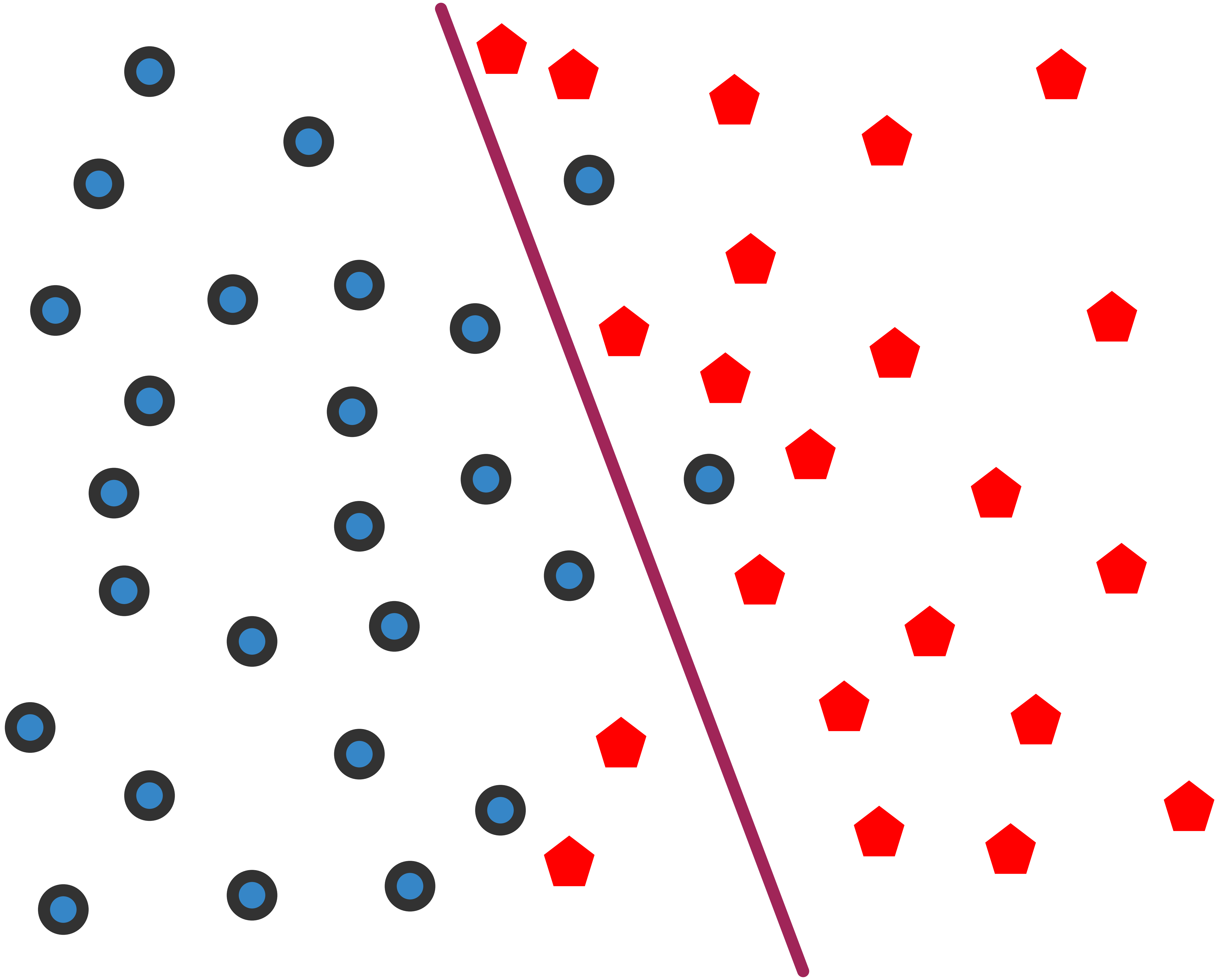
Logistic regression



In this video you will:

- get acquainted with problem of credit risk assessment
- learn how train logistic regression on big data
- learn how to evaluate its performance


£50
20

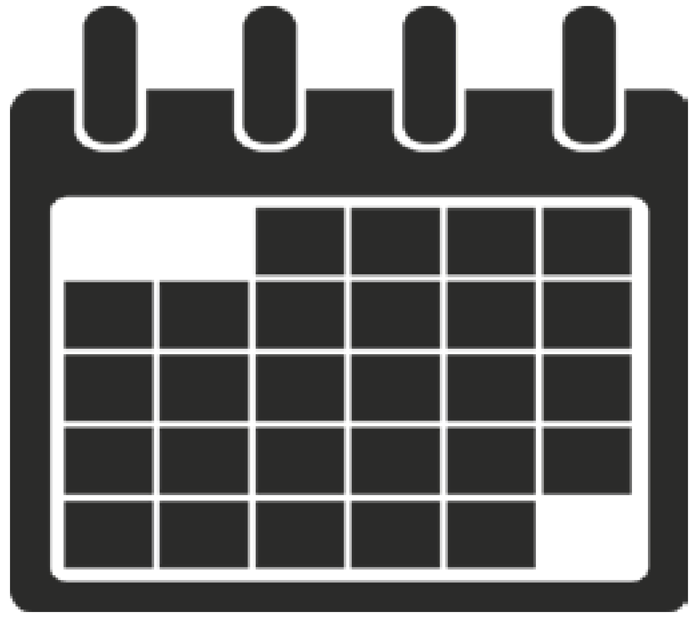
```
data = spark_session.read.csv(
    "/user/pmezentsev/default_of_credit_card_clients",
    header=True)
```

```
data.printSchema()
```

```
root
 |-- ID: string (nullable = true)
 |-- LIMIT_BAL: string (nullable = true)
 |-- SEX: string (nullable = true)
 |-- EDUCATION: string (nullable = true)
 |-- MARRIAGE: string (nullable = true)
 |-- AGE: string (nullable = true)
 |-- PAY_0: string (nullable = true)
 |-- PAY_2: string (nullable = true)
 |-- PAY_3: string (nullable = true)
 |-- PAY_4: string (nullable = true)
 |-- PAY_5: string (nullable = true)
 |-- PAY_6: string (nullable = true)
 |-- BILL_AMT1: string (nullable = true)
 |-- BILL_AMT2: string (nullable = true)
 |-- BILL_AMT3: string (nullable = true)
 |-- BILL_AMT4: string (nullable = true)
 |-- BILL_AMT5: string (nullable = true)
 |-- BILL_AMT6: string (nullable = true)
 |-- PAY_AMT1: string (nullable = true)
 |-- PAY_AMT2: string (nullable = true)
 |-- PAY_AMT3: string (nullable = true)
 |-- PAY_AMT4: string (nullable = true)
 |-- PAY_AMT5: string (nullable = true)
 |-- PAY_AMT6: string (nullable = true)
 |-- default payment next month: string (nullable = true)
```

ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE
1	20000	2	2	1	24
2	120000	2	2	2	26
3	90000	2	2	2	34
4	50000	2	2	1	37
5	50000	1	2	1	57

Sept.	Aug.	July	June	May	Apr.
PAY_0	**PAY_2**	**PAY_3**	**PAY_4**	**PAY_5**	**PAY_6**
2	2	-1	-1	-2	-2
-1	2	0	0	0	2
0	0	0	0	0	0
0	0	0	0	0	0
-1	0	-1	0	0	0



Sept.	Aug.	July	June	May	Apr.
BILL_AMT1	**BILL_AMT2**	**BILL_AMT3**	**BILL_AMT4**	**BILL_AMT5**	**BILL_AMT6**
3913	3102	689	0	0	0
2682	1725	2682	3272	3455	3261
29239	14027	13559	14331	14948	15549
46990	48233	49291	28314	28959	29547
8617	5670	35835	20940	19146	19131

Sept.	Aug.	July	June	May	Apr.
PAY_AMT1	**PAY_AMT2**	**PAY_AMT3**	**PAY_AMT4**	**PAY_AMT5**	**PAY_AMT6**
0	689	0	0	0	0
0	1000	1000	1000	0	2000
1518	1500	1000	1000	1000	5000
2000	2019	1200	1100	1059	1000
2000	34481	10000	9000	689	679

