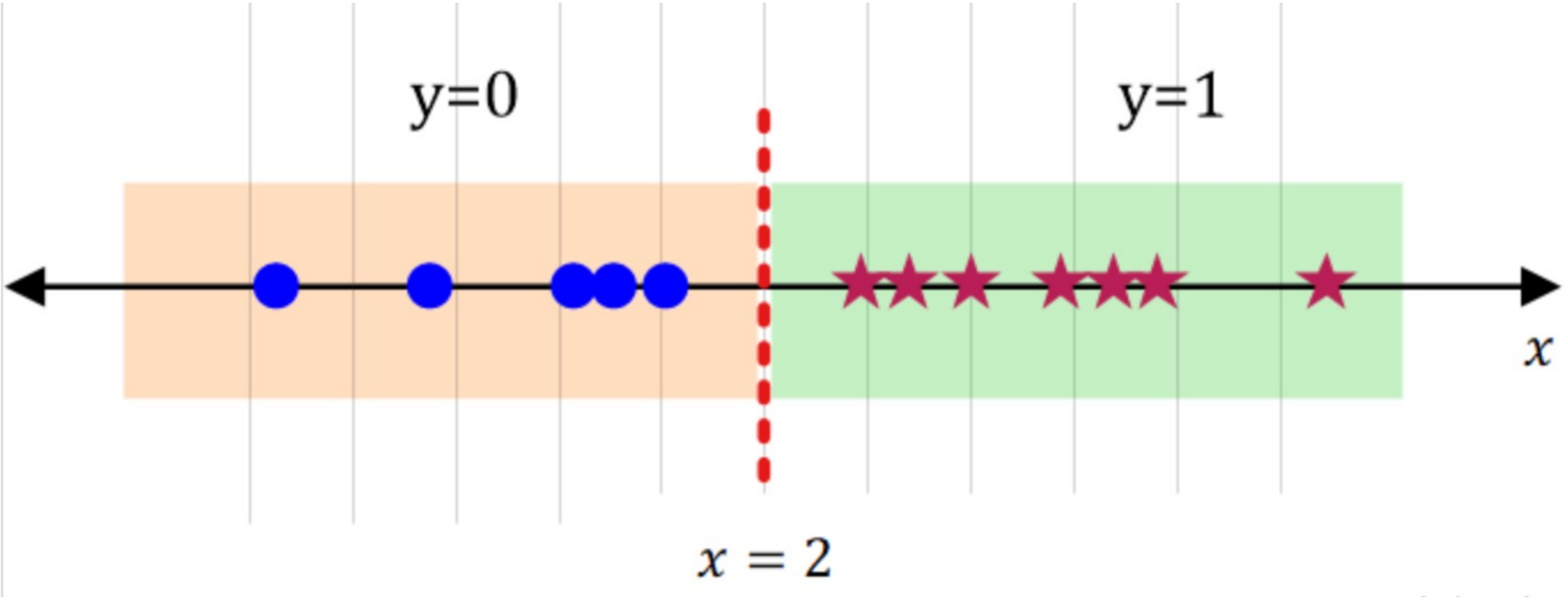
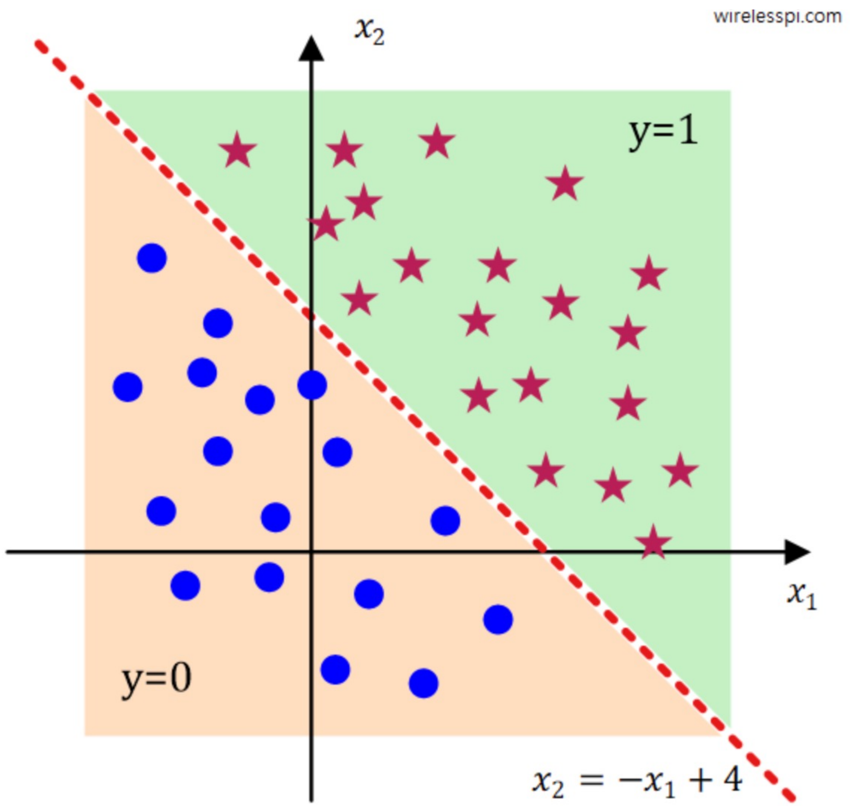
# Logistic Regression

