# Classification problem

## Multinomial classificataion, softmax function

Khiem Nguyen

| | |
|---|---|
| Email | khiem.nguyen@glasgow.ac.uk |
| MS Teams | khiem.nguyen@glasgow.ac.uk |
| Whatsapp | +44 7729 532071 (Emergency only) |


May 18, 2025

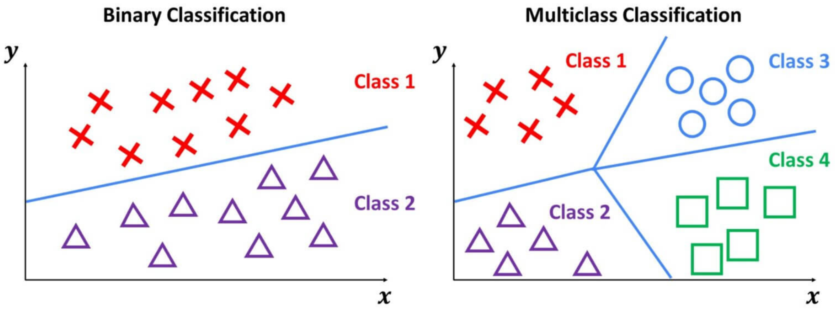
University of Glasgow

# Table of Contents

# Background: Some application examples

- Which major will a college student choose, given grades, stated likes and dislikes, etc.?
- Which blood type does a person have, given the results of various diagnostic tests?
- Which candidate will a person vote for, given particular demographic characteristics?
- Which fruit does an image show? – One image shows one fruit.

# Background: Illustration



# Background: Terminologies

- *Multiclass (classification) problem*: The problem of classiying instances/examples into one of three or more classes (see binary classification problem in last lecture)

- *Multinomial logistic regression*: A classification method that generalizes logistic regression to multiclass problems

- Some different names of multinomial logistic regression:
  - *polytomous logistic regression*
  - *softmax regression* – The model representation is established based on softmax functions.
  - *multinomial logit*
  - *maximum entropy* (MaxEnt) classifier – the Loss function is based on the maximum entropy.

For those who are interested:

Maximum entropy probability distribution: Wikipedia - Click on me!

# Classification for multiclass problems

In the binary classification we consider:

$$\underbrace{\mathrm{P}(y=0|\mathbf{x})}_{1-f_{\mathbf{w},b}(\mathbf{x})}+\underbrace{\mathrm{P}(y=1|\mathbf{x})}_{f_{\mathbf{w},b}(\mathbf{x})}=1,$$

where $\mathrm{P}(y=k|\mathbf{x})$, $k=0,1$, is the probability that $y=k$ given the input example $\mathbf{x}$.

The logistic regression can be generalized to multiclass problems, i.e. with more than two possible discrete outcomes

- $K>2$ classes/categories: $y \in \{1,2,\dots,K\}$
- $n$ input features: $\mathbf{x}=(x_1,x_2,\dots,x_n)$.

Disclaimer

- In statistics, we normally use the capitalized $Y$ to imply it is a *Random Variable*. *Thus, you may encounter in the literature the notation* $\mathrm{P}(Y=k)$ *with* $Y$ *being the random variable.*
- We don't need to understand these technical terminologies in detail. *What we write here is not necessarily mathematically rigorous but it expresses the right ideas and formulation.*

# Classification for multiclass problems: Notation

- There are $K$ classes/categories the response $y$ can belong to
- **Data set**: $m$ training examples with inputs of $n$ features and $K$ classes

$$\begin{aligned} \text{dataset} \quad & \mathrm{D} = \left\{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})\right\}, \\ \text{independent variables/inputs} \quad & \mathbf{x}^{(i)} = (x_1^{(i)}, \dots, x_n^{(i)}) \in \mathbb{R}^n, \\ \text{dependent variables/outputs} \quad & y^{(i)} \in \{1, \dots, K\} \end{aligned}$$

