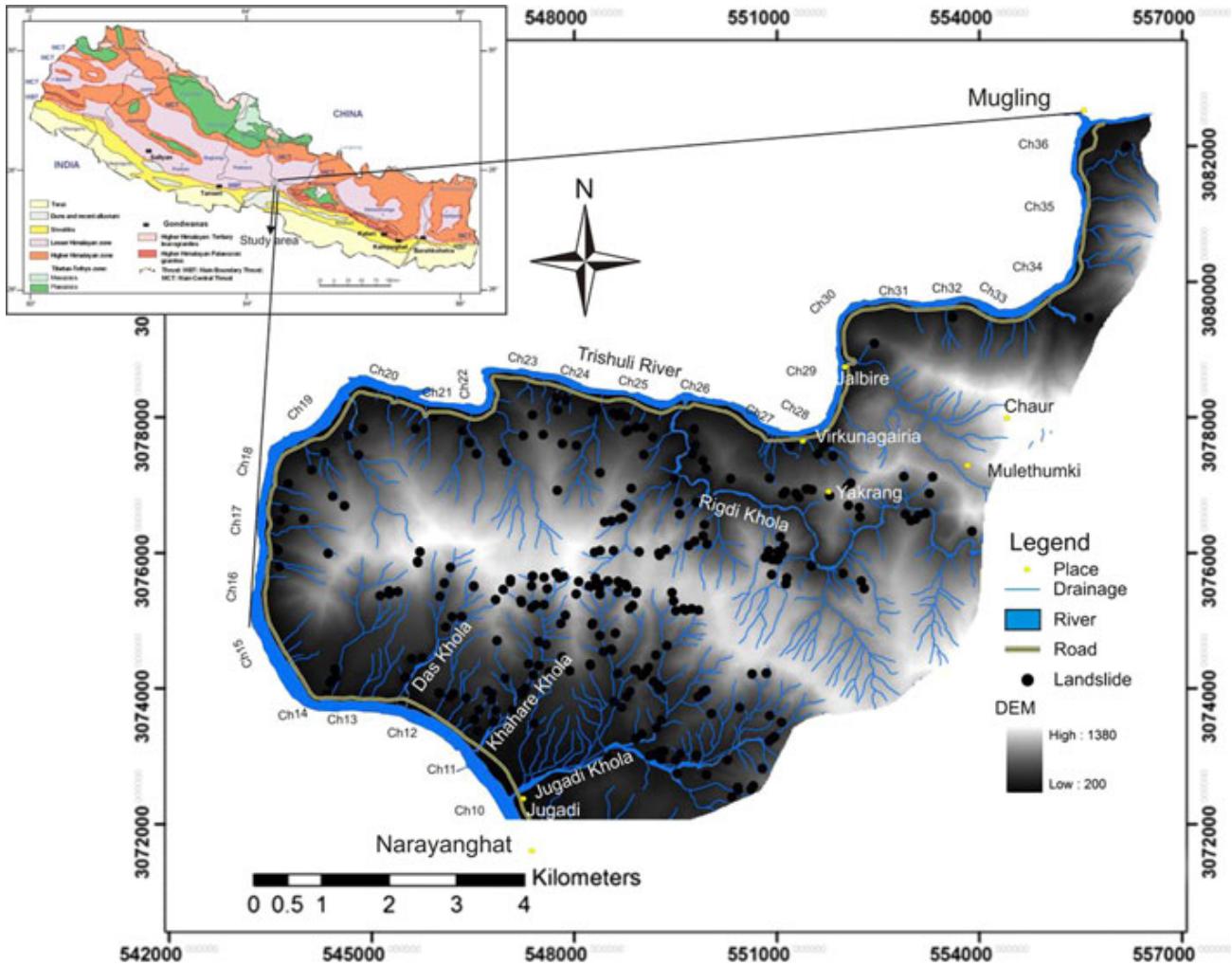


Estrada de 36km que liga Mugling–Narayanghat (Distrito Chitwan, Nepal central)

A susceptibilidade ao deslizamento é a probabilidade de uma ocorrência de deslizamento de terra em uma dada área. É o grau em que um terreno pode ser afetado por movimentos de declive, ou seja, uma estimativa de onde deslizamentos podem ocorrer.



Devkota, K. C., Regmi, A. D., Pourghasemi, H. R., Yoshida, K., Pradhan, B., Ryu, I. C., ... & Althuwaynee, O. F. (2013). Landslide susceptibility mapping using certainty factor, index of entropy and logistic regression models in GIS and their comparison at Mugling–Narayanghat road section in Nepal Himalaya. *Natural hazards*, 65(1), 135-165.

Fatores condicionantes (características):

- *gradiente de inclinação,*
- *aspecto da inclinação,*
- *curvatura do plano,*
- *altitude,*
- *índice de potência da corrente (SPI),*
- *índice de umidade topográfica (TWI),*
- *índice de transporte de sedimentos do fluxo (ITS),*
- *geologia,*
- *uso do solo,*
- *distância das falhas,*
- *distância dos rios e*
- *distância das estradas.*

Variável dependente (y) é uma variável binária representando a presença ou ausência de deslizamentos.

