

Spatial  
Ecology

# Perceptron

Antonio Fonseca

# Agenda

## 1) The “less optimal” hyperplane methods

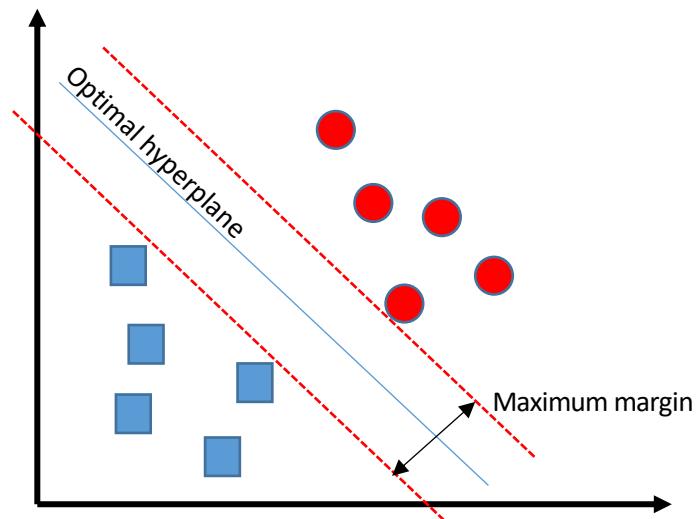
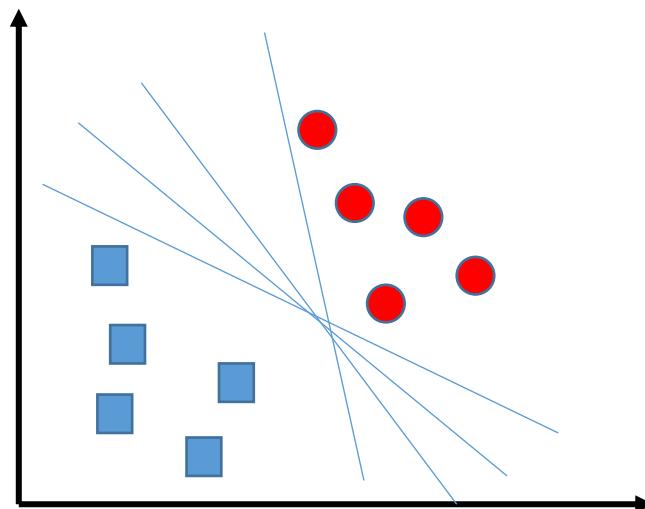
- Links between SVM and Logistic Regression
- Review on Linear Regression
- Minimizing loss functions
- Regularization

## 2) Perceptron

- The universal approximator
- Intro to optimizers
- Hands-on tutorial

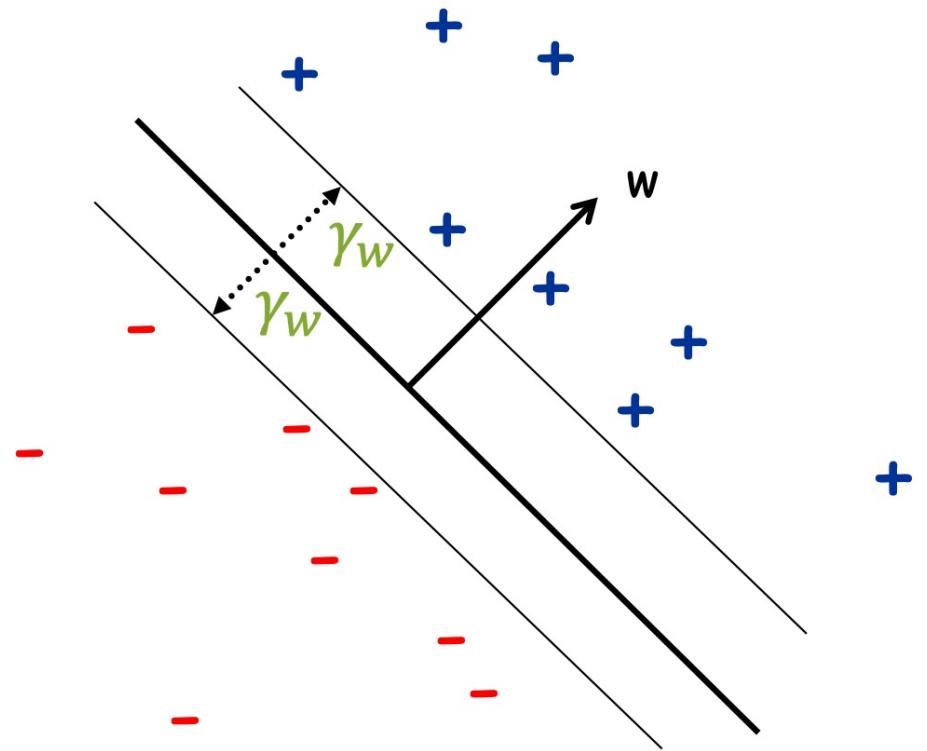
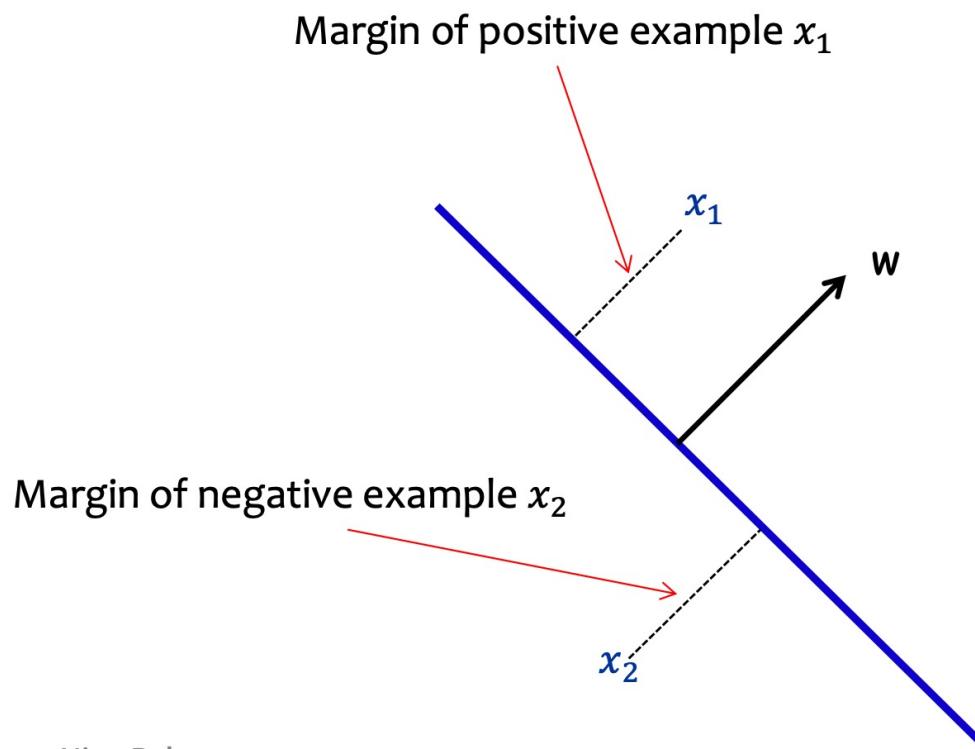
# Support Vector Machine