- Probability that the outcome $y$ given the input example $\mathbf{x}$ is of class $k$:

$$\mathrm{P}(y = k|\mathbf{x})$$

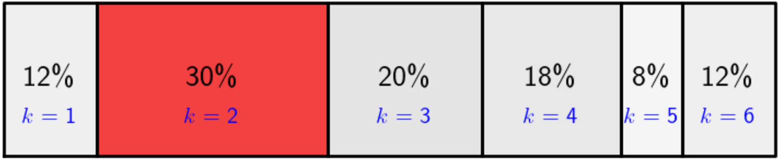
*Set of probabilities forms a probability distribution*

$$\sum_{k=1}^{K} \mathrm{P}(y = k|\mathbf{x}) = 1$$

# Table of Contents

# Making prediction

- To make prediction on which class one particular example $\mathbf{x}$ is
  - Calculate all the $K$ probabilities $\mathrm{P}(y = k|\mathbf{x})$, $k = 1, \ldots, K$
  - Vote for the highest probability.




Example: $\hat{y} = 2$ as $\mathrm{P}(y = 2|\mathbf{x})$ is the highest probability among all $\mathrm{P}(y = k|\mathbf{x})$ $\forall k = 1, \ldots, K$.

---

We make prediction $\hat{y}^{(i)} = m$

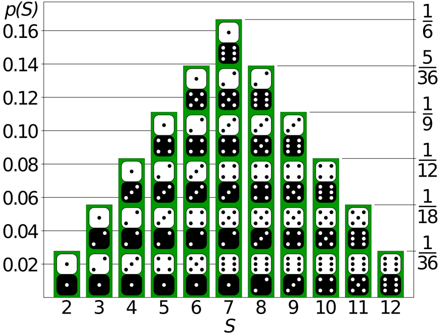
with $\mathrm{P}(y = m|\mathbf{x}^{(i)})$ being the highest probability among all $\mathrm{P}(y = k|\mathbf{x}^{(i)})$, $k = 1, \ldots K$

$$\hat{y} = m \quad \Leftrightarrow \quad \mathrm{P}(y = m|\mathbf{x}^{(i)}) = \max\left\{\mathrm{P}(y = 1|\mathbf{x}^{(i)}), \ldots, \mathrm{P}(y = K|\mathbf{x}^{(i)})\right\}$$

# Probability distribution

**_Set of probabilities forms a probability distribution_**

$$\sum_{k=1}^{K} \mathrm{P}(y = k|\mathbf{x}) = 1$$



# Softmax functions

$$\sum_{k=1}^{K} \mathrm{P}(y = k|\mathbf{x}) = 1$$

# Softmax functions

$$\boxed{\sum_{k=1}^{K} \mathrm{P}(y = k|\mathbf{x}) = 1}$$

➥ A widely used technique for representing this formulation: softmax formulation

$$\mathrm{softmax}(k; z_1, \dots, z_K) = \frac{\mathrm{e}^{z_k}}{\sum_{l=1}^{K} \mathrm{e}^{z_l}}$$

$$\Rightarrow \quad \sum_{l=1}^{K} \mathrm{softmax}(k, z_1, \dots, z_K) = \underbrace{\left(\mathrm{e}^{z_1} + \cdots + \mathrm{e}^{z_K}\right)}_{\sum_{k=1}^{K} \mathrm{e}^{z_k}} \Big/ \left(\sum_{l=1}^{K} \mathrm{e}^{z_l}\right) = 1$$

$$\mathrm{softmax}(1; \dots) = \frac{\mathrm{e}^{z_1}}{\sum_{l=1}^{K} \mathrm{e}^{z_l}} \quad \Bigg| \quad \mathrm{softmax}(2; \dots) = \frac{\mathrm{e}^{z_2}}{\sum_{l=1}^{K} \mathrm{e}^{z_l}} \quad \Bigg| \dots \Bigg| \quad \mathrm{softmax}(K; \dots) = \frac{\mathrm{e}^{z_K}}{\sum_{l=1}^{K} \mathrm{e}^{z_l}}$$

# Model representation

- Following the above explanation:

$$\mathrm{P}(y = k|\mathbf{x}) = \mathrm{softmax}(k; z_1, \ldots, z_K)$$

- Probability that $y = k$ given an input $\mathbf{x}$:

$$\mathrm{P}(y = k|\mathbf{x}) = \frac{\mathrm{e}^{\mathbf{w}_k \cdot \mathbf{x} + b_k}}{\sum_{l=1}^{K} \mathrm{e}^{\mathbf{w}_l \cdot \mathbf{x} + b_l}}$$