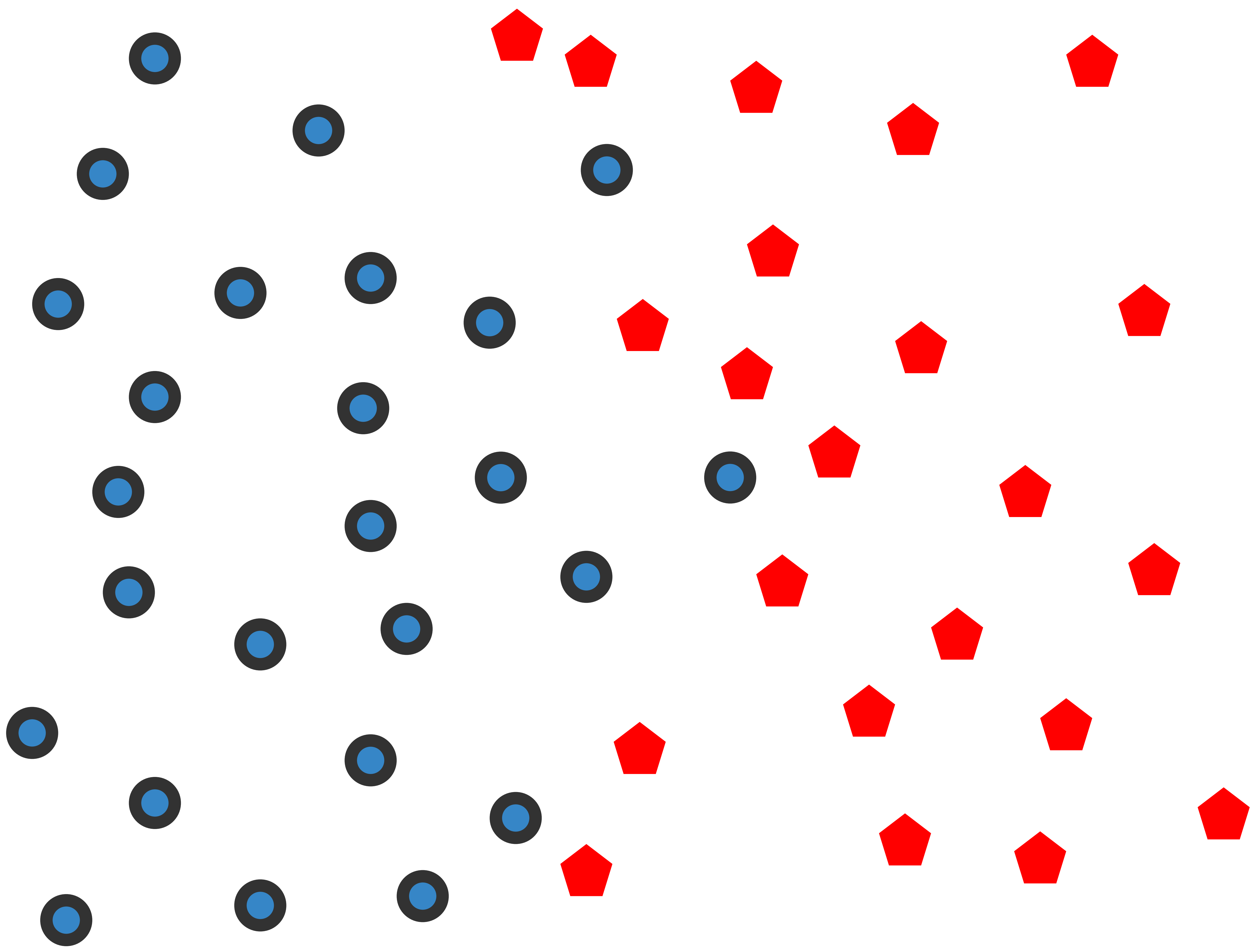
default payment next month
1
1
0
0
0

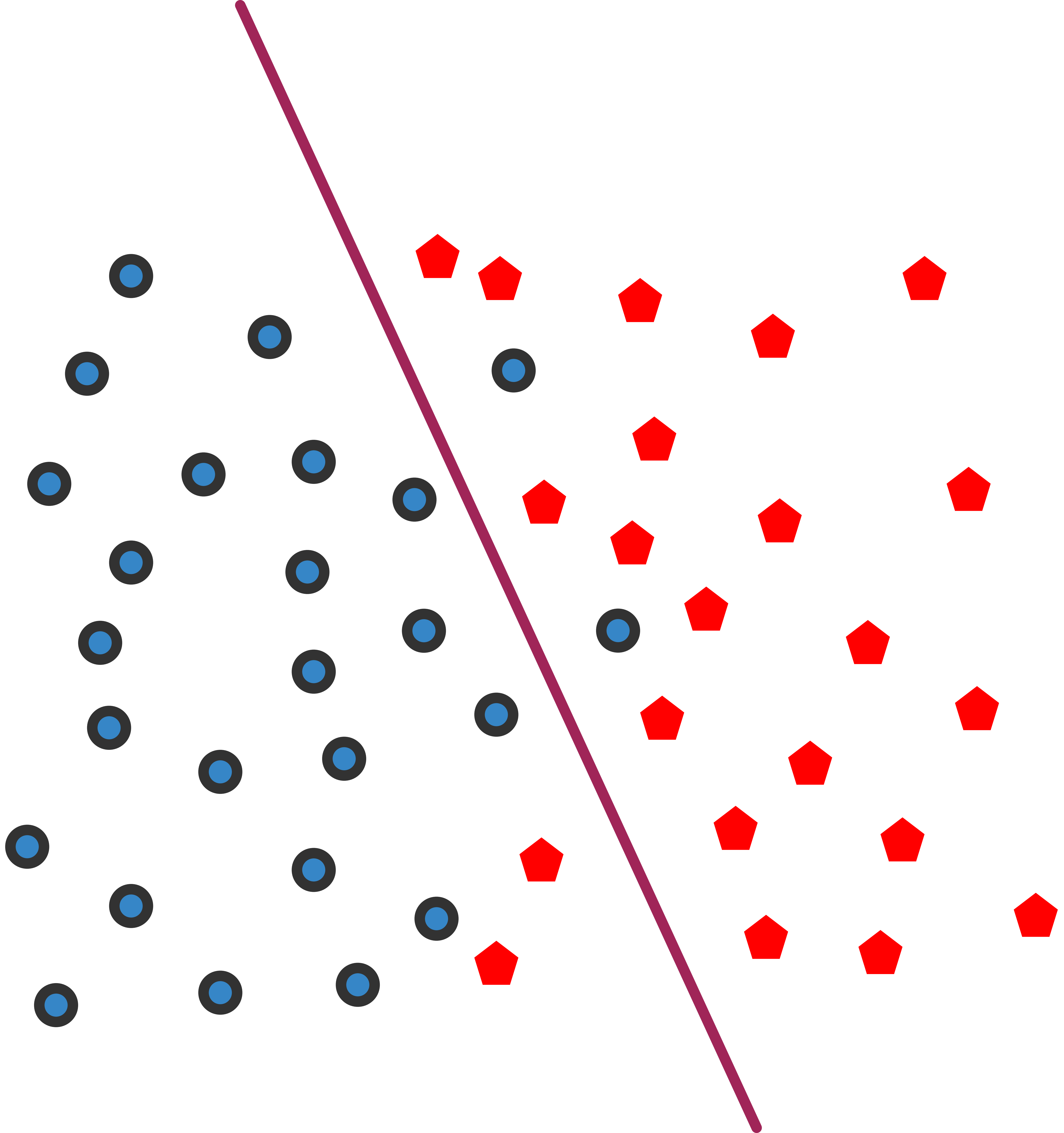
1/0

1/0

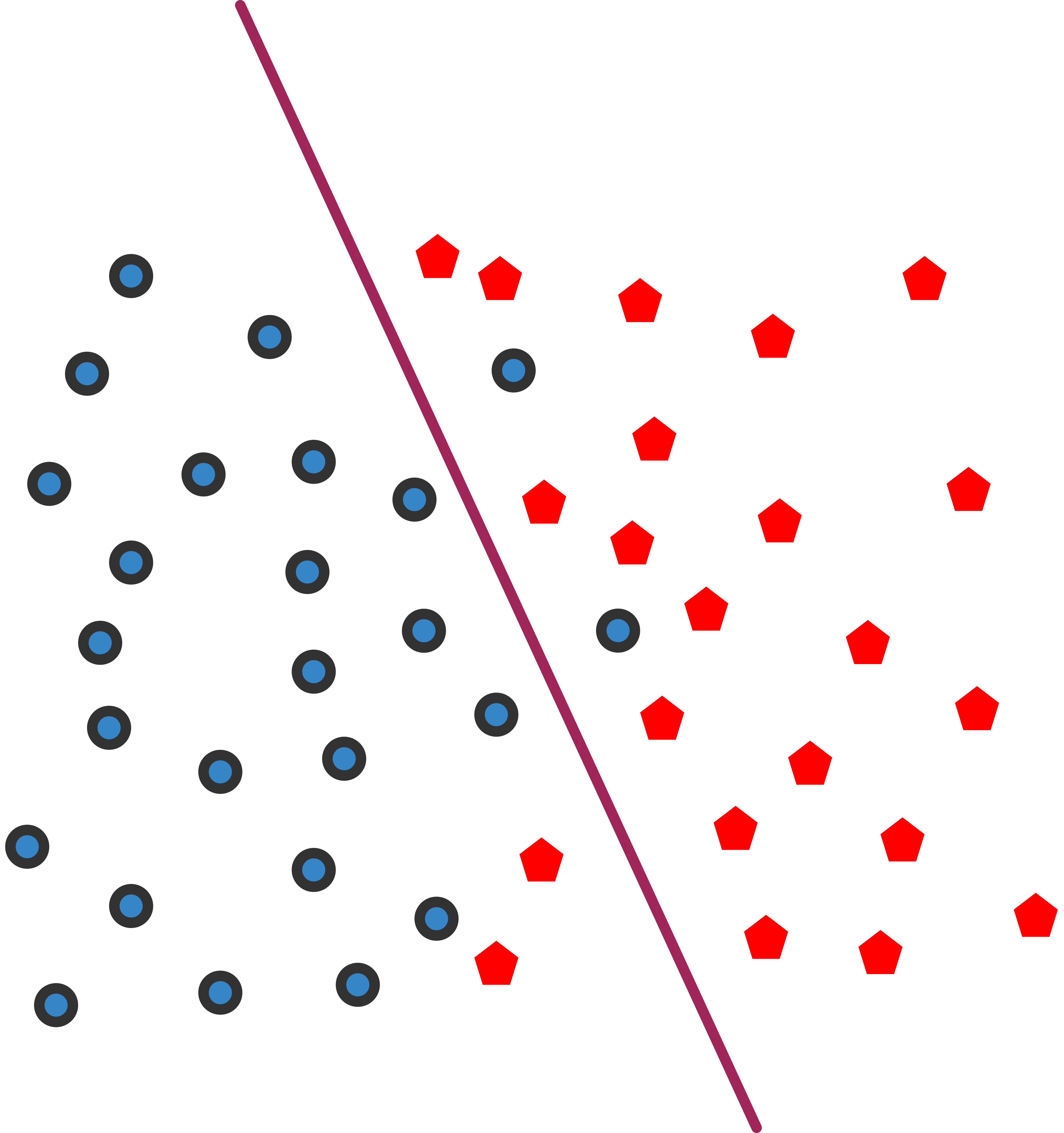
Probability

Logistic regression