# Simple logistic regression



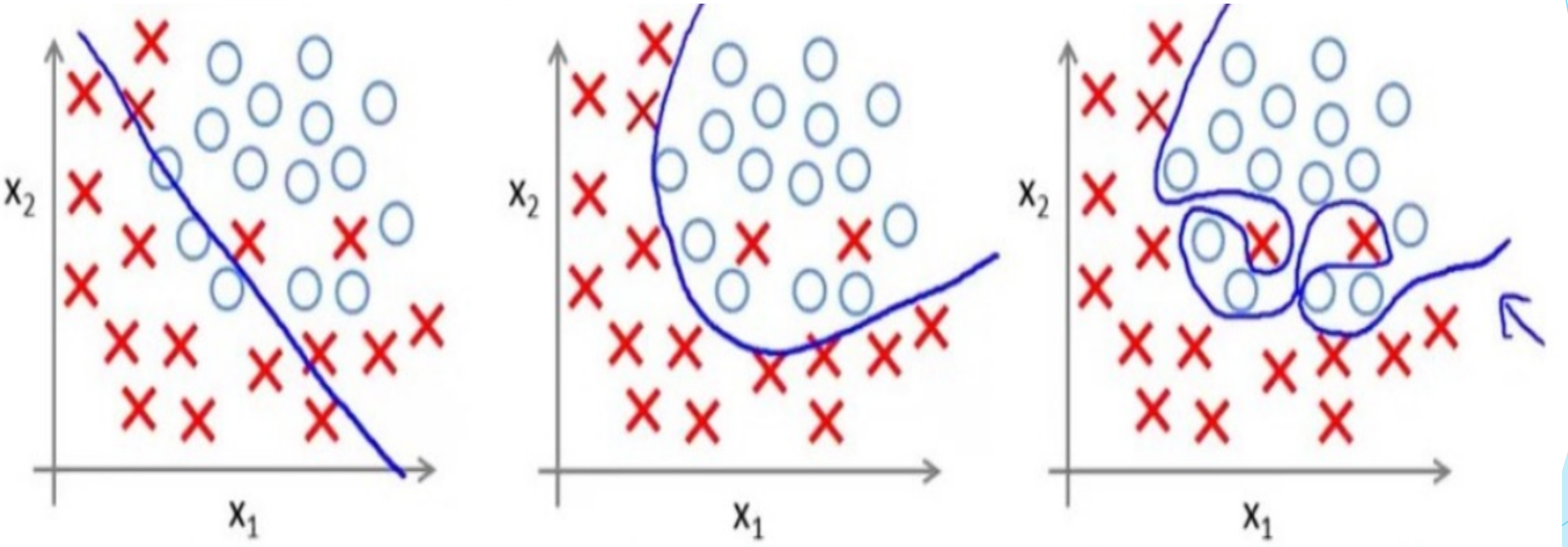

$$z = \theta_0 + \theta_1 \cdot x$$

# Multivariate Logistic Regression




$$z = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2$$