- With each $\mathrm{P}(y = k|\mathbf{x})$ we associate the model coefficients

$$\mathbf{w}_k = (w_{k1}, \cdots, w_{kn}) \in \mathbb{R}^n, \quad b_k \in \mathbb{R}$$

- For $K$ classes, we have $\mathbf{w}_1, \ldots, \mathbf{w}_K$ and $b_1, \ldots b_K$

  $\rightarrow \quad (K \times n) + K = K \times (n + 1)$ model parameters.

# Decision boundaries

- The decision boundary between the class $\#j$ ($C_j$) and and the class $\#k$ ($C_k$) is given by

$$\mathrm{P}(y=j|\mathbf{x}) = \mathrm{P}(y=k|\mathbf{x}) \Leftrightarrow \frac{\mathrm{e}^{\mathbf{w}_j \cdot \mathbf{x} + b_j}}{\sum_{l=1}^{K} \mathrm{e}^{\mathbf{w}_l \cdot \mathbf{x} + b_l}} = \frac{\mathrm{e}^{\mathbf{w}_k \cdot \mathbf{x} + b_k}}{\sum_{l=1}^{K} \mathrm{e}^{\mathbf{w}_l \cdot \mathbf{x} + b_l}} \Leftrightarrow \mathbf{w}_j \cdot \mathbf{x} + b_j = \mathbf{w}_k \cdot \mathbf{x} + b_k$$

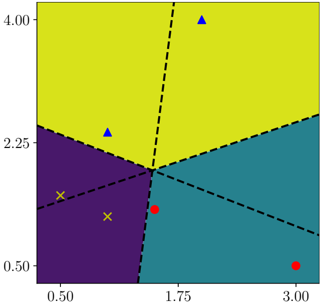
It's a hyperplane: $(\mathbf{w}_k - \mathbf{w}_j) \cdot \mathbf{x} + (b_k - b_j) = 0.$

# Decision boundaries

➥ The decision boundary between the class $\#j$ ($C_j$) and and the class $\#k$ ($C_k$) is given by

$$\mathrm{P}(y=j|\mathbf{x}) = \mathrm{P}(y=k|\mathbf{x}) \Leftrightarrow \frac{\mathrm{e}^{\mathbf{w}_j\cdot\mathbf{x}+b_j}}{\sum_{l=1}^{K}\mathrm{e}^{\mathbf{w}_l\cdot\mathbf{x}+b_l}} = \frac{\mathrm{e}^{\mathbf{w}_k\cdot\mathbf{x}+b_k}}{\sum_{l=1}^{K}\mathrm{e}^{\mathbf{w}_l\cdot\mathbf{x}+b_l}} \Leftrightarrow \mathbf{w}_j\cdot\mathbf{x}+b_j = \mathbf{w}_k\cdot\mathbf{x}+b_k$$

It's a hyperplane: $(\mathbf{w}_k - \mathbf{w}_j)\cdot\mathbf{x} + (b_k - b_j) = 0.$



# Table of Contents

# Loss function

- **Model parameters/Trainable parameters**:

$$\mathbf{W} = \begin{bmatrix} —\mathbf{w}_1— \\ —\mathbf{w}_2— \\ \vdots \\ —\mathbf{w}_K— \end{bmatrix} = \begin{bmatrix} w_{11} & w_{21} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{K1} & w_{K2} & \cdots & w_{Kn} \end{bmatrix}, \qquad \mathbf{b} = (b_1, b_2, \cdots, b_K)$$

**Loss function**

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} \delta_{k,y^{(i)}} \log \left[ \mathrm{P}(y = k | \mathbf{x}^{(i)}) \right]$$

$$\delta_{k,y^{(i)}} = \begin{cases} 1 & \text{if } k = y^{(i)} \\ 0 & \text{if } k \neq y^{(i)} \end{cases}$$