Find **the optimal** hyperplane in an N-dimensional space that distinctly classifies the data points.



# Geometric Margin

Definition: The margin of example  $x$  w.r.t. a linear sep.  $w$  is the distance from  $x$  to the plane



# SVM Optimization

Hinge loss function

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases}$$

Loss function for the SVM

$$\min_w \lambda \|w\|^2 + \sum_{i=1}^n (1 - y_i \langle x_i, w \rangle)_+$$

Gradients

$$\frac{\delta}{\delta w_k} \lambda \|w\|^2 = 2\lambda w_k$$

$$\frac{\delta}{\delta w_k} (1 - y_i \langle x_i, w \rangle)_+ = \begin{cases} 0, & \text{if } y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases}$$

Updating the weights:

No misclassification

$$w = w - \alpha \cdot (2\lambda w)$$

Misclassification

$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w)$$

# SVM for Regression

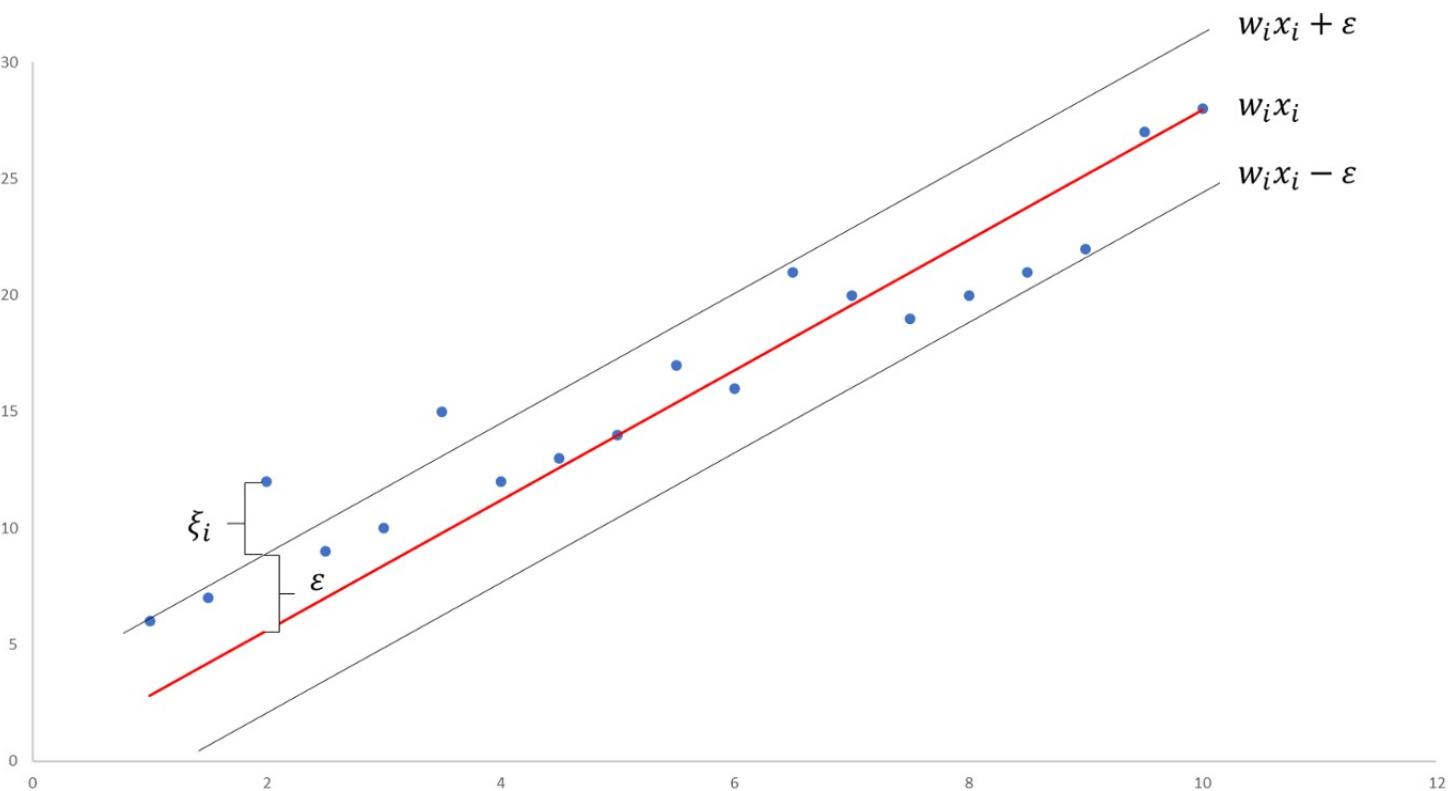
For any value that falls outside of  $\epsilon$ , we can denote its deviation from the margin as  $\xi$ .

Loss

$$\text{MIN } \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n |\xi_i|$$

Constraints

$$|y_i - w_i x_i| \leq \epsilon + |\xi_i|$$



# SVM for Regression

- The best fit line is the hyperplane that has the maximum number of points.
- Limitations
  - The fit time complexity of SVR is more than quadratic with the number of samples
  - SVR scales poorly with number of samples (e.g., >10k samples). For large datasets, **Linear SVR** or **SGD Regressor**
  - Underperforms in cases where the number of features for each data point exceeds the number of training data samples
  - Underperforms when the data set has more noise, i.e. target classes are overlapping.