# Which one is a better classifier?



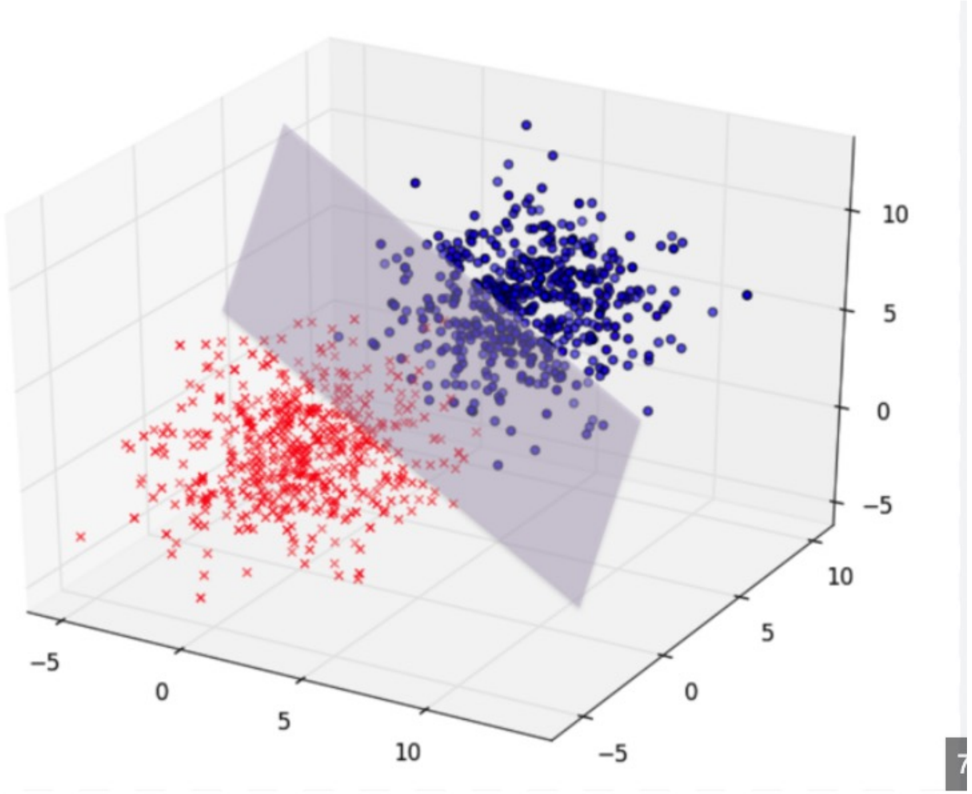

P1

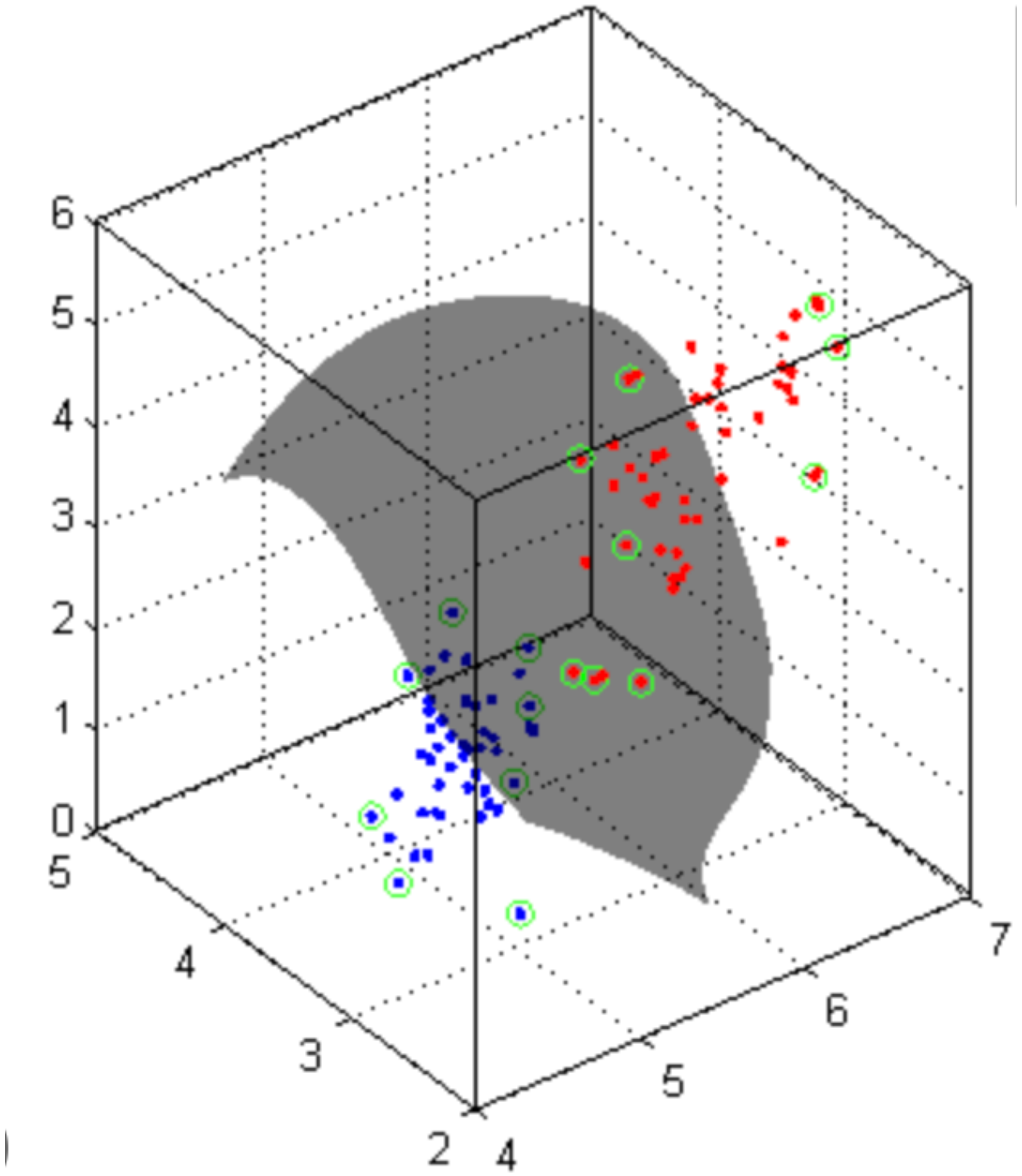
P2

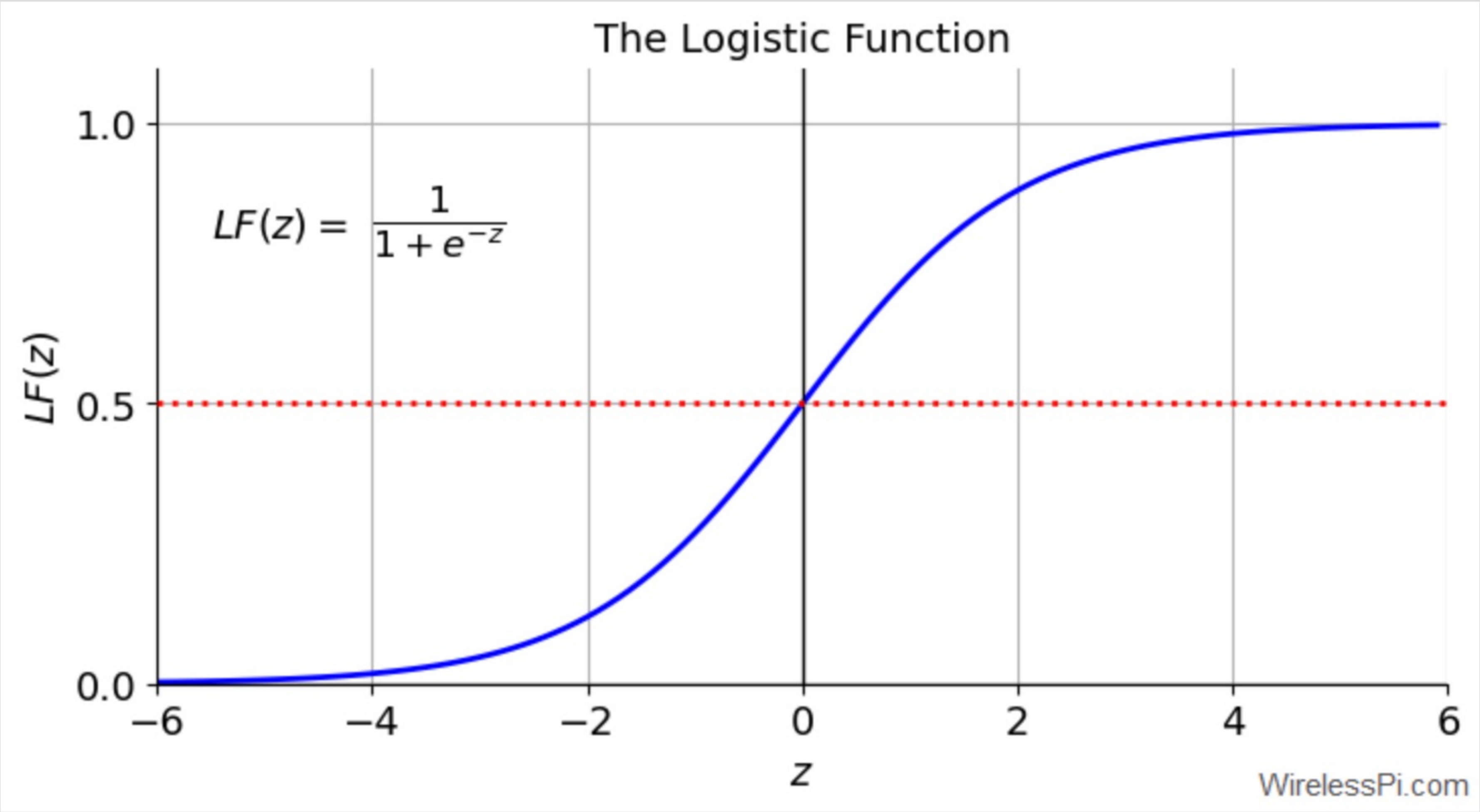
P3

# Multivariate Logistic Regression

## (3 dimensional example)




6
5
4
3
2
1
0
5
4
3
2
4
5
6
7


The Logistic Function
$LF(z) = \frac{1}{1+e^{-z}}$
LF(z)
1.0
0.5
0.0
−6
−4
−2
0
2
4
6
z
WirelessPi.com