# Loss function: A bit more explanation

$$\begin{aligned}\mathcal{L}(\mathbf{W}, \mathbf{b}) &= -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\delta_{k,y^{(i)}}\log\left[\mathrm{P}(y=k|\mathbf{x}^{(i)})\right] \\ &= -\frac{1}{m}\sum_{i=1}^{m}\Big\{\delta_{1,y^{(i)}}\log\left[\mathrm{P}(y=1|\mathbf{x}^{(i)})\right] + \delta_{2,y^{(i)}}\log\left[\mathrm{P}(y=2|\mathbf{x}^{(i)})\right] \\ &\qquad + \cdots + \delta_{K-1,y^{(i)}}\log\left[\mathrm{P}(y=K-1|\mathbf{x}^{(i)})\right] + \delta_{K,y^{(i)}}\log\left[\mathrm{P}(y=K|\mathbf{x}^{(i)})\right]\Big\}\end{aligned}$$

and

$$\begin{aligned}\delta_{k,y^{(i)}} &= \begin{cases}1 & \text{if } k = y^{(i)} \\ 0 & \text{if } k \neq y^{(i)}\end{cases} \\ &= \begin{cases}\mathrm{True} & \text{if } y^{(i)} \text{is of class } k \\ \mathrm{False} & \text{if } y^{(i)} \text{is NOT of class } k\end{cases}\end{aligned}$$

# Loss function: binary classification

- Let us revisit the binary classification problem

$$\mathrm{P}(y=0|\mathbf{x}^{(i)}) = \frac{e^{\mathbf{w}_0\cdot\mathbf{x}^{(i)}+b_0}}{e^{\mathbf{w}_0\cdot\mathbf{x}^{(i)}+b_0}+e^{\mathbf{w}_1\cdot\mathbf{x}^{(i)}+b_1}}, \quad \mathrm{P}(y=1|\mathbf{x}^{(i)}) = \frac{e^{\mathbf{w}_1\cdot\mathbf{x}^{(i)}+b_1}}{e^{\mathbf{w}_0\cdot\mathbf{x}^{(i)}+b_0}+e^{\mathbf{w}_1\cdot\mathbf{x}^{(i)}+b_1}}$$

$$\mathbf{W} = \begin{bmatrix} w_{01} & w_{02} & \cdots & w_{0n} \\ w_{11} & w_{21} & \cdots & w_{1n} \end{bmatrix}, \quad \mathbf{b} = (b_0, b_1)$$

- **Loss function**

$$\begin{aligned} \mathcal{L}(\mathbf{W},\mathbf{b}) &= -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=0}^{1}\delta_{k,y^{(i)}}\log(\mathrm{P}(y^{(i)}=k)) \\ &= -\frac{1}{m}\sum_{i=1}^{m}\left\{\delta_{0,y^{(i)}}\log(\mathrm{P}(y^{(i)}=0))+\delta_{1,y^{(i)}}\log(\mathrm{P}(y^{(i)}=1))\right\} \end{aligned}$$

$$\delta_{0,y^{(i)}} = \begin{cases} 1 & \text{if } y^{(i)}=0 \\ 0 & \text{if } y^{(i)}=1 \end{cases} \Rightarrow \delta_{0,y^{(i)}} = 1-y^{(i)}, \quad \delta_{1,y^{(i)}} = \begin{cases} 1 & \text{if } y^{(i)}=1 \\ 0 & \text{if } y^{(i)}=0 \end{cases} \Rightarrow \delta_{1,y^{(i)}} = y^{(i)}$$

## Loss function: binary classification

- **Model function**

$$\mathrm{P}(y=1|\mathbf{x}^{(i)}) = \frac{\mathrm{e}^{\mathbf{w}_1\cdot\mathbf{x}^{(i)}+b_1}}{\mathrm{e}^{\mathbf{w}_0\cdot\mathbf{x}^{(i)}+b_0}+\mathrm{e}^{\mathbf{w}_1\cdot\mathbf{x}^{(i)}+b_1}} = \frac{1}{1+\mathrm{e}^{\mathbf{w}_0\cdot\mathbf{x}^{(i)}+b_0-\mathbf{w}_1\cdot\mathbf{x}^{(i)}-b_1}}$$
$$= \frac{1}{1+\mathrm{e}^{(\mathbf{w}_0-\mathbf{w}_1)\cdot\mathbf{x}^{(i)}+b_0-b_1}}$$
$$= \frac{1}{1+\mathrm{e}^{-(\mathbf{w}\cdot\mathbf{x}^{(i)}+b)}} = f_{\mathbf{w},b}(\mathbf{x}^{(i)}), \quad \text{with } \mathbf{w}=\mathbf{w}_1-\mathbf{w}_0, \quad b=b_1-b_0$$