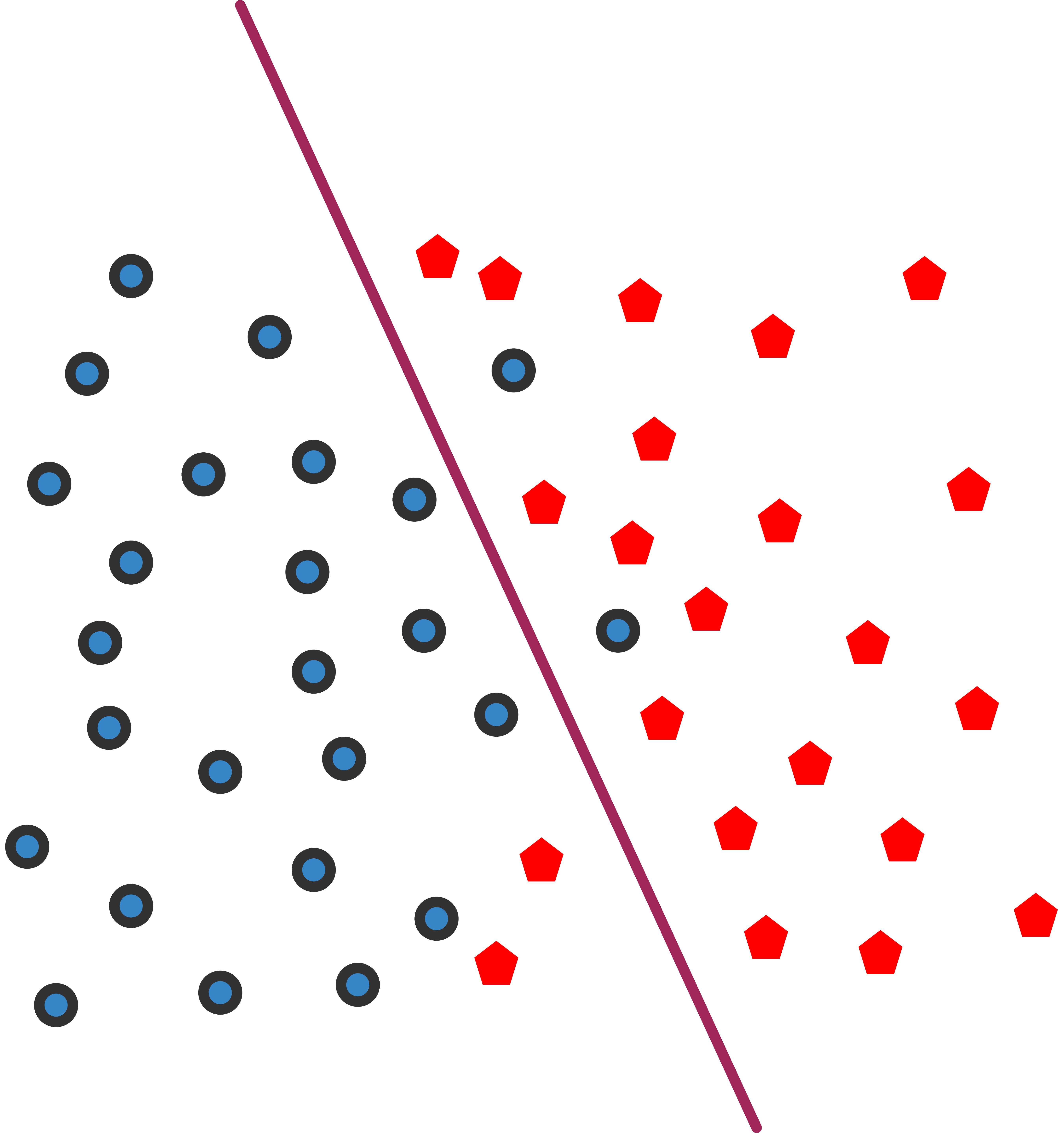


$$x = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}$$



$$x = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}$$

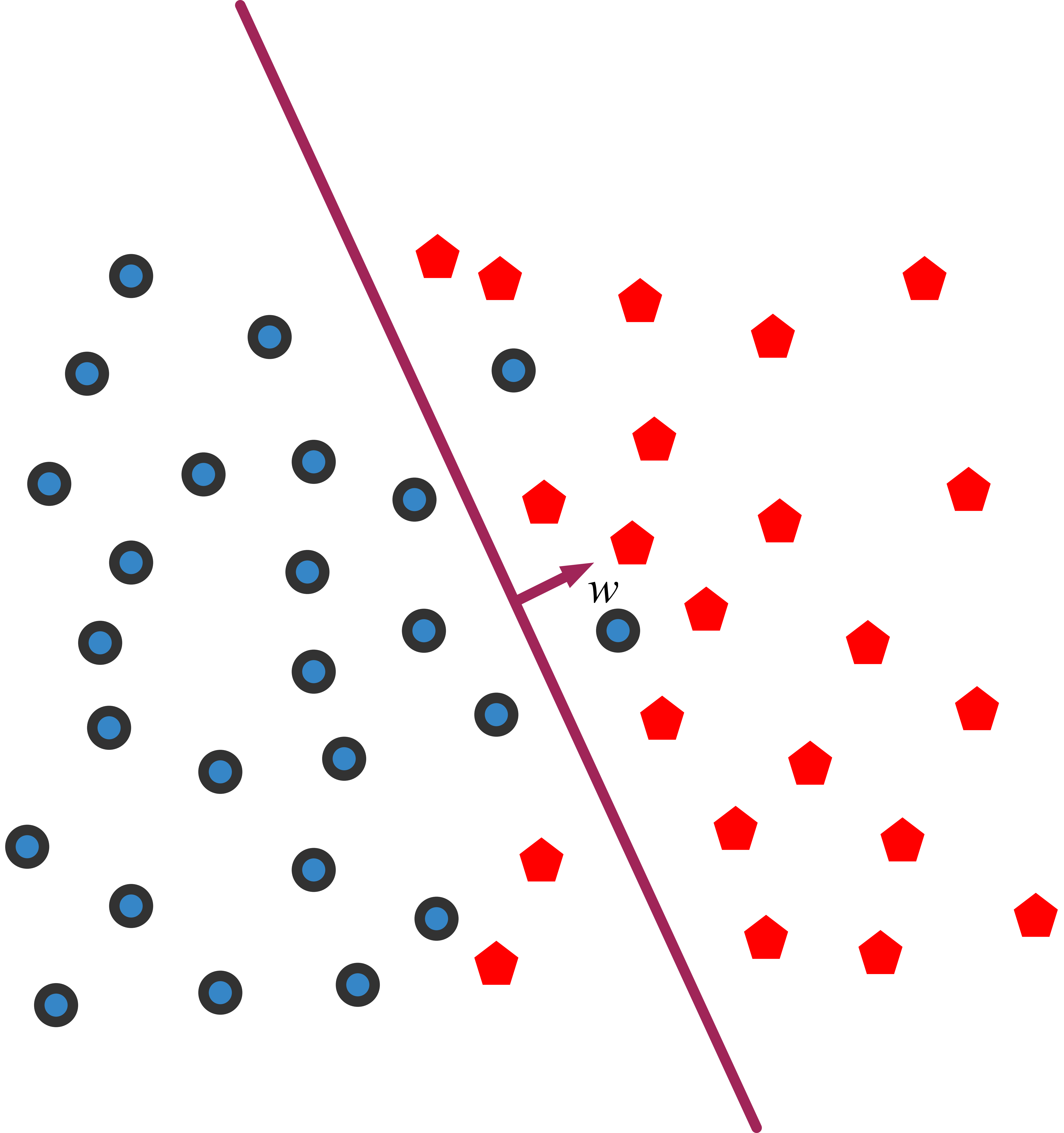
$$y \in (0,1)$$