# Logistic regression steps:

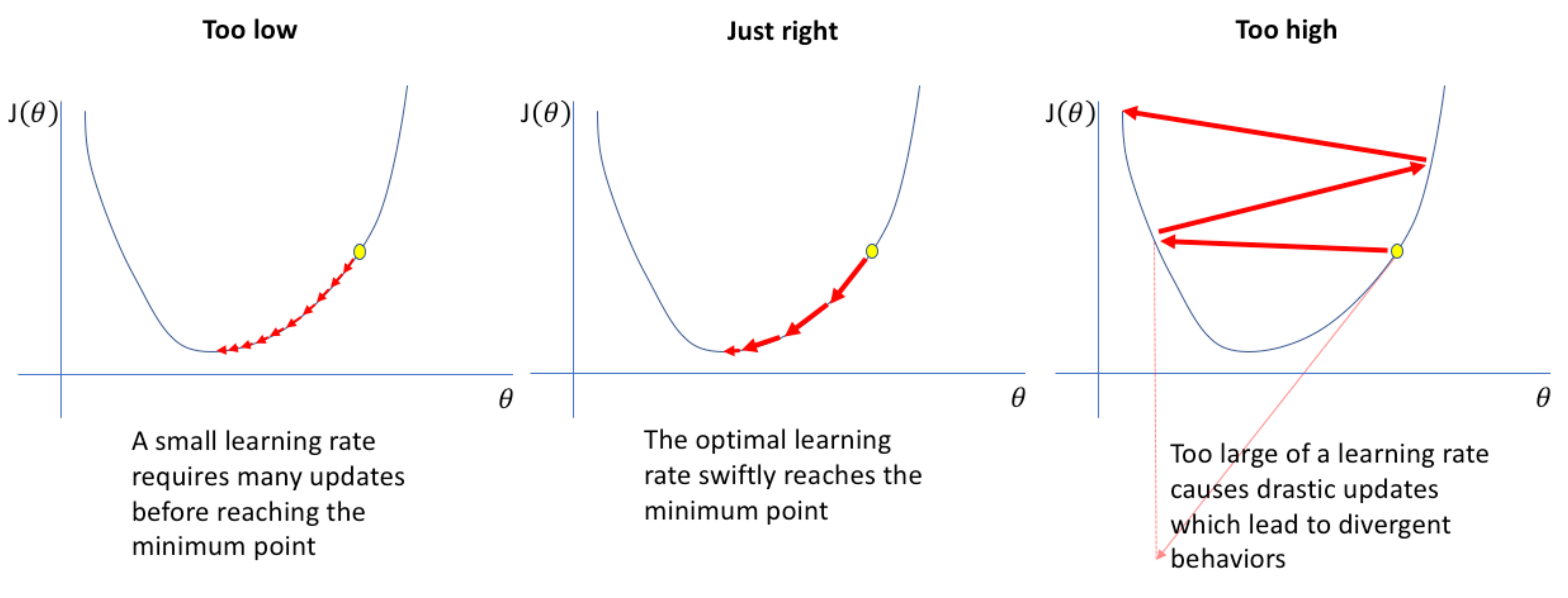
- Select a random $\theta$ as the initial parameter
- Calculate $\hat{y}$= $\sigma\ (\theta^T \mathrm{x})$
- Calculate the difference between $\hat{y}$ and actual y