$$\Rightarrow \quad \mathrm{P}(y=1|\mathbf{x}^{(i)}) = f_{\mathbf{w},b}(\mathbf{x}^{(i)}), \quad \mathrm{P}(y=0|\mathbf{x}^{(i)}) = 1-f_{\mathbf{w},b}(\mathbf{x}^{(i)})$$

- **Loss function**

$$\mathcal{L}(\mathbf{W},\mathbf{b}) = -\frac{1}{m}\sum_{i=1}^{m}\left\{\underbrace{\delta_{0,y^{(i)}}}_{1-y^{(i)}}\log(\underbrace{\mathrm{P}(y^{(i)}=0)}_{1-f_{\mathbf{w},b}(\mathbf{x}^{(i)})}) + \underbrace{\delta_{1,y^{(i)}}}_{y^{(i)}}\log(\underbrace{\mathrm{P}(y^{(i)}=1)}_{f_{\mathbf{w},b}(\mathbf{x}^{(i)})})\right\}$$
$$= -\frac{1}{m}\sum_{i=1}^{m}\left\{(1-y^{(i)})\log\left(1-f_{\mathbf{w},b}(\mathbf{x}^{(i)})\right) + y^{(i)}\log\left(f_{\mathbf{w},b}(\mathbf{x}^{(i)})\right)\right\}$$

# Gradient descent

- We need to compute the gradient of $\mathcal{L}(\mathbf{W}, \mathbf{b}) = \mathcal{L}(w_{11}, \dots w_{Kn}, b_1, \dots, b_K)$:

$$\frac{\partial J}{\partial w_{kj}} = \frac{1}{m} \sum_{i=1}^{m} \left(p_k^{(i)} - \delta_{k,y^{(i)}}\right) x_j^{(i)},$$

$$\frac{\partial J}{\partial b_k} = \frac{1}{m} \sum_{i=1}^{m} \left(p_k^{(i)} - \delta_{k,y^{(i)}}\right) 1$$

where

$$p_k^{(i)} = \mathrm{P}(y = k | \mathbf{x}^{(i)}) = \frac{\exp(\mathbf{w}_k \cdot \mathbf{x}^{(i)} + b_k)}{\sum_{l=1}^{K} \exp(\mathbf{w}_l \cdot \mathbf{x}^{(i)} + b_l)}$$

- If we set $w_{k0}^{(i)} = 1, \forall i = 1, \dots, n, \forall l = 1, \dots, K$, we can say:

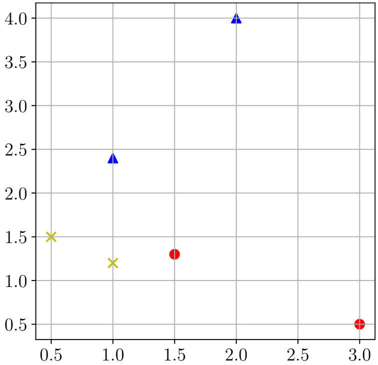
  The derivative of $\mathcal{L}$ is the error times the input.