$$x = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix}$$

$$y \in (0,1)$$

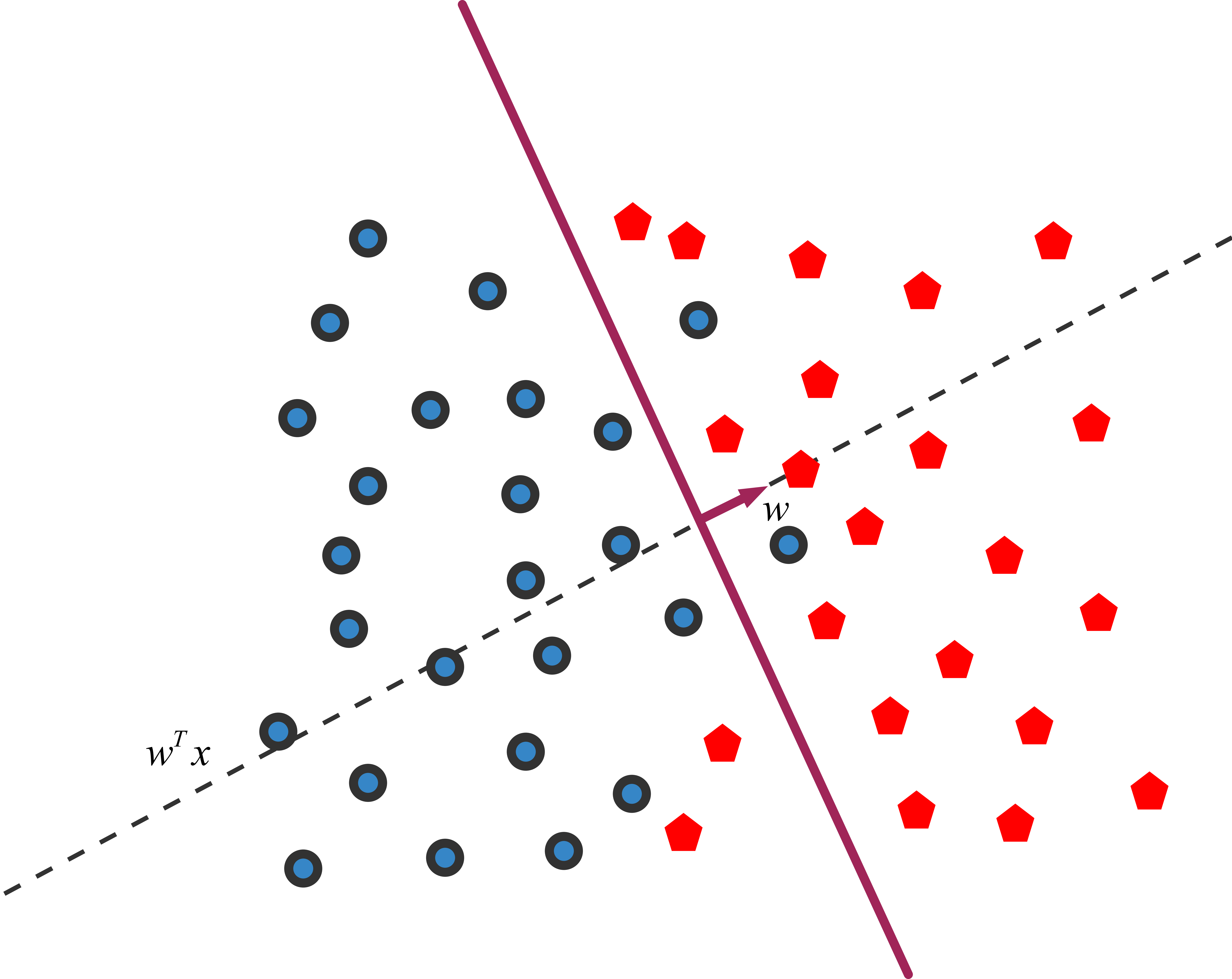
$$w = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \ldots \\ w_n \end{pmatrix}$$