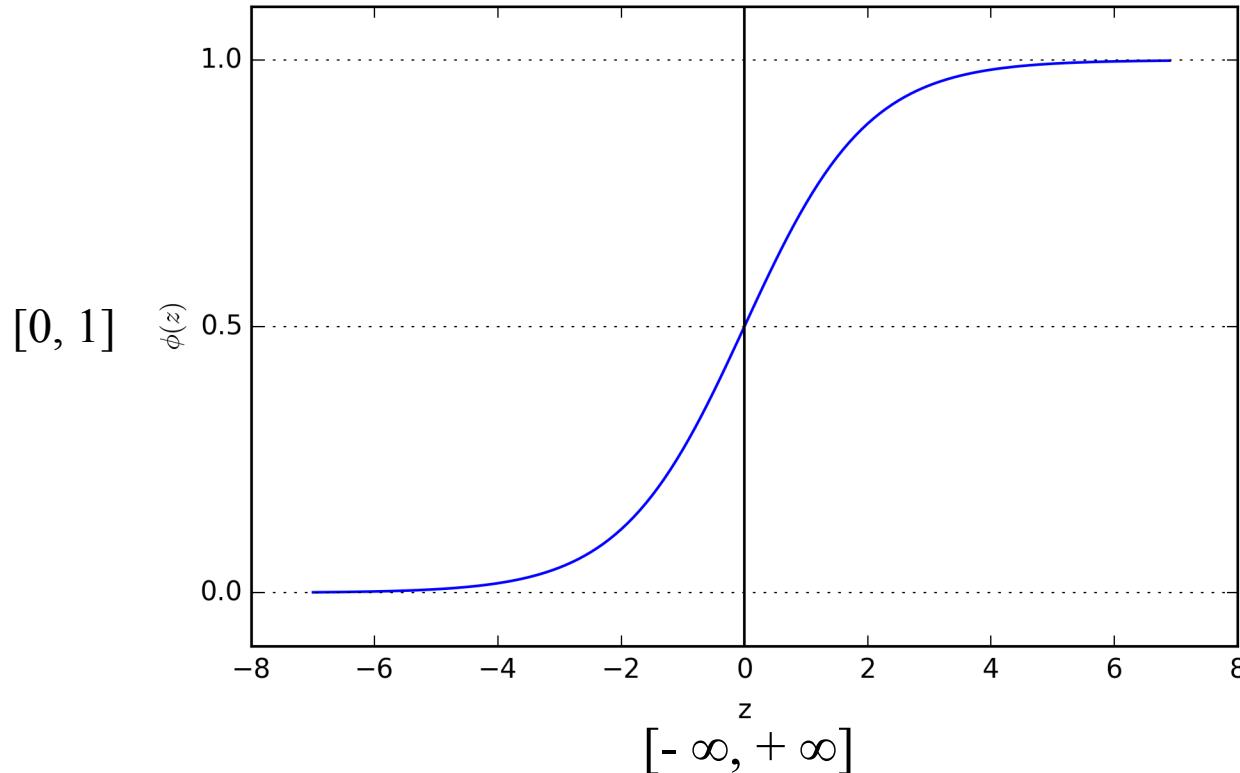
$$\frac{p}{(1 - p)} \quad \begin{array}{l} p = \text{probabilidade de ocorrer um deslizamento} \\ [0, 1] \end{array}$$

$$logit(p) = \log \frac{p}{1 - p} \quad [-\infty, +\infty]$$

$$logit(p(y = 1|\mathbf{x})) = w_0x_0 + w_1x_1 + \dots + x_mw_m = \sum_{i=0}^m w_i x_i = \mathbf{w}^T \mathbf{x} = z$$