# Table of Contents

## A simple problem with three classes

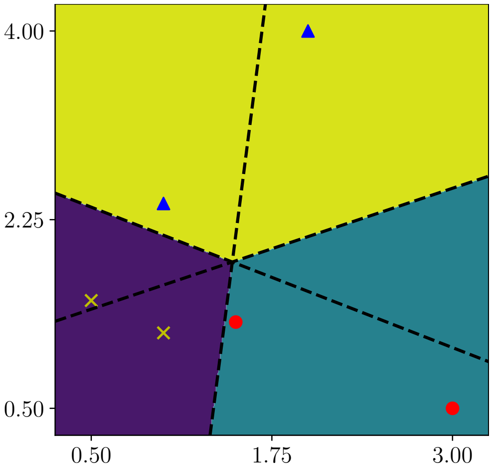
```
X_train = np.array([[0.5, 1.5], [1, 1.2], [1.5, 1.3], [3, 0.5], [2, 4], [1, 2.4]])
y_train = np.array([0, 0, 1, 1, 2, 2])
# Visualization
m, n = X_train.shape
N = len(np.unique(y_train))
markers = ['x', 'o', '^']
plt.figure(figsize=(4, 3))
for j in range(N):
    idx = (y_train == j)
    plt.scatter(X_train[idx,0], X_train[idx,1], marker=markers[j])

from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression(multi_class='multinomial')
lr_model.fit(X_train, y_train)
lr_model.predict(X_train)                # make prediction
lr_model.predict_prob(X_train)           # predict the probabilities

print(f"W = {lr_model.coef_}")           # 2D array: shape = (3, 2)
print(f"b = {lr_model.intercept_}")      # 1D array: shape = (3,)
```

# A simple problem with three classes: Results



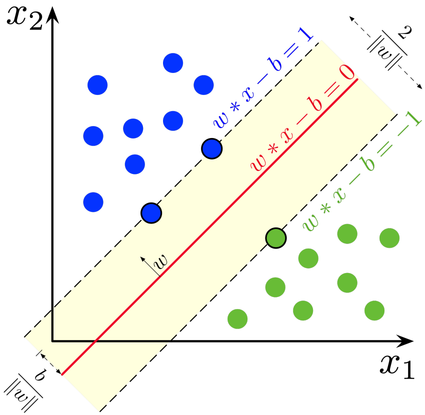
```
W = array([[-0.63895478, -0.37597786],
           [ 0.62693168, -0.49725914],
           [ 0.0120231 ,  0.873237  ]])
b = array([ 1.64147984, -0.00177224, -1.6397076 ])
```

# A simple problem with three classes: Decision boundaries



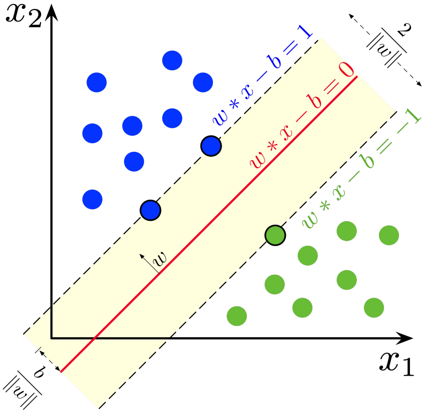

The code will be presented in Jupyter Notebook.

# Support Vector Machine for binary classification




- Support Vector Machines (SVMs) are among the best (and many believe is indeed the best) "off-the-shelf" supervised learning algorithm.
- The mathematics behind this is rather complex, beyond the scope of our live lecture.
- Fancy readers can read my mathematical notes (uploaded later!)

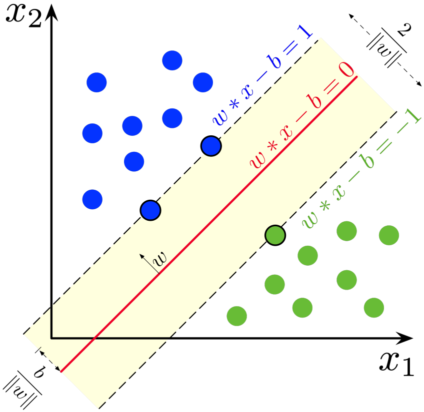
# Support Vector Machine for binary classification




- Maximum-margin hyperplane and margins for an Support Vector Machine (SVM) trained with samples from two classes. Samples on the margin are called the support vectors.
- The labels take the value $-1$ or $+1$:

$$y^{(i)} \in \{-1, 1\}$$

# Support Vector Machine for binary classification




- Maximum-margin hyperplane and margins for an Support Vector Machine (SVM) trained with samples from two classes. Samples on the margin are called the support vectors.
- The labels take the value $-1$ or $+1$:

$$y^{(i)} \in \{-1, 1\}$$

Optimization problem for the weight $\mathbf{w}$ and the intercept $b$:

$$\begin{aligned} &\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to} \quad y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) - 1 \geq 0 \quad \forall i \in \{1, \dots, m\} \end{aligned}$$