$$x = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}$$

$$y \in (0,1)$$

$$w = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_n \end{pmatrix}$$

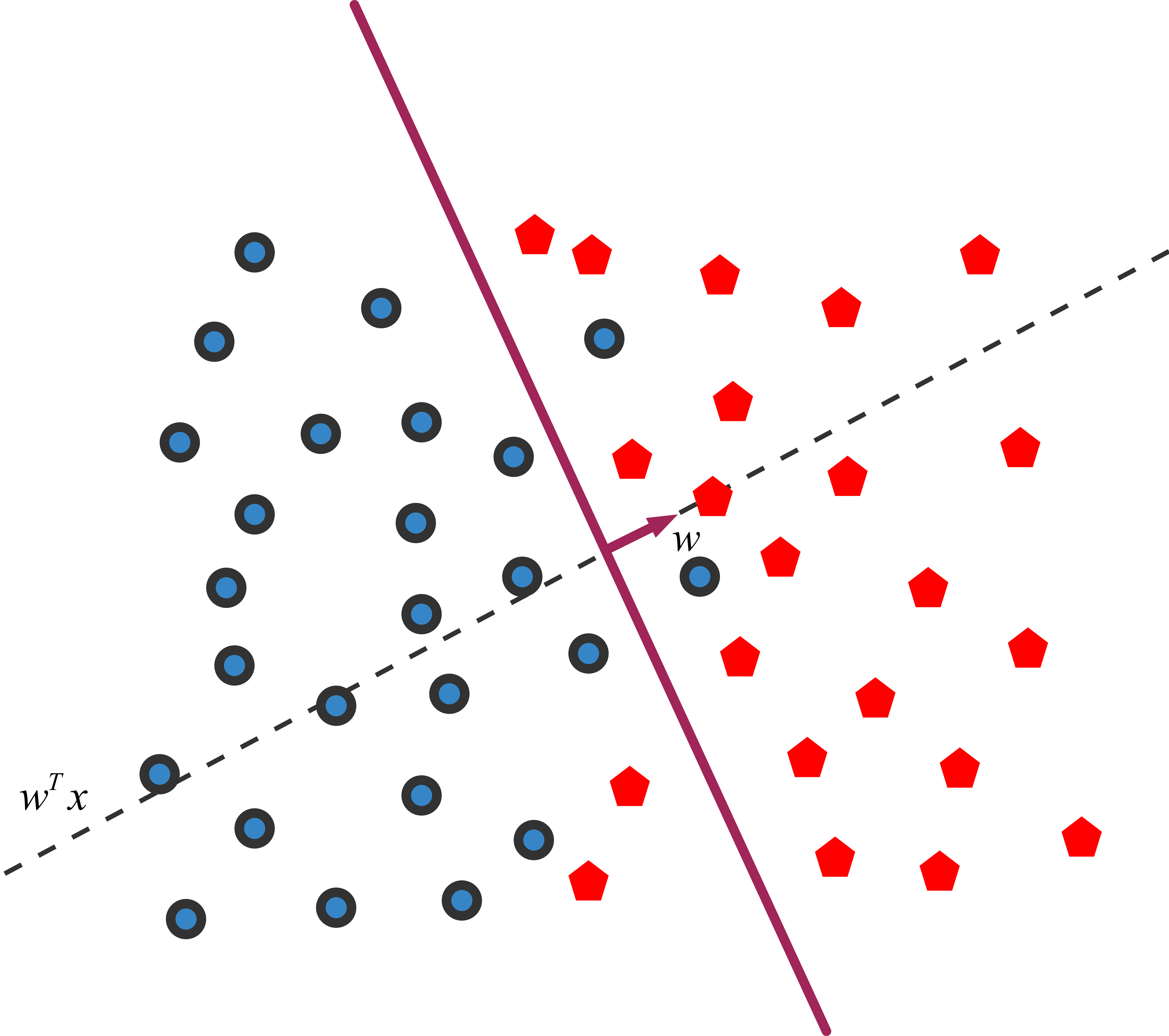


$$x = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}$$

$$y \in (0,1)$$

$$w = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_n \end{pmatrix}$$

$$\hat{y} = \frac{1}{1+e^{-w^T x}}$$

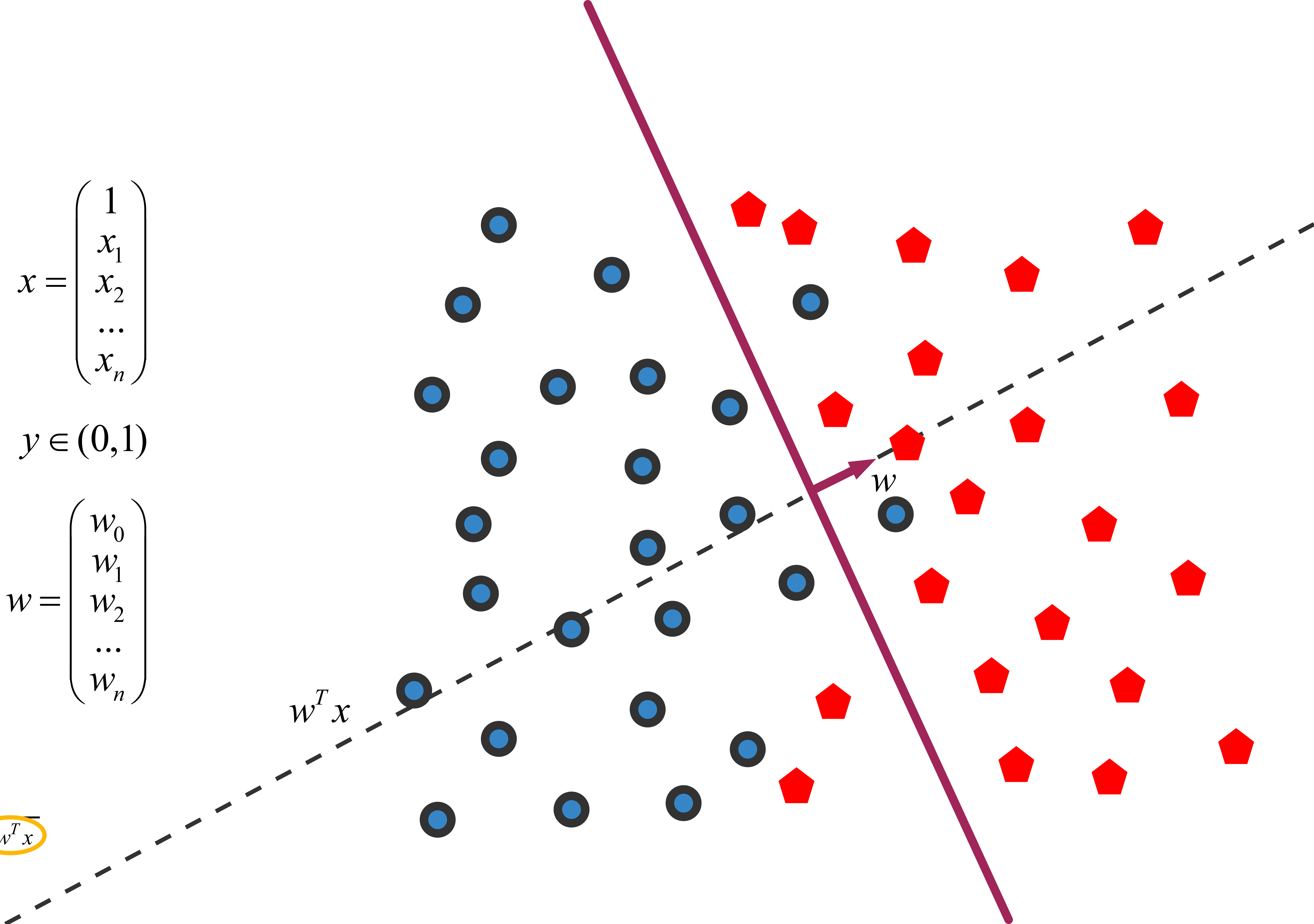


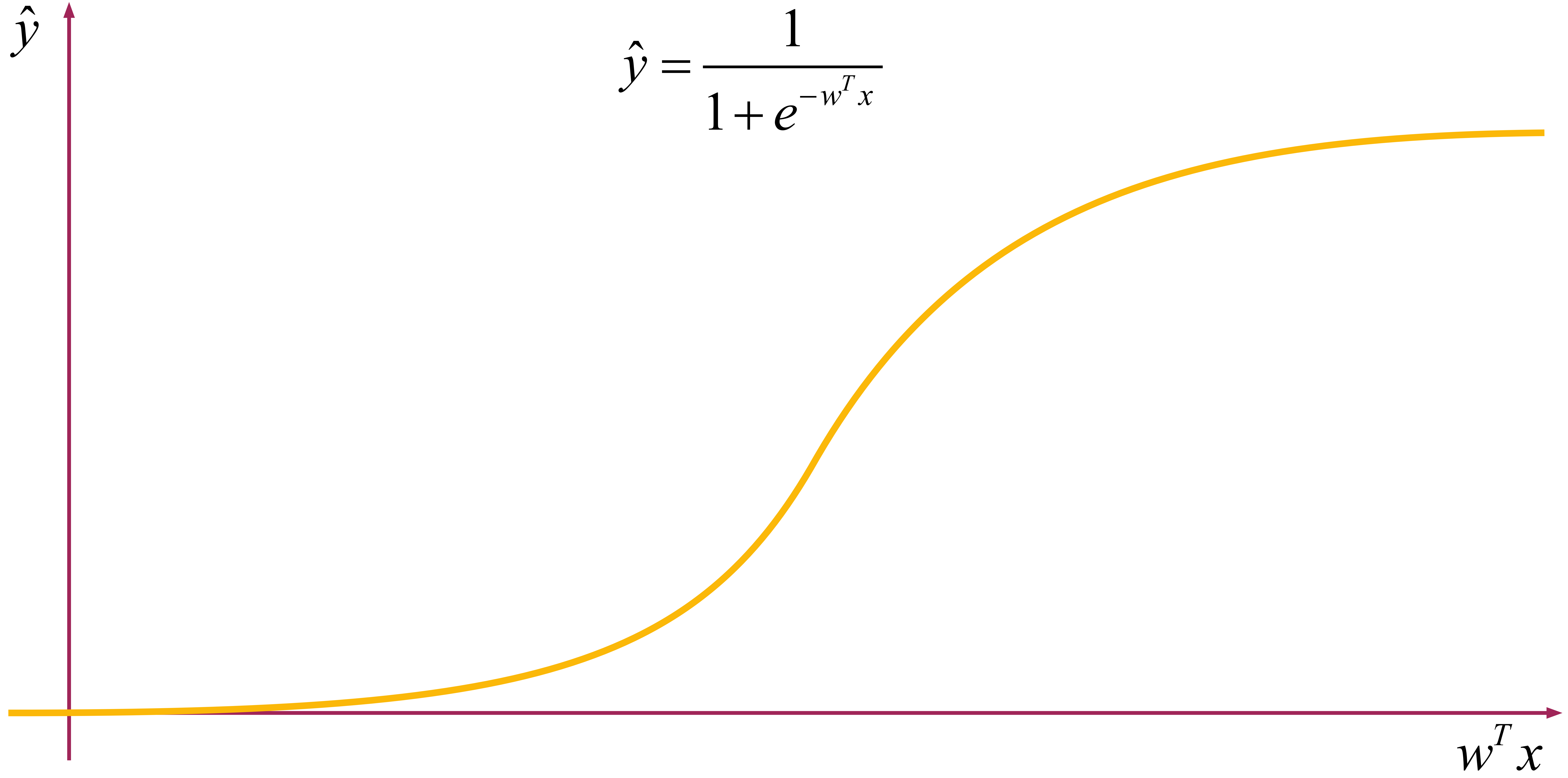
$$x = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \cdots \\ x_n \end{pmatrix}$$

$$y \in (0,1)$$

$$w = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \cdots \\ w_n \end{pmatrix}$$

$$\hat{y} = \frac{1}{1 + e^{-w^T x}}$$




$\hat{y}$
$$\hat{y} = \frac{1}{1+e^{-w^T x}}$$
$w^T x$

L-BFGS

```
from pyspark.ml.feature import SQLTransformer

my_sql_transformer = SQLTransformer(
    statement="""
    SELECT
       cast(LIMIT_BAL         as int),
       cast(SEX         as int),
       cast(EDUCATION         as int),
       cast(MARRIAGE         as int),
       cast(AGE         as int),
       cast(PAY_0         as int),
       cast(PAY_2         as int),
       cast(PAY_3         as int),
       cast(PAY_4         as int),
       cast(PAY_5         as int),
       cast(PAY_6         as int),
       cast(BILL_AMT1         as int),
       cast(BILL_AMT2         as int),
       cast(BILL_AMT3         as int),
       cast(BILL_AMT4         as int),
       cast(BILL_AMT5         as int),
       cast(BILL_AMT6         as int),
       cast(PAY_AMT1         as int),
       cast(PAY_AMT2         as int),
       cast(PAY_AMT3         as int),
       cast(PAY_AMT4         as int),
       cast(PAY_AMT5         as int),
       cast(PAY_AMT6         as int),
       cast('default payment next month'         as int) as label
    FROM __THIS__
    """)
```

```
assembler = VectorAssembler()\
  .setInputCols([
 'LIMIT_BAL',
 'SEX',
 'EDUCATION',
 'MARRIAGE',