# A “non-optimal” hyperplane approach

$$\min_w \lambda \|w\|^2 + \sum_i \max\{0, 1 - y_i w^T x_i\}$$

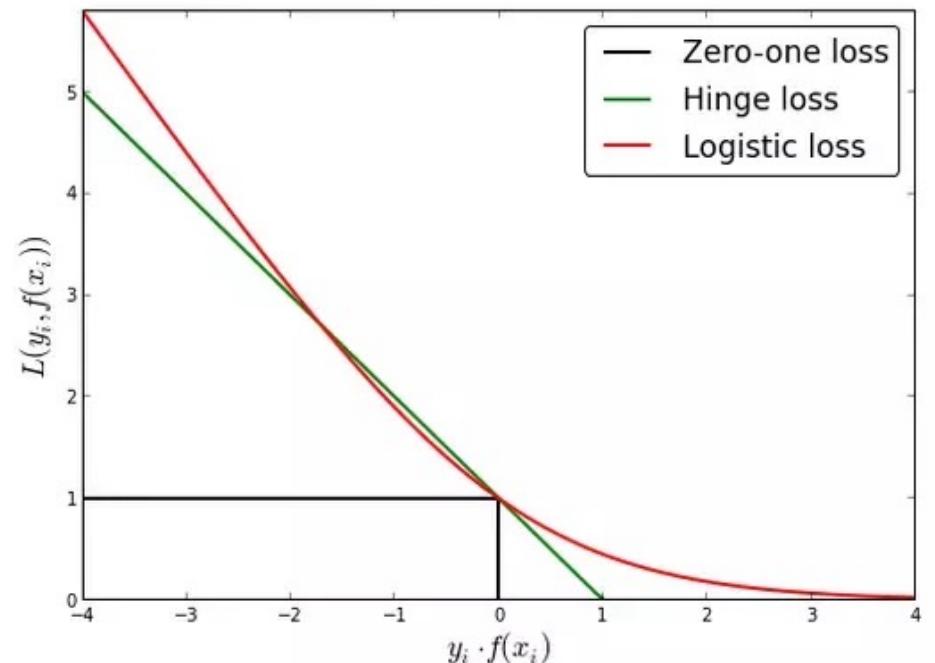
SVM

$$\min_w \lambda \|w\|^2 + \sum_i \log(1 + \exp(1 - y_i w^T x_i))$$

Logistic Regression (LR)

Main differences:

- SVM maximizes the margin between the closest support vectors
- LR maximizes the class probability
- SVM produces discrete classes (1 or 0)
- LR produces probabilistic values (sigmoid of the log likelihood function)



# Review on Linear Regression

*Task (T)*

$$\begin{array}{l} \text{Input } x \in \mathbb{R}^n \\ \text{Weights } w \in \mathbb{R}^n \end{array} \quad \hat{y} = w^T x$$

$$f(x, w) = x_1 w_1 + x_2 w_2 + \cdots + x_n w_n$$

*Dataset*

$$(X, y) \quad \left\{ \begin{array}{l} (X_{train}, y_{train}) \\ (X_{test}, y_{test}) \end{array} \right.$$

*Performance (P)*

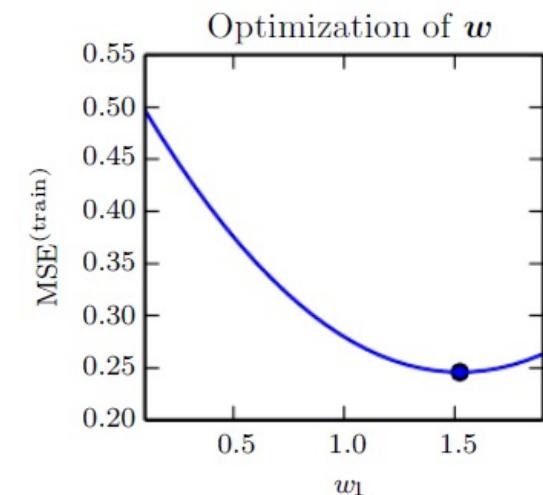
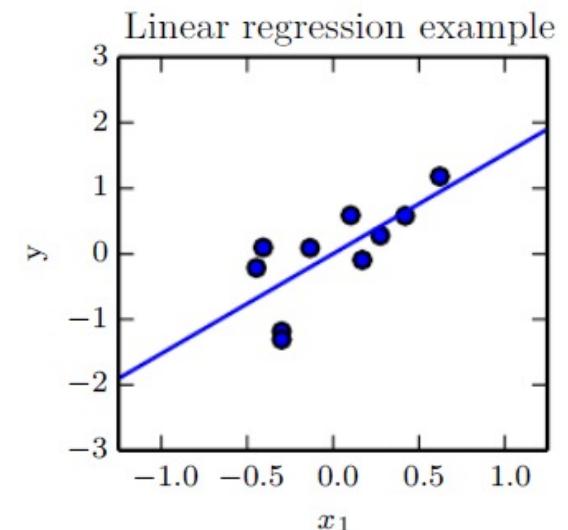
$$MSE_{test} = \frac{1}{m} \sum_i (\hat{y}_{test} - y_{test})_i^2$$