# Support Vector Machine: A bit of mathematics

- Optimization problem for the weight $\mathbf{w}$ and the intercept $b$:

$$\begin{aligned} &\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to} \quad y^{(i)}(\mathbf{w} \bullet \mathbf{x}^{(i)} + b) - 1 \geq 0 \quad \forall i \in \{1, \dots, m\} \end{aligned}$$

- A standard method for solving the optimization problem with inequality constraints is the Lagrange multipliers and the Karush-Kuhn-Tucker (KKT) conditions.
- Minimize the Lagrangian with respect to $\mathbf{w}$, $b$, and $\Lambda = (\lambda_1, \dots, \lambda_m)$:

$$\mathcal{L}(\mathbf{w}, b, \Lambda) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \lambda_i \left[y^{(i)}(\mathbf{w} \bullet \mathbf{x}^{(i)} + b) - 1\right]$$

- Karush-Kuhn-Tucker (KKT) conditions:

$$\begin{aligned} \frac{\partial}{\partial w_i}\mathcal{L}(\mathbf{w}, b, \Lambda) = 0 \quad \forall i = 1, \dots, n, \quad \frac{\partial}{\partial b}\mathcal{L}(\mathbf{w}, b, \Lambda) \quad &= 0 \\ \lambda_i \left[1 - y^{(i)}(\mathbf{w} \bullet \mathbf{x}^{(i)} + b)\right] = 0 \quad &\forall i = 1, \dots m, \\ 1 - y^{(i)}(\mathbf{w} \bullet \mathbf{x}^{(i)} + b) \leq 0 \quad &\forall i = 1, \dots, m, \\ \lambda_i \geq 0 \quad &\forall i = 1, \dots, m \end{aligned}$$

# Support Vector Machine: A bit of mathematics (optional)

If we resolve the equations $\partial_{w_i}\mathcal{L} = 0$ and $\partial_b \mathcal{L} = 0$ in the above Karush-Kuhn-Tucker conditions, the following result is obtained

$$\mathbf{w} = \sum_{i=1}^{m} \lambda_i y^{(i)} \mathbf{x}^{(i)}, \quad \sum_{i=1}^{m} \lambda_i y^{(i)} = 0.$$

We can calculate

$$\mathcal{L}(\mathbf{w}, b, \Lambda) = \sum_{i=1}^{m} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \lambda_i \lambda_j (\mathbf{x}^{(i)} \bullet \mathbf{x}^{(j)})$$

Together with the constraints $\lambda_i \leq 0$, we can solve the equivalent optimization problem

$$\max_{\lambda_i} \quad W(\Lambda) = \sum_{i=1}^{m} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \lambda_i \lambda_j \, \mathbf{x}^{(i)} \bullet \mathbf{x}^{(j)}$$

$$\text{such that} \quad \lambda_i \leq 0 \quad \forall i = 1, \dots, m,$$

$$\sum_{i=1}^{m} \lambda_i y^{(i)} = 0$$

## Kernels for Support Vector Machine

- Of course, we want to have nonlinear decision boundary just like how we use logistic regression.
- We can replace the dot product $\mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}$ by the dot product of two new feature transformation $\Phi(\mathbf{x}^{(i)}) \cdot \Phi(\mathbf{x}^{(j)})$ in the above optimization problem.
  ❑ For example, we the feature transformation of the polynomial type (problem with two features)
$$\Phi(\mathbf{x}) = \begin{bmatrix} x_1 & x_2 & x_1^2 & x_2^2 & x_1 x_2 \end{bmatrix}^T$$
- More generally, we can even replace dot product $\mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}$ by a kernel $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ measuring how similar $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$.
  ❑ For instance, we can measure how similar two data points $\mathbf{x}$ and $\mathbf{z}$ by a Gaussian kernel ($\sigma$ is chosen by the user)
$$K(\mathbf{x}, \mathbf{z}) = \exp\left( -\frac{\|\mathbf{x} - \mathbf{z}\|}{2\sigma^2} \right).$$

# 1.4. Support Vector Machines

**Support vector machines (SVMs)** are a set of supervised learning methods used for classification, regression and outliers detection.

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

Link to official documentation. Click on me!