 'AGE',
 'PAY_0', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6',
 'BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6',
 'PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6'
    ])\
  .setOutputCol("features")
```

```
data02 = assembler.transform(data01).select('features', 'label')
```

```
train, test = data02.randomSplit([0.7,0.3], seed=1)
```

```
lr = LogisticRegression()
lr_model = lr.fit(train)
```

```
scored_test = lr_model.transform(test)
scored_train = lr_model.transform(train)
```

```
scored_test.limit(5).toPandas()
```

	features	label	rawPrediction	probability	prediction
0	(10000.0, 2.0, 1.0, 2.0, 23.0, 1.0, -2.0, -2.0...	0	[1.04333251803, -1.04333251803]	[0.739492505681, 0.260507494319]	0.0
1	(20000.0, 1.0, 2.0, 2.0, 23.0, 1.0, -2.0, -2.0...	0	[1.04244635215, -1.04244635215]	[0.739321755498, 0.260678244502]	0.0
2	(20000.0, 1.0, 3.0, 2.0, 40.0, 1.0, -2.0, -2.0...	0	[1.0287134211, -1.0287134211]	[0.736666389661, 0.263333610339]	0.0
3	(20000.0, 2.0, 2.0, 1.0, 26.0, 1.0, -2.0, -2.0...	1	[0.976449927859, -0.976449927859]	[0.726403236781, 0.273596763219]	0.0
4	(20000.0, 2.0, 2.0, 1.0, 48.0, 1.0, -2.0, -2.0...	1	[0.840721980366, -0.840721980366]	[0.698617251621, 0.301382748379]	0.0

```
scored_test.limit(5).toPandas()
```

	features	label	rawPrediction	probability	prediction
0	(10000.0, 2.0, 1.0, 2.0, 23.0, 1.0, -2.0, -2.0...	0	[1.04333251803, -1.04333251803]	[0.739492505681, 0.260507494319]	0.0
1	(20000.0, 1.0, 2.0, 2.0, 23.0, 1.0, -2.0, -2.0...	0	[1.04244635215, -1.04244635215]	[0.739321755498, 0.260678244502]	0.0
2	(20000.0, 1.0, 3.0, 2.0, 40.0, 1.0, -2.0, -2.0...	0	[1.0287134211, -1.0287134211]	[0.736666389661, 0.263333610339]	0.0
3	(20000.0, 2.0, 2.0, 1.0, 26.0, 1.0, -2.0, -2.0...	1	[0.976449927859, -0.976449927859]	[0.726403236781, 0.273596763219]	0.0
4	(20000.0, 2.0, 2.0, 1.0, 48.0, 1.0, -2.0, -2.0...	1	[0.840721980366, -0.840721980366]	[0.698617251621, 0.301382748379]	0.0