*Training*

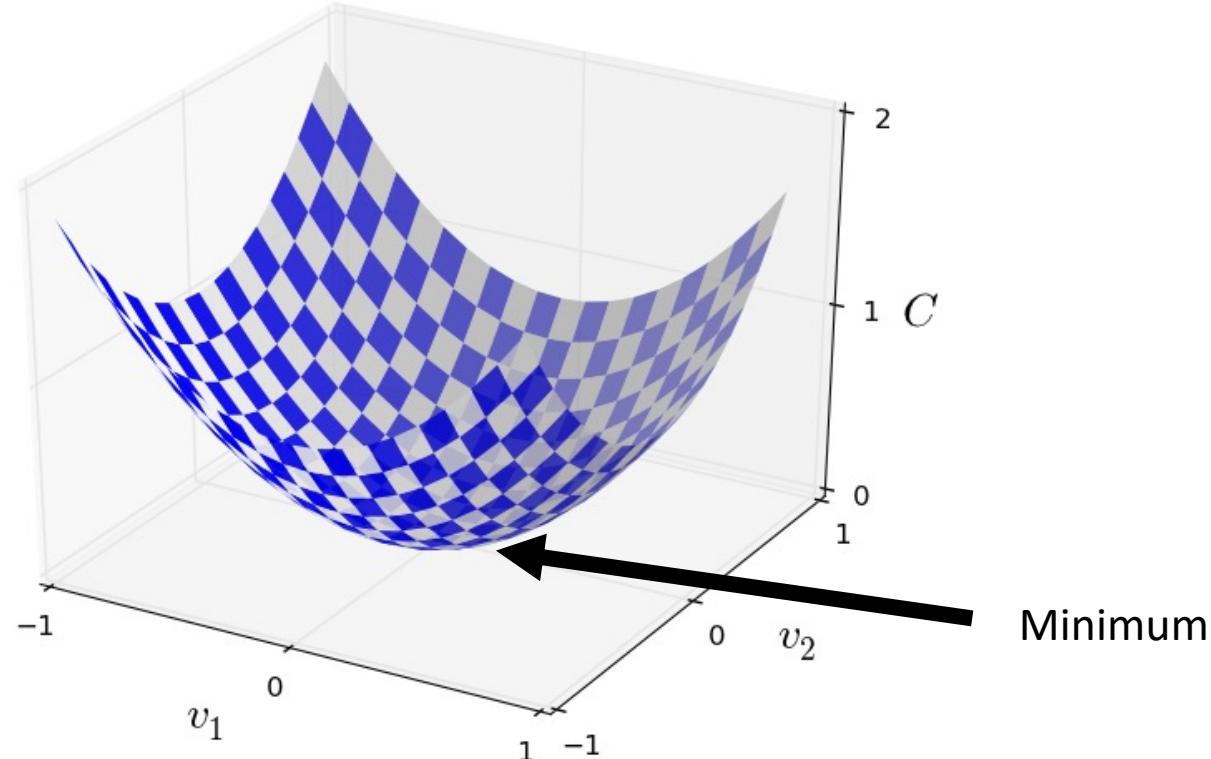
$$\nabla_w \left( \frac{1}{m} \sum_i (w^T X_{train} - y_{train})_i^2 \right) = 0$$

Solves linear problems

Can't solve more complex problems (e.g., XOR problem)



# Loss Minimization



Convex loss functions can be solved by differentiation, at the point where Loss is minimum the derivative wrt to parameters should be 0!

# Linear Regression Optimization

- Add an offset  $w_0$ :  $f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T \mathbf{x} + w_0$ ,  $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots (\mathbf{x}_n, y_n)\}$

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \sum_{i=1}^n (\mathbf{w}^T \mathbf{x}_i + w_0 - y_i)^2$$

$$= \arg \min_{\mathbf{w}} L(\mathbf{w}; \mathcal{D})$$

- Set  $\frac{\partial L(\mathbf{w}; \mathcal{D})}{\partial w_i} = 0$  for each  $i$

# Mean squared error loss

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \sum_{i=1}^n (\mathbf{w}^T \mathbf{x}_i + w_0 - y_i)^2$$

Rewrite:

$$\begin{aligned}(X\mathbf{w} - \mathbf{y})^T(X\mathbf{w} - \mathbf{y}) &= (\mathbf{w}^T X^T - \mathbf{y}^T)(X\mathbf{w} - \mathbf{y}) \\&= \mathbf{w}^T X^T X \mathbf{w} - \mathbf{w}^T X^T \mathbf{y} - \mathbf{y}^T X \mathbf{w} + \mathbf{y}^T \mathbf{y} \\&= \mathbf{w}^T X^T X \mathbf{w} - 2\mathbf{w}^T X^T \mathbf{y} + \mathbf{y}^T \mathbf{y}.\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial w} \mathbf{w}^T X^T X \mathbf{w} - 2\mathbf{w}^T X^T \mathbf{y} + \mathbf{y}^T \mathbf{y} &= 0 \\2X^T X \mathbf{w} - 2X^T \mathbf{y} &= 0 \\X^T X \mathbf{w} &= X^T \mathbf{y} \\\mathbf{w} &= (X^T X)^{-1} X^T \mathbf{y}\end{aligned}$$

# Regularization

- Ridge regression: penalize with L2 norm

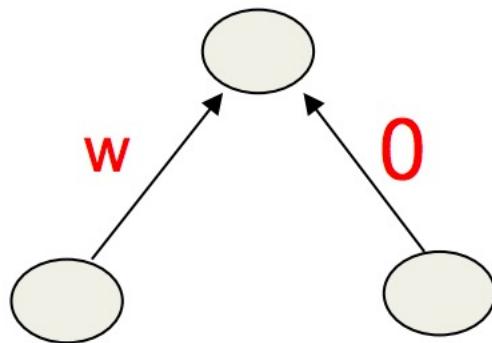
$$\mathbf{w}^* = \arg \min \sum_i L(f(\mathbf{x}_i; \mathbf{w}), y_i) + \lambda \sum_{j=1}^m w_j^2$$

- Closed form solution exists  $\mathbf{w}^* = (\lambda I + X^T X)^{-1} X^T \mathbf{y}$
- LASSO regression: penalize with L1 norm

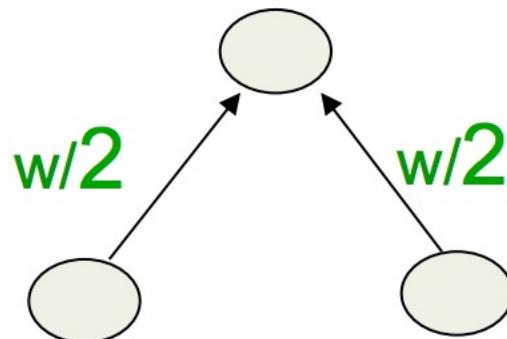
$$\mathbf{w}^* = \arg \min \sum_i L(f(\mathbf{x}_i; \mathbf{w}), y_i) + \lambda \sum_{j=1}^m |w_j|$$

- No closed form solution but still convex (optimal solution can be found)