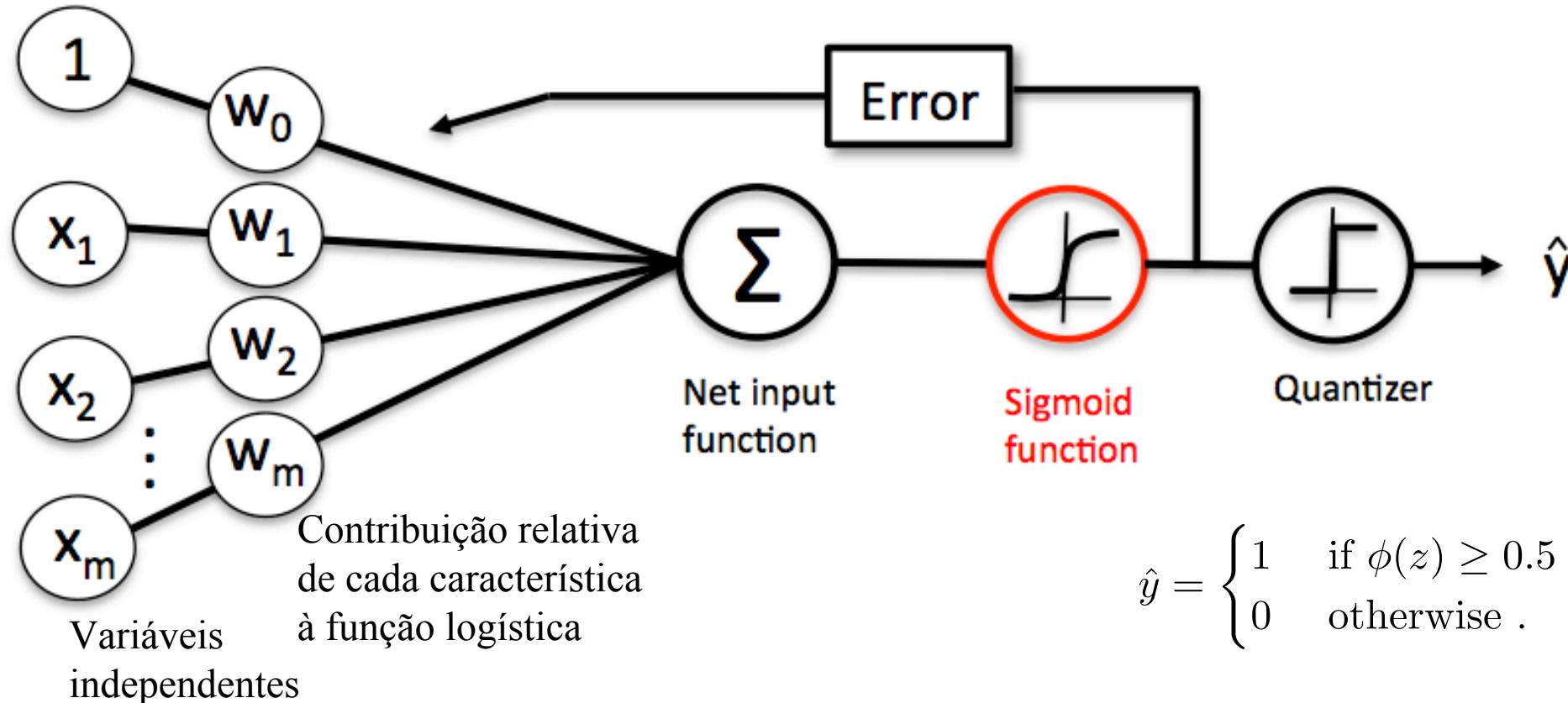
$$\phi(z) = p \quad \frac{p}{1 - p} = e^z \quad \Rightarrow \quad \phi(z) = \frac{1}{1 + e^{-z}} \quad : \text{função } \textcolor{blue}{logistic}$$

$$\phi(z) = \frac{1}{1 + e^{-z}} : \text{sigmoid}$$



Regressão logística

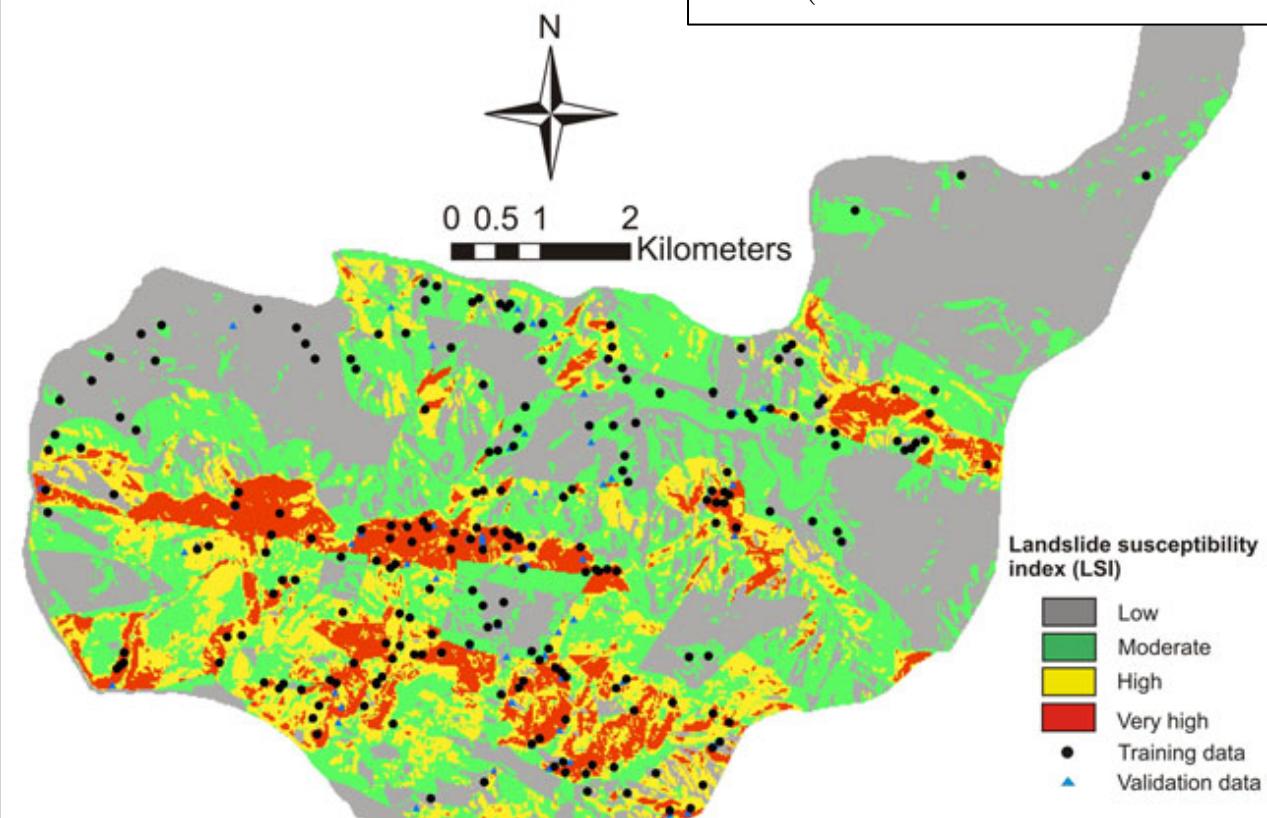
$$z = w_0x_0 + w_1x_1 + \cdots + x_mw_m = \sum_{i=0}^m w_i x_i = \mathbf{w}^T \mathbf{x}$$