$$cost(\hat{y},\mathrm{y}) = -\mathrm{y}\ \log\hat{y} - (1-y)\log(1-\hat{y})$$

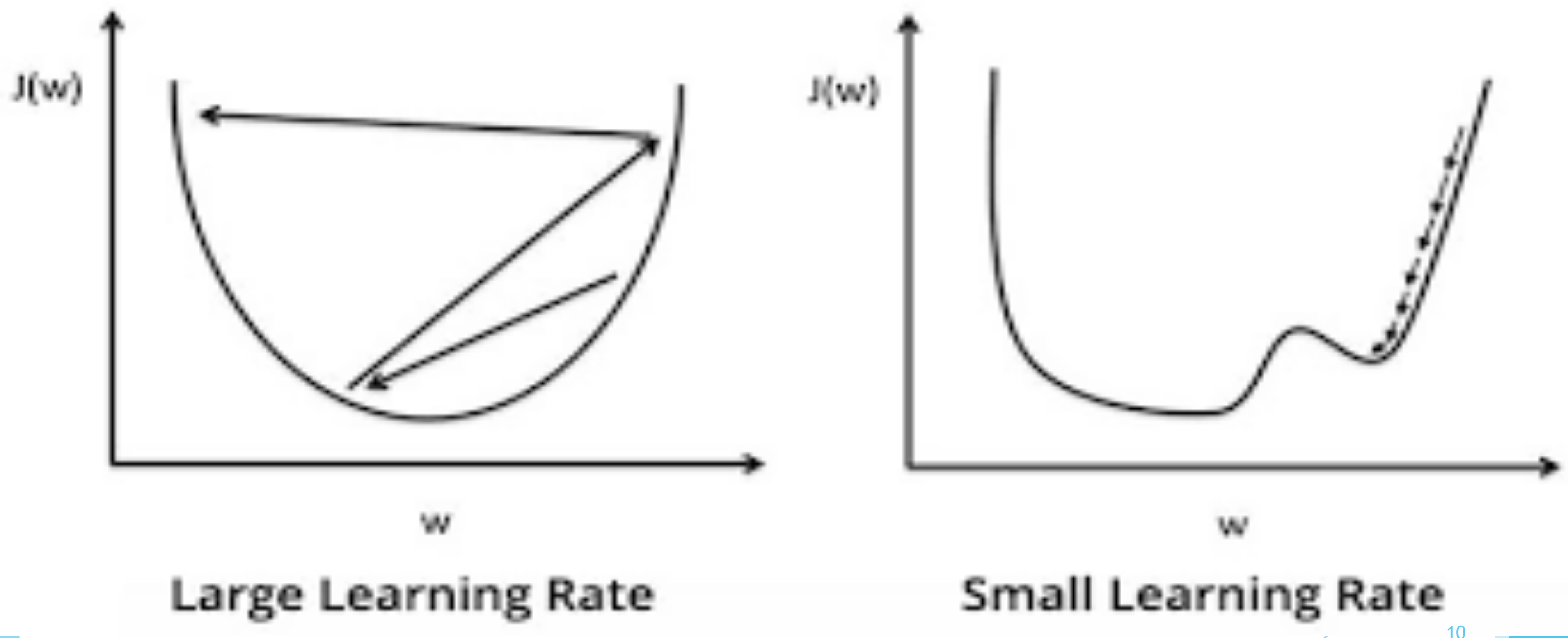
- Calculate cost function for all training samples ($J(\theta)$)
- Update th parameters by gradient descent