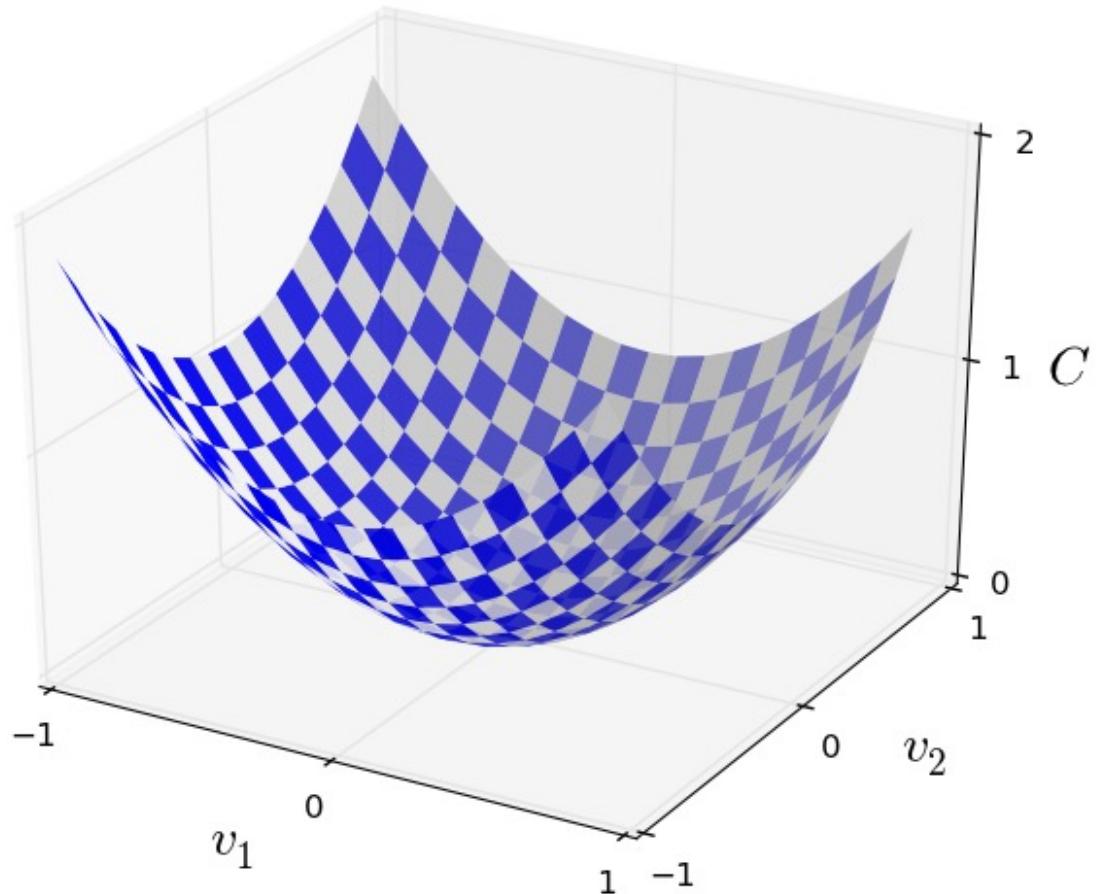
# Regularization



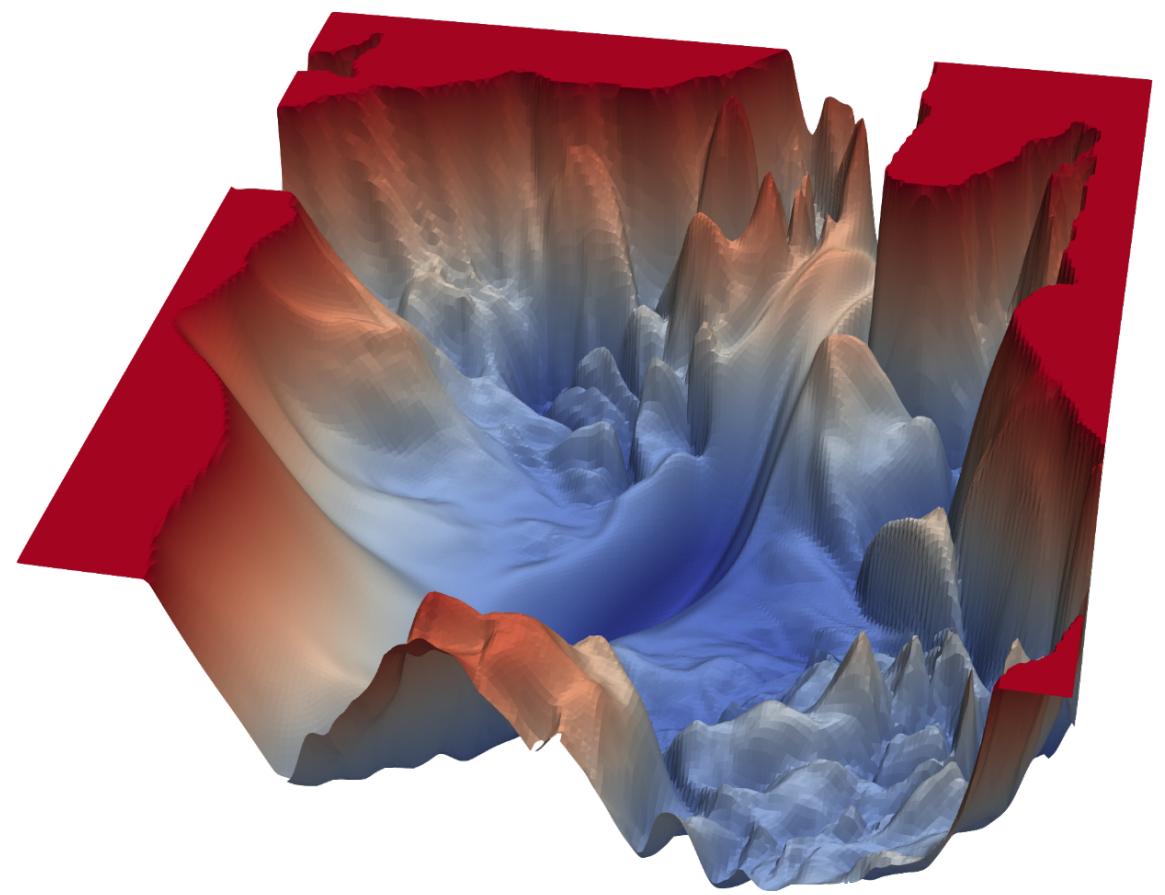
- Prefers to share smaller weights
- Makes model smoother
- More Convex