$$\hat{y} = \begin{cases} 1 & \text{if } \phi(z) \geq 0.5 \\ 0 & \text{otherwise .} \end{cases}$$

Resultados

$$z = (0.0214 \times \text{Slope}) + \text{Aspect} + (0.08 \times \text{Altitude}) + (0.0221 \times \text{Plan curvature}) \\ + (0.038 \times \text{SPI}) + (-0.2831 \times \text{TWI}) + (-0.002 \times \text{STI}) + \text{Lithology} \\ + \text{Land use} + (-0.000041 \times \text{Distance from faults}) \\ + (-0.00038 \times \text{Distance from roads}) \\ + (-0.0004 \times \text{Distance from rivers}) - 18.991$$



Aprendizado dos pesos

Objetivo: maximizar a *Likelihood* (verossimilhança)

$$L(\mathbf{w}) = P(\mathbf{y}|\mathbf{x}; \mathbf{w}) = \prod_{i=1}^n P(y^{(i)}|x^{(i)}; \mathbf{w}) = \prod_{i=1}^n \left(\phi(z^{(i)}) \right)^{y^{(i)}} \left(1 - \phi(z^{(i)}) \right)^{1-y^{(i)}}$$

Assume-se que as amostras individuais são independentes entre si

Mais fácil: maximizar a *Log-Likelihood* (log da verossimilhança)

$$l(\mathbf{w}) = \log L(\mathbf{w}) = \sum_{i=1}^n \left[y^{(i)} \log \left(\phi(z^{(i)}) \right) + \left(1 - y^{(i)} \right) \log \left(1 - \phi(z^{(i)}) \right) \right]$$

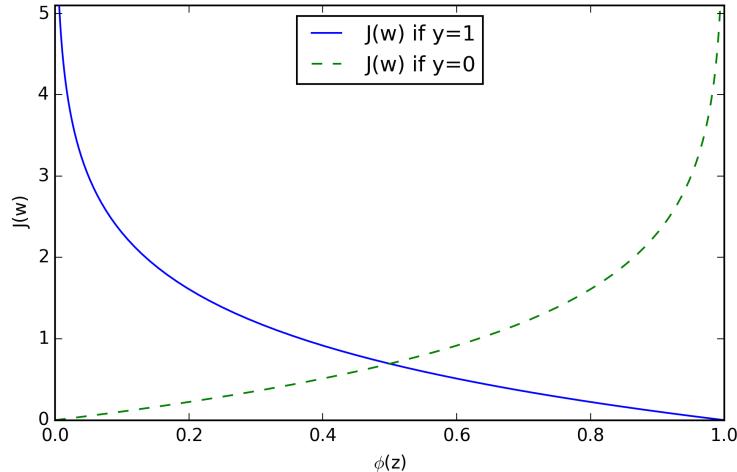
Aprendizado dos pesos

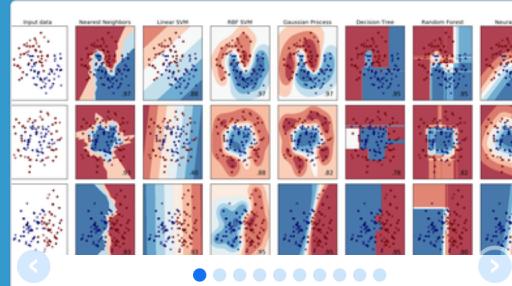
...ou minimizar a função de custo equivalente.

$$J(\mathbf{w}) = \sum_{i=1}^n \left[-y^{(i)} \log \left(\phi(z^{(i)}) \right) - (1 - y^{(i)}) \log \left(1 - \phi(z^{(i)}) \right) \right]$$

$$J(\phi(z), y; \mathbf{w}) = -y \log (\phi(z)) - (1 - y) \log (1 - \phi(z)).$$

$$J(\phi(z), y; \mathbf{w}) = \begin{cases} -\log (\phi(z)) & \text{if } y = 1 \\ -\log (1 - \phi(z)) & \text{if } y = 0 \end{cases}$$





scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.

