- The cross-entropy loss is a smooth, convex, and differentiable substitute, which allows us to use optimization methods like gradient descent.

Multi-class logistic regression

- Now consider a problem where we have K classes and every sample only belongs to one class (for simplicity).



Multi-class logistic regression (cont.)

- For each class k, $\sigma_k(x;\mathbf{W})$ predicts the probability of $y = k$.
  - i.e., $P(y = k|x, \mathbf{W})$
- For each data point x_0, $\sum_{k=1}^{K} P(y = k|x_0, \mathbf{W})$ must be 1
  - W denotes a matrix of w_i's in which each w_i is a weight vector dedicated for class label i.
- On a new input x, to make a prediction, we pick the class that maximizes $\sigma_k(x;\mathbf{W})$:

$$\alpha(x) = \underset{k=1,\dots,K}{\arg\max}\ \sigma_k(x;\mathbf{W})$$

if $\sigma_k(x;\mathbf{W}) > \sigma_j(x;\mathbf{W})\ \forall j \neq k$ **then decide** C_k

Multi-class logistic regression (cont.)

- $K > 2$ and $y \in \{1, 2, \dots, K\}$

$$\sigma_k(x, \mathbf{W}) = P(y = k|x) = \frac{\exp(w_k^T x)}{\sum_{j=1}^{K} \exp(w_j^T x)}$$

- Normalized exponential (Aka **Softmax**)
- if $w_k^T x \gg w_j^T x$ for all $j \neq k$ then $P(C_k|x) \approx 1$ and $P(C_j|x) \approx 0$
- Note : remember from Bayes theorem:

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{\sum_{j=1}^{K} P(x|C_j)P(C_j)}$$

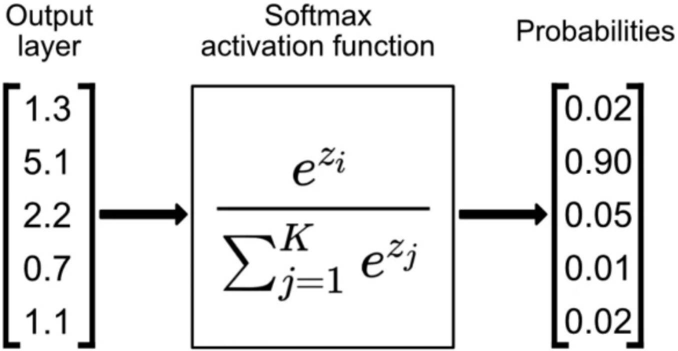
Multi-class logistic regression (cont.)

- Softmax function **smoothly** highlights the maximum probability and is differentiable.
- Compare it with max(.) function which is strict and non-differentiable
- Softmax can also handle negative values because we are using exponential function
- And it gives us probability for each class since:

$$\sum_{k=1}^{K} \frac{\exp(w_k^T x)}{\sum_{j=1}^{K} \exp(w_j^T x)} = 1$$

Multi-class logistic regression (cont.)

- An example of applying softmax (note that $z_i = w^T x_i$):



Multi-class logistic regression (cont.)

- Again we set $J(W)$ as negative of log likelihood.
- We need $\hat{W} = \underset{W}{\arg\min}\ J(W)$

$$
\begin{aligned}
J(W) &= -\log \prod_{i=1}^{n} P(y^{(i)} | x^{(i)}, \mathbf{W}) \\
&= -\log \prod_{i=1}^{n} \prod_{k=1}^{K} \sigma_k(x^{(i)}; \mathbf{W})^{y_k^{(i)}} \\
&= -\sum_{i=1}^{n} \sum_{k=1}^{K} y_k^{(i)} \log(\sigma_k(x^{(i)}; \mathbf{W}))
\end{aligned}
$$

- If **i-th** sample belongs to class k then $y_k^{(i)}$ is 1 else 0.
- Again no closed-from solution for $\hat{W}$

Multi-class logistic regression (cont.)

- From previous slides we have:

$$J(W) = -\sum_{i=1}^{n}\sum_{k=1}^{K} y_k^{(i)} \log(\sigma_k(x^{(i)};\mathbf{W}))$$

- In which:

$$W = [w_1, w_2, \dots, w_K], \quad Y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{pmatrix} = \begin{pmatrix} y_1^{(1)} & \dots & y_K^{(1)} \\ y_1^{(2)} & \dots & y_K^{(2)} \\ \vdots & \ddots & \vdots \\ y_1^{(n)} & \dots & y_K^{(n)} \end{pmatrix}$$

- y is a vector of length K (1-of-K encoding)
  - For example $y = [0, 0, 1, 0]^T$ when the target class is C_3.

Multi-class logistic regression (cont.)