# Expectation

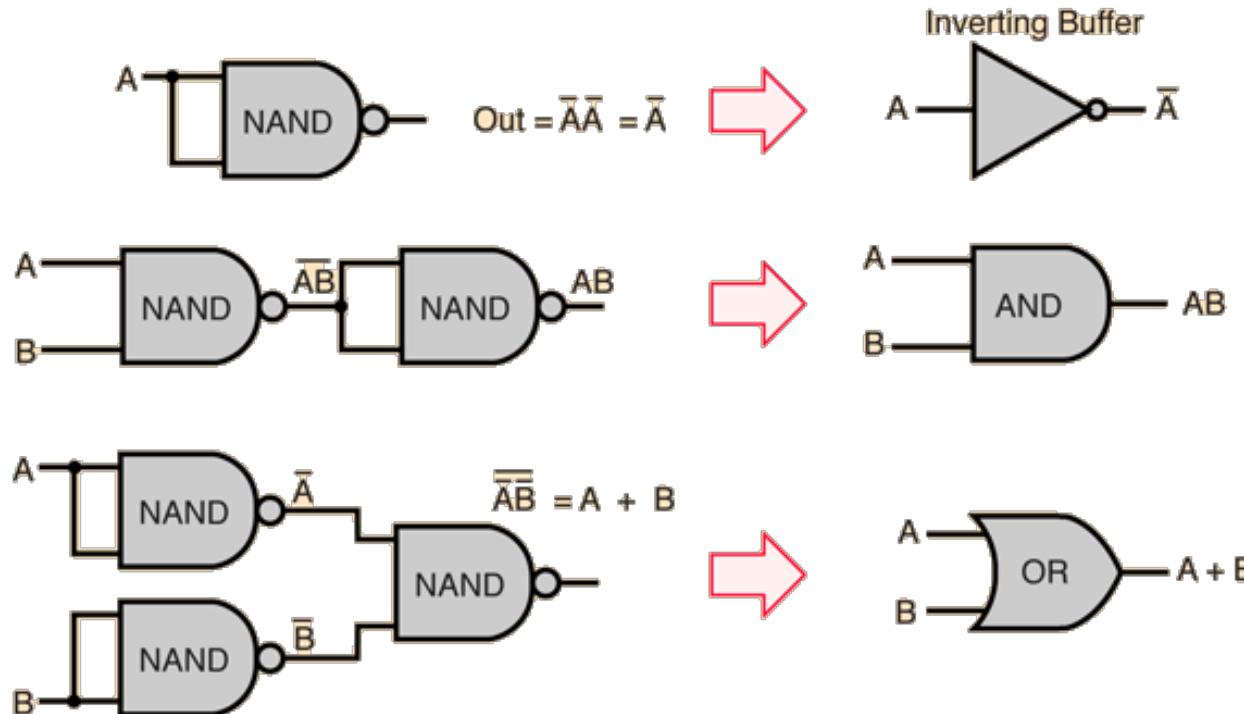


# Reality



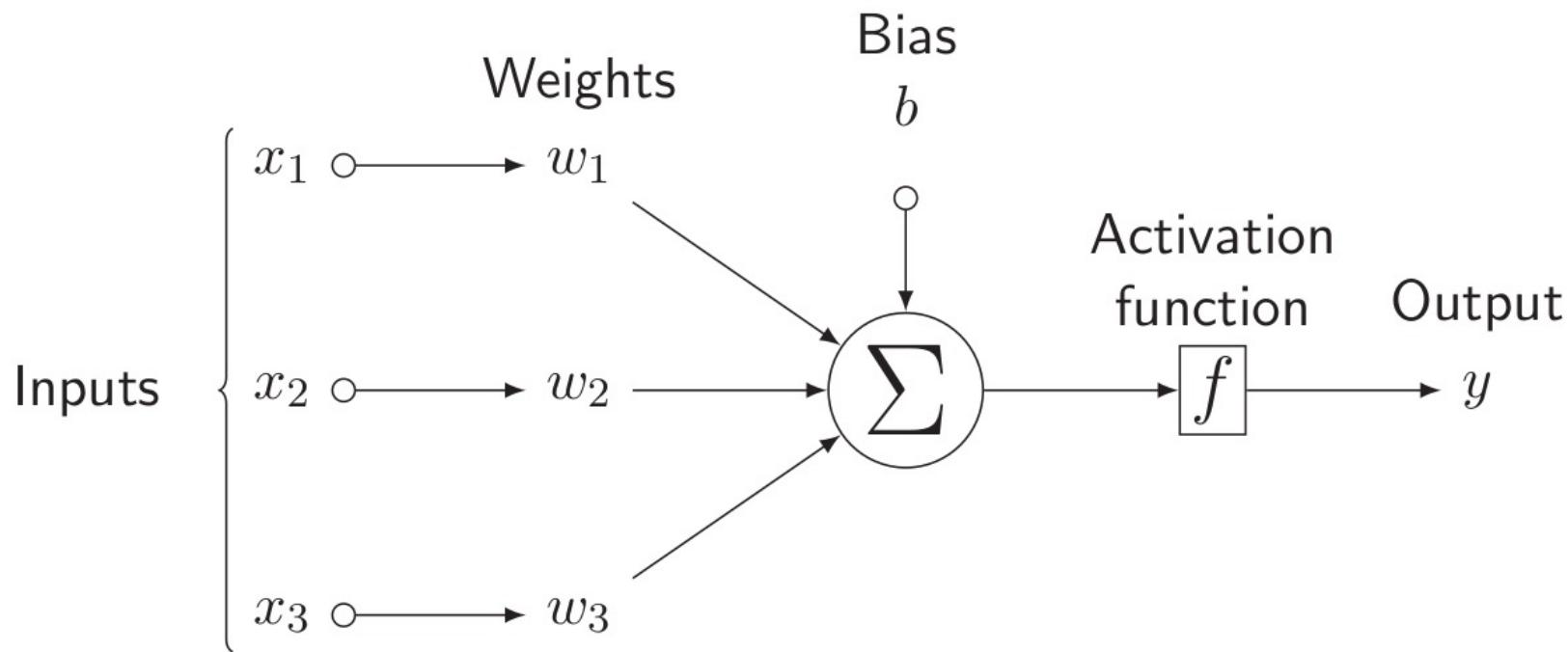
# Logic circuits with perceptrons

- NAND gates can be constructed from perceptrons
- NAND gates are universal for computation
  - Any computation can be built from NAND gates
  - Therefore, perceptrons are universal for computation



# Perceptron: Threshold Logic

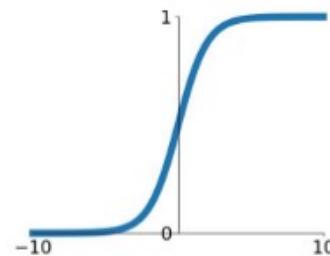
$$\mathcal{L}_{\text{perc}}(\mathbf{x}, y) = \begin{cases} 0 & \text{if } y\mathbf{w}^\top \mathbf{f}(\mathbf{x}) > 0 \\ -y\mathbf{w}^\top \mathbf{f}(\mathbf{x}) & \text{if } y\mathbf{w}^\top \mathbf{f}(\mathbf{x}) \leq 0 \end{cases}$$