Algorithms: SVM, nearest neighbors, random forest, ...

[— Examples](#)

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, ridge regression, Lasso, ...

[— Examples](#)

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, ...

[— Examples](#)

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Algorithms: PCA, feature selection, non-negative matrix factorization.

[— Examples](#)

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning

Modules: grid search, cross validation, metrics.

[— Examples](#)

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.

Modules: preprocessing, feature extraction.

[— Examples](#)

Exemplo: base Íris

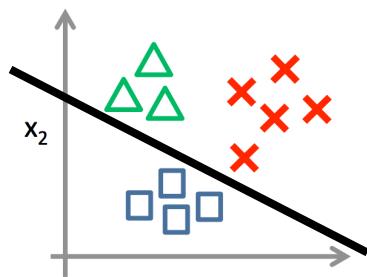
```
from sklearn import datasets  
import numpy as np  
  
iris = datasets.load_iris() ← Leitura da base  
X = iris.data[:, [2, 3]]  
y = iris.target  
  
print('Class labels:', np.unique(y))
```

```
from sklearn.preprocessing import StandardScaler  
  
sc = StandardScaler()  
sc.fit(X_train) ← ???  
X_train_std = sc.transform(X_train)  
X_test_std = sc.transform(X_test)
```

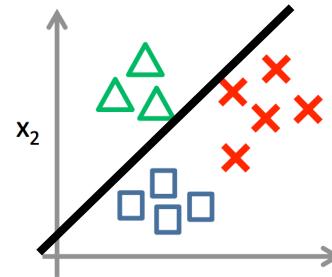
Treinamento

```
from sklearn.linear_model import LogisticRegression  
  
lr = LogisticRegression(C=1000.0, random_state=0)  
lr.fit(X_train_std, y_train) ←  
  
plot_decision_regions(X_combined_std, y_combined,  
                      classifier=lr, test_idx=range(105, 150))
```

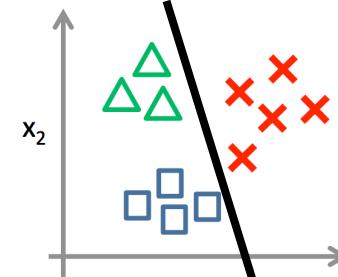
One-vs-all (one-vs-rest):



One-vs-all (one-vs-rest):



One-vs-all (one-vs-rest):

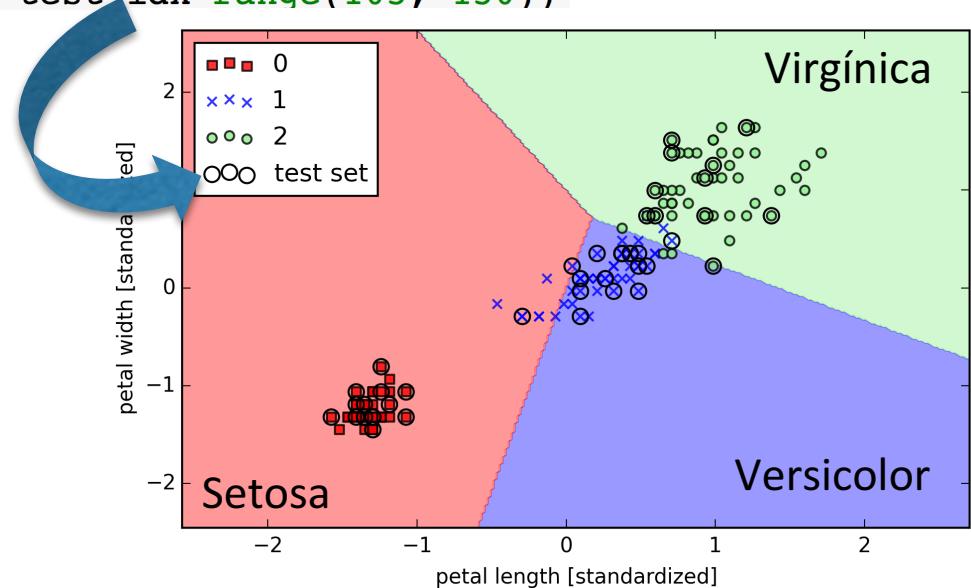


Teste

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(C=1000.0, random_state=0)
lr.fit(X_train_std, y_train)