$$\theta_j = \ \theta_{j-1} - \ \gamma\ \nabla J(\theta)$$

# Which is the best Learning rate



# Small learning rate



# Evaluating the Logistic Regression Model

- Accuracy
- Precision
- Recall
- F1 score
- Confusion Matrix

# Sklearn implimentation

```
class sklearn.linear_model.LogisticRegression(penalty='l2', *, dual=False,
tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None,
random_state=None, solver='lbfgs', max_iter=100, multi_class='deprecated',
verbose=0, warm_start=False, n_jobs=None, l1_ratio=None)
```

[source]

Logistic Regression (aka logit, MaxEnt) classifier.

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross-entropy loss if the 'multi_class' option is set to 'multinomial'. (Currently the 'multinomial' option is supported only by the 'lbfgs', 'sag', 'saga' and 'newton-cg' solvers.)

This class implements regularized logistic regression using the 'liblinear' library, 'newton-cg', 'sag', 'saga' and 'lbfgs' solvers. **Note that regularization is applied by default**. It can handle both dense and sparse input. Use C-ordered arrays or CSR matrices containing 64-bit floats for optimal performance; any other input format will be converted (and copied).

The 'newton-cg', 'sag', and 'lbfgs' solvers support only L2 regularization with primal formulation, or no regularization. The 'liblinear' solver supports both L1 and L2 regularization, with a dual formulation only for the L2 penalty. The Elastic-Net regularization is only supported by the 'saga' solver.

Read more in the User Guide.

**penalty** : *{'l1', 'l2', 'elasticnet', None}, default='l2'*

Specify the norm of the penalty:

- `None` : no penalty is added;
- `'l2'` : add a L2 penalty term and it is the default choice;
- `'l1'` : add a L1 penalty term;
- `'elasticnet'` : both L1 and L2 penalty terms are added.

| **penalty** | $r(w)$ | |
|---|---|---|
| `None` | $0$ | |
| $\ell_1$ | $\|w\|_1$ | Lasso cost function |
| $\ell_2$ | $\frac{1}{2}\|w\|_2^2 = \frac{1}{2}w^T w$ | Ridge cost function |
| `ElasticNet` | $\frac{1-\rho}{2}w^T w + \rho\|w\|_1$ | |

**solver** : *{'lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', 'sag', 'saga'}, default='lbfgs'*

Algorithm to use in the optimization problem. Default is 'lbfgs'. To choose a solver, you might want to consider the following aspects:

- For small datasets, 'liblinear' is a good choice, whereas 'sag' and 'saga' are faster for large ones;
- For multiclass problems, only 'newton-cg', 'sag', 'saga' and 'lbfgs' handle multinomial loss;
- 'liblinear' and 'newton-cholesky' can only handle binary classification by default. To apply a one-versus-rest scheme for the multiclass setting one can wrapt it with the `OneVsRestClassifier`.
- 'newton-cholesky' is a good choice for `n_samples` >> `n_features`, especially with one-hot encoded categorical features with rare categories. Be aware that the memory usage of this solver has a quadratic dependency on `n_features` because it explicitly computes the Hessian matrix.