# Activation functions

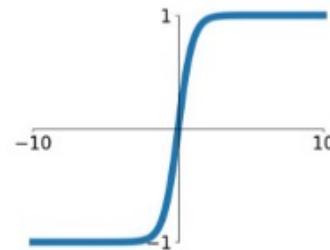
**Sigmoid**

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



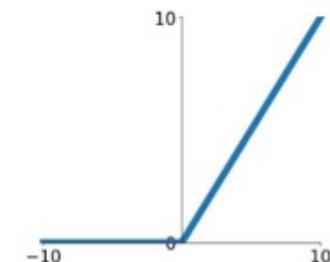
**tanh**

$$\tanh(x)$$



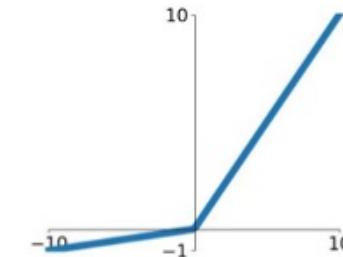
**ReLU**

$$\max(0, x)$$



**Leaky ReLU**

$$\max(0.1x, x)$$

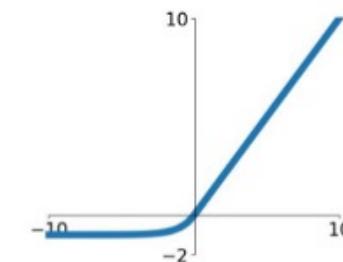


**Maxout**

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

**ELU**

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



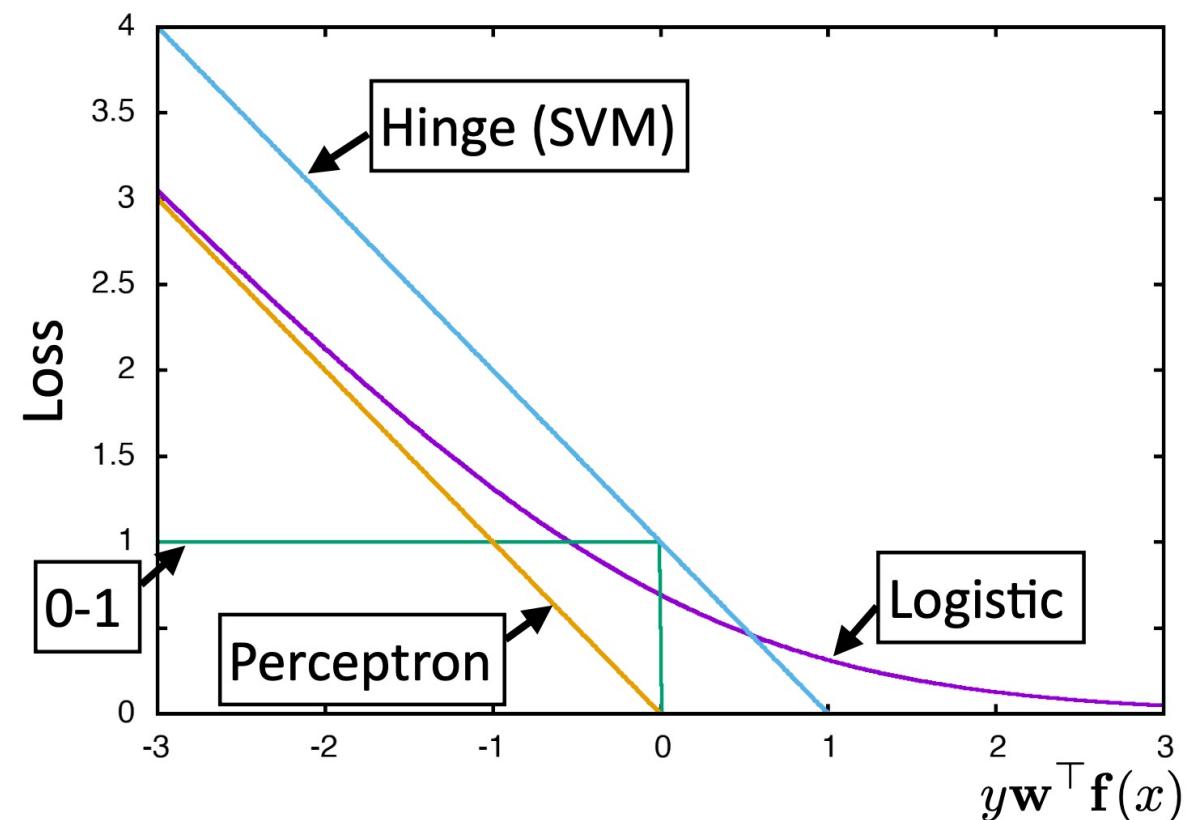
# (Putting things in perspective)

$$\mathcal{L}_{\text{lr}}(\mathbf{x}, y) = \begin{cases} -y\mathbf{w}^\top \mathbf{f}(\mathbf{x}) + \log(1 + \exp(y\mathbf{w}^\top \mathbf{f}(\mathbf{x}))) & \text{if } y = +1 \text{ (positive)} \\ \log(1 + \exp(-y\mathbf{w}^\top \mathbf{f}(\mathbf{x}))) & \text{if } y = -1 \text{ (negative)} \end{cases}$$

$$\mathcal{L}_{\text{perc}}(\mathbf{x}, y) = \begin{cases} 0 & \text{if } y\mathbf{w}^\top \mathbf{f}(\mathbf{x}) > 0 \\ -y\mathbf{w}^\top \mathbf{f}(\mathbf{x}) & \text{if } y\mathbf{w}^\top \mathbf{f}(\mathbf{x}) \leq 0 \end{cases}$$

Main differences:

- Perceptron: gradient-based optimization
- LR: probabilistic model
- Perceptron: if the data are linearly separable, perceptron is guaranteed to converge.
- LR: likelihood can never truly be maximized with a finite  $\mathbf{w}$  vector.