plot_decision_regions(X_combined_std, y_combined,
                      classifier=lr, test_idx=range(105, 150))
```



Execução do modelo

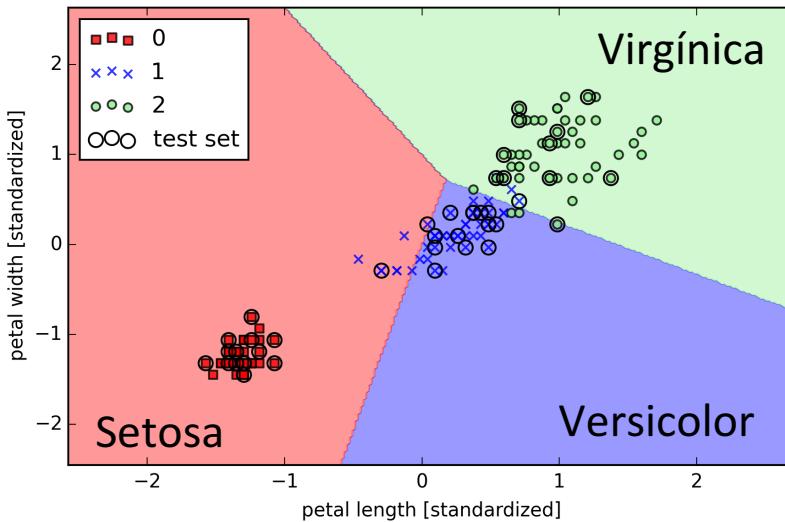
Amostras