$$w^T x$$

```
scored_test.limit(5).toPandas()
```

	features	label	rawPrediction	probability	prediction
0	(10000.0, 2.0, 1.0, 2.0, 23.0, 1.0, -2.0, -2.0...	0	[1.04333251803, -1.04333251803]	[0.739492505681, 0.260507494319]	0.0
1	(20000.0, 1.0, 2.0, 2.0, 23.0, 1.0, -2.0, -2.0...	0	[1.04244635215, -1.04244635215]	[0.739321755498, 0.260678244502]	0.0
2	(20000.0, 1.0, 3.0, 2.0, 40.0, 1.0, -2.0, -2.0...	0	[1.0287134211, -1.0287134211]	[0.736666389661, 0.263333610339]	0.0
3	(20000.0, 2.0, 2.0, 1.0, 26.0, 1.0, -2.0, -2.0...	1	[0.976449927859, -0.976449927859]	[0.726403236781, 0.273596763219]	0.0
4	(20000.0, 2.0, 2.0, 1.0, 48.0, 1.0, -2.0, -2.0...	1	[0.840721980366, -0.840721980366]	[0.698617251621, 0.301382748379]	0.0

$w^T x$

$$\hat{y} = \frac{1}{1+e^{-w^T x}}$$

```
scored_test.limit(5).toPandas()
```

	features	label	rawPrediction	probability	prediction
0	(10000.0, 2.0, 1.0, 2.0, 23.0, 1.0, -2.0, -2.0...	0	[1.04333251803, -1.04333251803]	[0.739492505681, 0.260507494319]	0.0
1	(20000.0, 1.0, 2.0, 2.0, 23.0, 1.0, -2.0, -2.0...	0	[1.04244635215, -1.04244635215]	[0.739321755498, 0.260678244502]	0.0
2	(20000.0, 1.0, 3.0, 2.0, 40.0, 1.0, -2.0, -2.0...	0	[1.0287134211, -1.0287134211]	[0.736666389661, 0.263333610339]	0.0
3	(20000.0, 2.0, 2.0, 1.0, 26.0, 1.0, -2.0, -2.0...	1	[0.976449927859, -0.976449927859]	[0.726403236781, 0.273596763219]	0.0
4	(20000.0, 2.0, 2.0, 1.0, 48.0, 1.0, -2.0, -2.0...	1	[0.840721980366, -0.840721980366]	[0.698617251621, 0.301382748379]	0.0

$w^T x$

$$\hat{y} = \frac{1}{1+e^{-w^T x}}$$

$\hat{y} > 0{,}5$

```python
from pyspark.ml.evaluation import BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator()
```

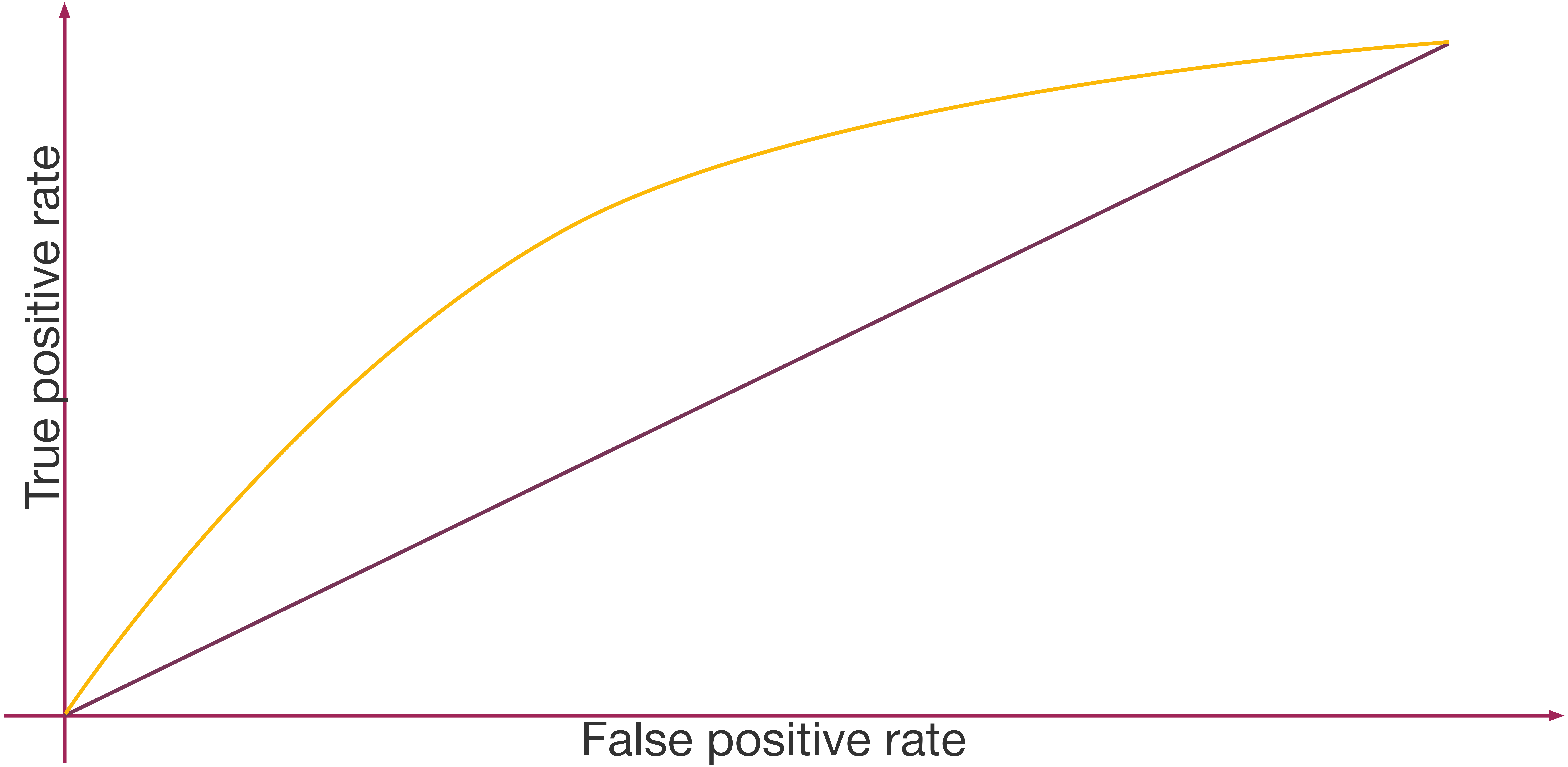
```
evaluator.evaluate(scored_train, {evaluator.metricName: 'areaUnderROC'})
```

```
0.722522648109
```

```
evaluator.evaluate(scored_test, {evaluator.metricName: 'areaUnderROC'})
```

```
0.728012080672
```

ROC



In this video you:

- have got acquainted with problem of credit risk assessment
- have learned how to train logistic regression on big data
- have learned how to evaluate its performance