# Optimizers

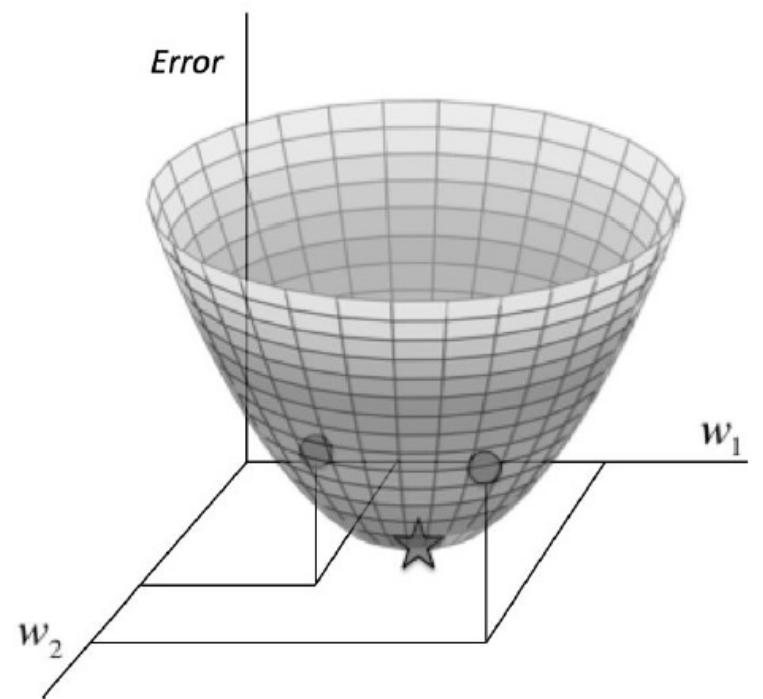
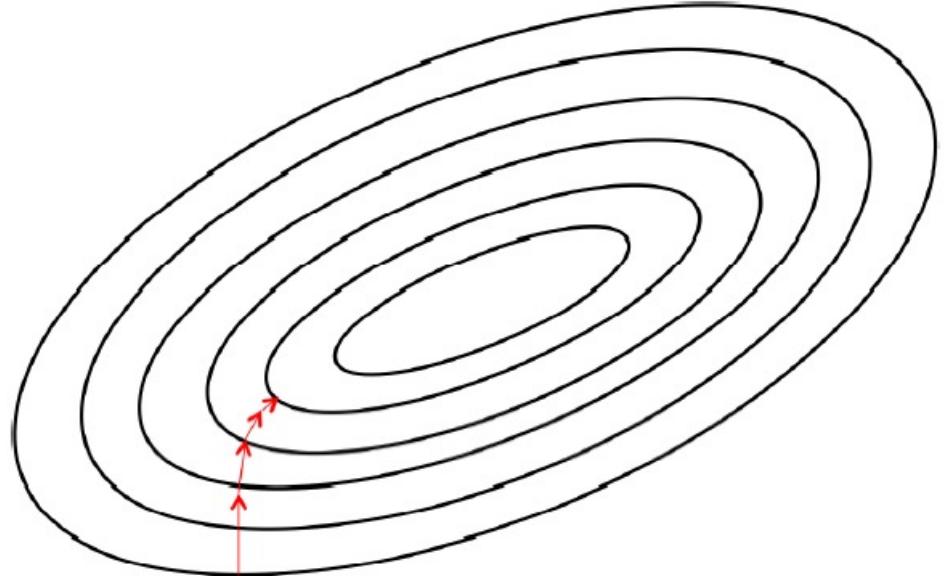
Gradient

$$\Delta w_k = -\frac{\partial E}{\partial w_k}$$

$$= -\frac{\partial}{\partial w_k} \left( \frac{1}{m} \sum_i (w^T X_i - y_i)_i^2 \right)$$

$$w_{i+1} = w_i + \Delta w_k$$

Stochastic gradient descent (**SGD**)



# Optimizers

## Hyperparameters

- Learning rate ( $\alpha$ )

$$\begin{aligned}\Delta w_k &= -\alpha \frac{\partial E}{\partial w_k} \\ &= -\alpha \frac{\partial}{\partial w_k} \left( \frac{1}{m} \sum_i (w^T X_i - y_i)_i^2 \right)\end{aligned}$$

$$w_{i+1} = w_i + \Delta w_k$$

## Stochastic gradient descent (SGD)

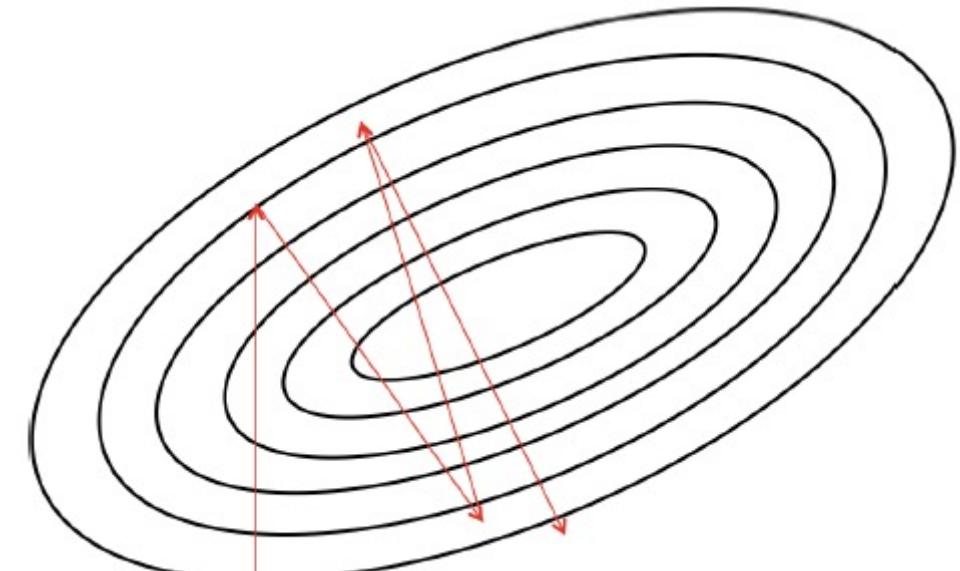
Practical test:

lr\_val = [1; 0.1; 0.01]

momentum\_val = 0

nesterov\_val = 'False'

decay\_val = 1e-6



Result of a large learning rate  $\alpha$

# Optimizers

Hyperparameters

- Learning rate ( $\alpha$ )

$$\Delta w_k = -\alpha \frac{\partial E}{\partial w_k}$$

$$= -\alpha \frac{\partial}{\partial w_k} \left( \frac{1}{m} \sum_i (w^T X_i - y_i)_i^2 \right)$$

$$w_{i+1} = w_i + \Delta w_k$$

Stochastic gradient descent (**SGD**)



Watch out for local minimal areas