```
lr.predict_proba(X_test_std[17,:])
```

```
array([[ 0.000, 0.995, 0.005]])
```

```
lr.predict_proba(X_test_std[37,:])
```

```
array([[ 0.000, 0.383, 0.617]])
```



Problemas com o treinamento de modelos

Overfitting (super-especialização) = Modelo com **alta variância**

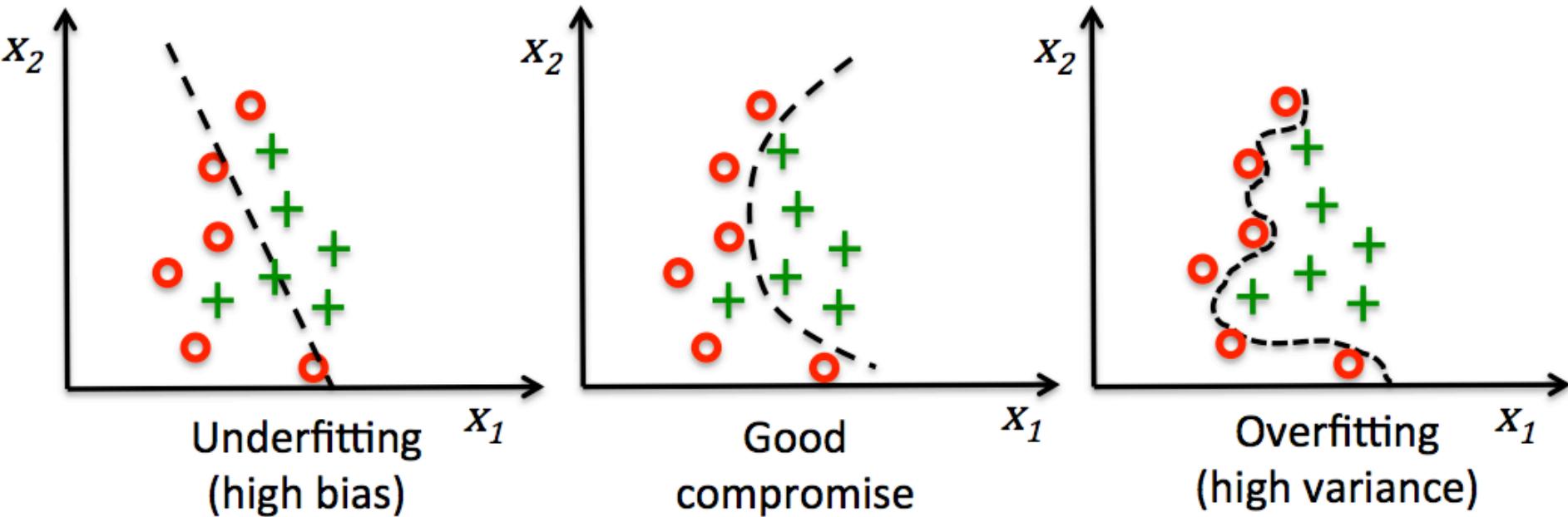
- Pode ser causado por uma quantidade muito grande parâmetros que faz o modelo complexo demais em relação aos dados

Mede o quanto um modelo é sensível à aleatoriedade dos dados de treinamento

Underfitting (sub-especialização) = Modelo com **alto viés (bias)**

- Modelo não é complexo o suficiente para capturar o padrão nos dados de treinamento

Mede o erro sistemático do modelo, que não é afetado pela aleatoriedade dos dados de treino

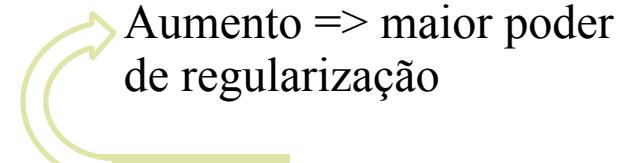


Método para encontrar um bom tradoff *bias-variance*

Regularização

Rationale: introduzir informação adicional para penalizar valores muito altos de pesos

Regularização $L2$



$$J(\mathbf{w}) = -\sum_{i=1}^n \left[y^{(i)} \log (\phi(z^{(i)})) - (1 - y^{(i)}) \log (1 - \phi(z^{(i)})) \right] + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

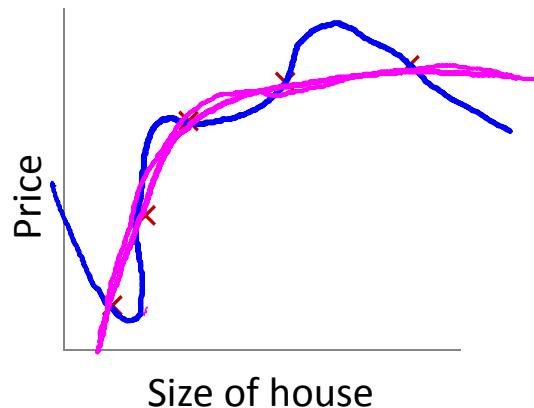
$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j} = \eta \sum_{i=1}^n \left(y^{(i)} - \phi(z^{(i)}) \right) x_j^{(i)} - \eta \lambda w_j$$

$$\frac{\lambda}{2} \|\mathbf{w}\|^2 = \frac{\lambda}{2} \sum_{j=1}^m w_j^2$$

for $j \in \{1, 2, \dots, m\}$ (i.e., $j \neq 0$) pois o peso w_0 não passa por regularização.

Regularização $L2$...: intuição

$$J(\mathbf{w}) = - \sum_{i=1}^n \left[y^{(i)} \log (\phi(z^{(i)})) - (1 - y^{(i)}) \log (1 - \phi(z^{(i)})) \right] + \frac{\lambda}{2} \|\mathbf{w}\|^2$$



$$\underline{w_0 + w_1 x_1 + \cdots + x_m w_m}$$



Q: What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?

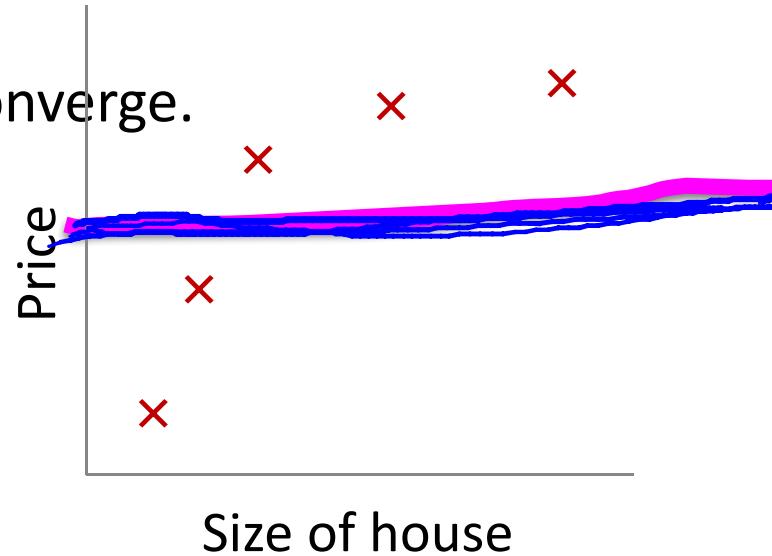
- a. Algorithm works fine; setting λ to be very large can't hurt it
