Perceptron

Antonio Fonseca

Agenda

1) Finalize SVM/SVR (remaining from Class 1)

2) Introduction to optimization

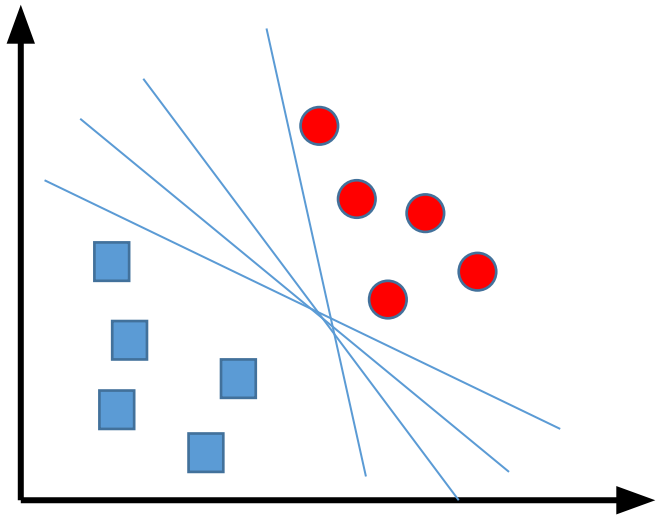
- Review on Linear Regression
- Minimizing loss functions
- Regularization

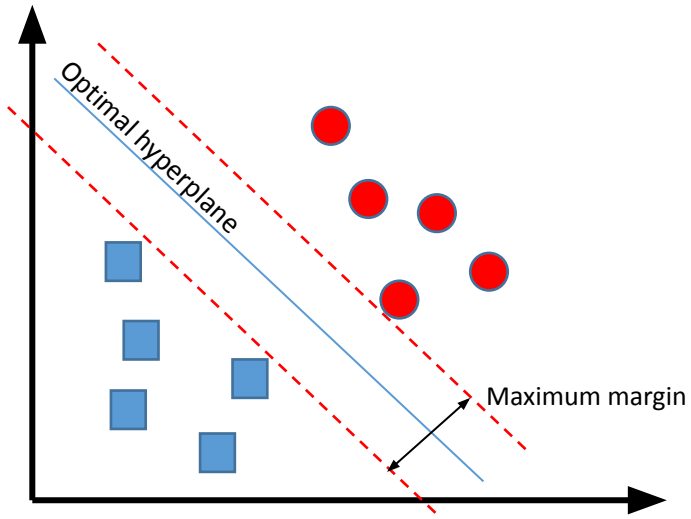
3) Perceptron

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

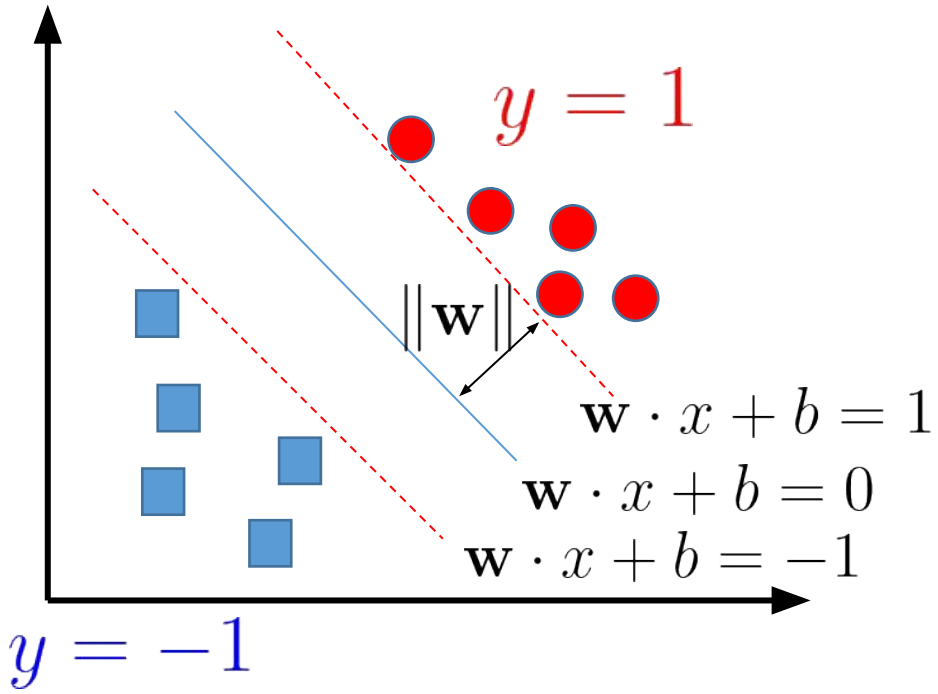
Support Vector Machine

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





Support Vector Machine



Hyperplane equation: $f(x) = \mathbf{w} \cdot x + b$

Distance (D) from a point to the hyperplane

$$D = \frac{|\mathbf{w} \cdot x + b|}{\|\mathbf{w}\|}$$

Minimize the weights, increase distance

Classification task

$$\begin{cases} wx_i + b \geq +1 & \text{when } y_i = +1 \\ wx_i + b \leq -1 & \text{when } y_i = -1, \end{cases}$$