- Update rule for gradient descent:

$$w_j^{t+1} = w_j^t - \eta \nabla_W J(W^t)$$
$$\nabla_{w_j} J(W) = \sum_{i=1}^{n} (\sigma_j(x^{(i)}; \mathbf{W}) - y_j^{(i)}) x^{(i)}$$

- w_j^t denotes the weight vector for class j (since in multi-class LR, each class has its own weight vector) in the t-th iteration

1. Introduction
2. Logistic Regression
3. Extra reading
   - Probabilistic view in classification
   - Probabilistic classifiers
4. References

Probabilistic view in classification problem

- In a classification problem:
  - Each **feature** is a **random variable** (e.g. a person's height)
  - The **class label** is also considered a **random variable** (e.g. a person could be overweight or not)
- We observe the feature values for a random sample and intend to find its class label
  - Evidence: Feature vector x
  - Objective: Class label

Definitions

- Posterior probability : The probability of a class label C_k given a sample x

$$P(C_k|x)$$

- Likelihood or class conditional probability : PDF of feature vector x for samples of class C_k

$$P(x|C_k)$$

- Prior probability : Probability of the label be C_k

$$P(C_k)$$

- $P(x)$: PDF of feature vector x
  - From total probability theorem:

$$P(x) = \sum_{k=1}^{K} P(x|C_k)P(C_k)$$

Probabilistic classifiers

- Probabilistic approaches can be divided in two main categories:
    - Generative
        - Estimate PDF $P(x, C_k)$ for each class C_k and then use it to find $P(C_k|x)$. Alternatively estimate both PDF $P(x|C_k)$ and $P(C_k)$ to find $P(C_k|x)$.
    - Discriminative
        - Directly estimate $P(C_k|x)$ for class C_k

Probabilistic classifiers (cont.)

- Let's assume we have input data x and want to classify the data into labels y.
- A generative model learns the **joint** probability distribution $P(x, y)$.
- A discriminative model learns the **conditional** probability distribution $P(y|x)$

Discriminative vs. Generative : example

- Suppose we have the following dataset in form of (x, y):

$$(1,0),(1,0),(2,0),(2,1)$$

- $P(x, y)$ is :

	$y=0$	$y=1$
$x=1$	$\frac{1}{2}$	0
$x=2$	$\frac{1}{4}$	$\frac{1}{4}$

- $P(y|x)$ is :

	$y=0$	$y=1$
$x=1$	1	0
$x=2$	$\frac{1}{2}$	$\frac{1}{2}$

Discriminative vs. Generative : example (cont.)

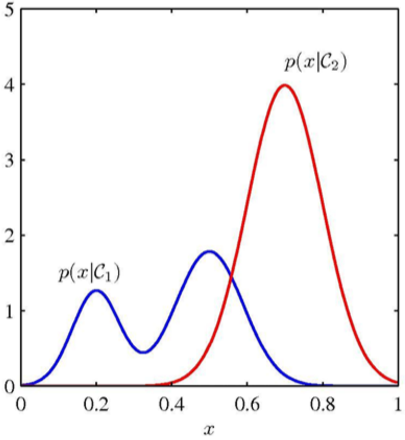
- The distribution $P(y|x)$ is the natural distribution for classifying a given sample x into class y.
  - This is why that algorithms which model this directly are called **discriminative** algorithms.
- Generative algorithms model $P(x, y)$, which can be transformed into $P(y|x)$ by Bayes rule and then used for classification.
  - However, the distribution $P(x, y)$ can also be used for other purposes.
  - For example we can use $P(x, y)$ to **generate** likely (x, y) pairs

Generative approach

1. Inference
   - Determine class conditional densities $P(x|C_k)$ and priors $P(C_k)$
   - Use Bayes theorem to find $P(C_k|x)$
2. Decision
   - Make optimal assignment for new input (after learning the model in the inference stage)
   - if $P(C_i|x) > P(C_j|x) \forall j \neq i$, then decide C_i .

Generative approach (cont.)

- Generative approach for a binary classification problem:



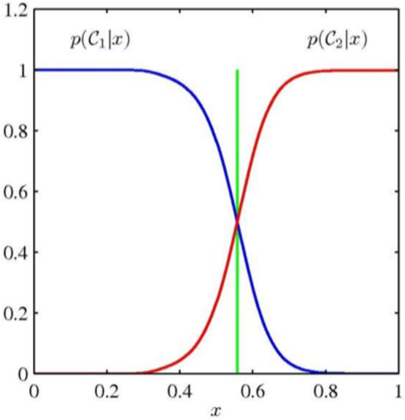
Figures adapted from Machine Learning and Pattern Recognition, Bishop

Discriminative approach

1. Inference
   - Determine the posterior class probabilities $P(C_k|x)$ directly.
2. Decision
   - Make optimal assignment for new input (after learning the model in the inference stage)
   - if $P(C_i|x) > P(C_j|x) \forall j \neq i$, then decide C_i .

Discriminative approach (cont.)

- Discriminative approach for a binary classification problem:



Figures adapted from Machine Learning and Pattern Recognition, Bishop

Contributions

- **These slides are authored by:**
  - Danial Gharib

[1] M. Soleymani Baghshah, "Machine learning." Lecture slides.

[2] A. Ng, "Ml-005, lecture 6." Lecture slides.

[3] C. M. Bishop, *Pattern Recognition and Machine Learning*. Information Science and Statistics, New York, NY: Springer, 1 ed., Aug. 2006.

[4] S. Fidler, "Csc411." Lecture slides.

[5] A. Ng and T. Ma, *CS229 Lecture Notes*. Updated June 11, 2023.