- b. Algorithm fails to eliminate overfitting.
- c. Algorithm results in underfitting. (Fails to fit even training data well).
- d. Gradient descent will fail to converge.

Q: What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?

- a. Algorithm works fine; setting λ to be very large can't hurt it
- b. Algorithm fails to eliminate overfitting.
- C.** Algorithm results in underfitting. (Fails to fit even training data well).
- d. Gradient descent will fail to converge.

$$w_0 + w_1 x_1 + \cdots + w_m x_m$$

Blue arrows point from the w_i terms to the x_i terms.



Regularização $L2$

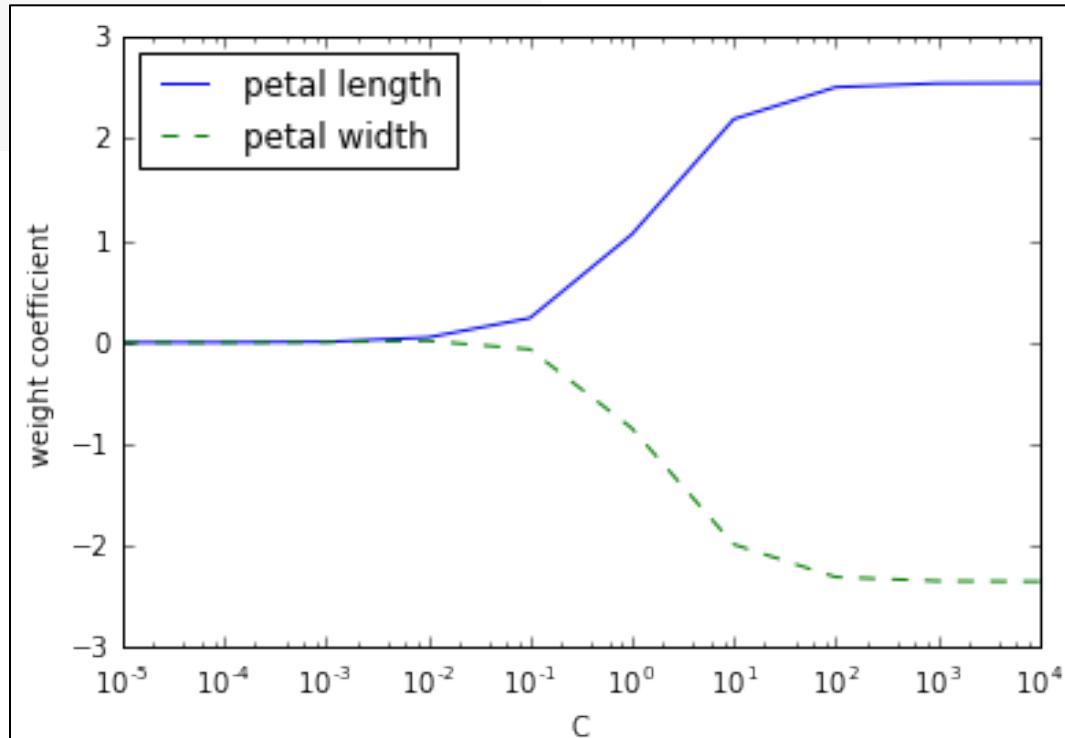
$$J(\mathbf{w}) = - \sum_{i=1}^n \left[y^{(i)} \log (\phi(z^{(i)})) - (1 - y^{(i)}) \log (1 - \phi(z^{(i)})) \right] + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

$$\text{Se} \quad C = \frac{1}{\lambda}$$

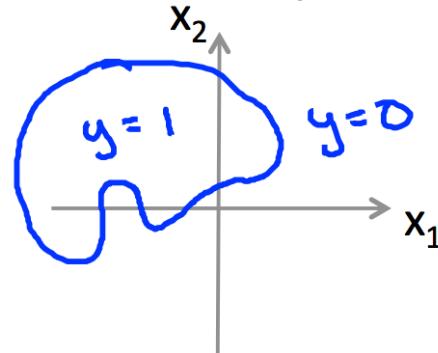
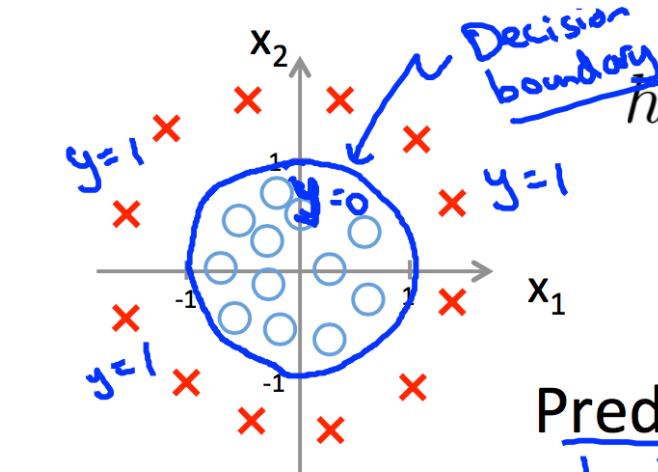
$$J(\mathbf{w}) = C \left[\sum_{i=1}^n \left(-y^{(i)} \log (\phi(z^{(i)})) - (1 - y^{(i)}) \log (1 - \phi(z^{(i)})) \right) \right] + \frac{1}{2} \|\mathbf{w}\|^2$$

```
weights, params = [], []
for c in np.arange(-5, 5):
    lr = LogisticRegression(C=10**c, random_state=0)
    lr.fit(X_train_std, y_train)
    weights.append(lr.coef_[1])
    params.append(10**c)
```

```
weights = np.array(weights)
```



Limites de decisão não lineares



$$h_{\theta}(x) = g(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2)$$

$$\Theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

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

$$h_{\theta}(x) = g(w_0 + w_1 x_1 + w_2 x_2 + w_3 \underline{x_1^2} + w_4 \underline{x_1^2 x_2} + w_5 \underline{x_1^2 x_2^2} + w_6 \underline{x_1^3 x_2} + \dots)$$

Divulgação científica



Hendrik Macedo

Escreve sobre Inteligência Artificial no Saense.

<http://www.saense.com.br/autores/artigos-publicados-por-hendrik-macedo/>