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

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

Gradients

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

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

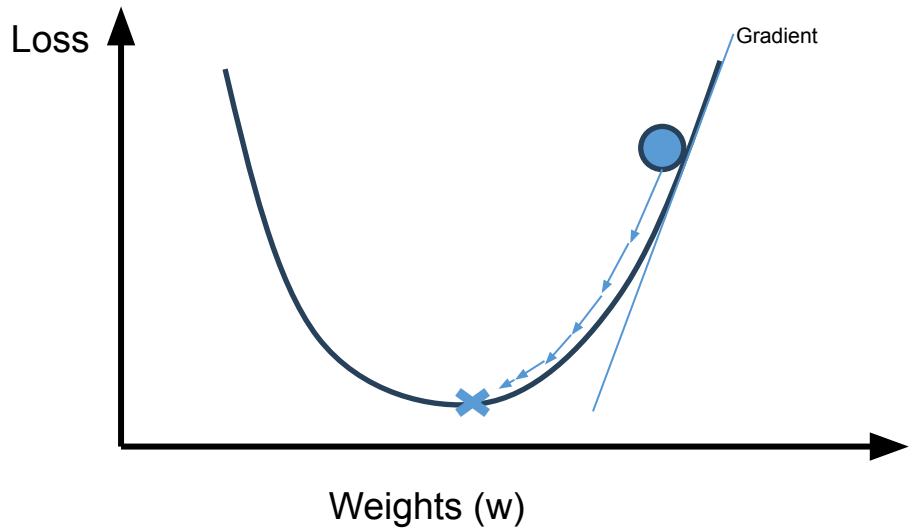
Updating the weights:

No misclassification

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

Misclassification

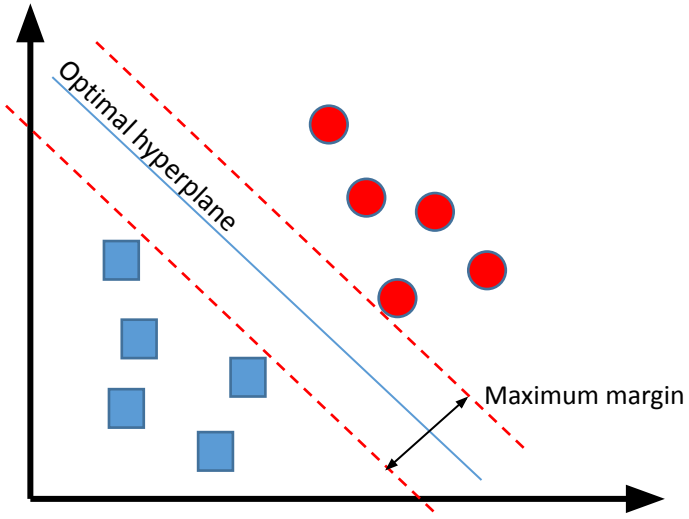
$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w)$$



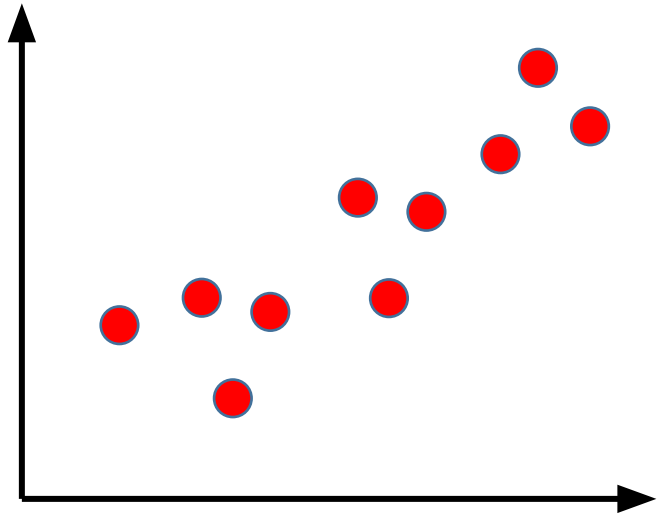
Support Vector Machine for Regression

How do I turn the SVM into a SVR?

SVM



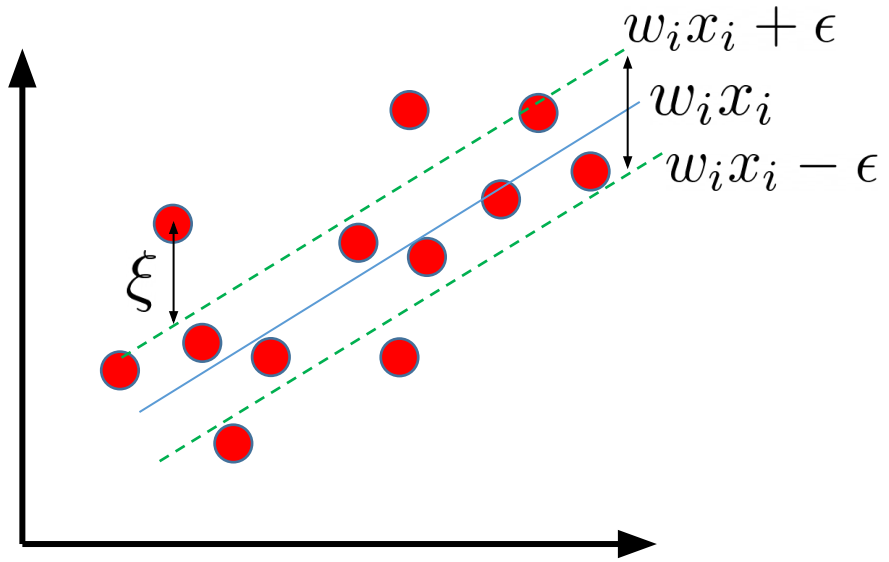
SVR(?)



SVR Optimization

Loss

$$L(y, f(x, \mathbf{w})) = \begin{cases} 0, & |y - f(x, \mathbf{w})| \leq \epsilon \\ |y - f(x, \mathbf{w})| & \text{o.w. ,} \end{cases}$$



Constraints

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

Deviation from the margin (slack)

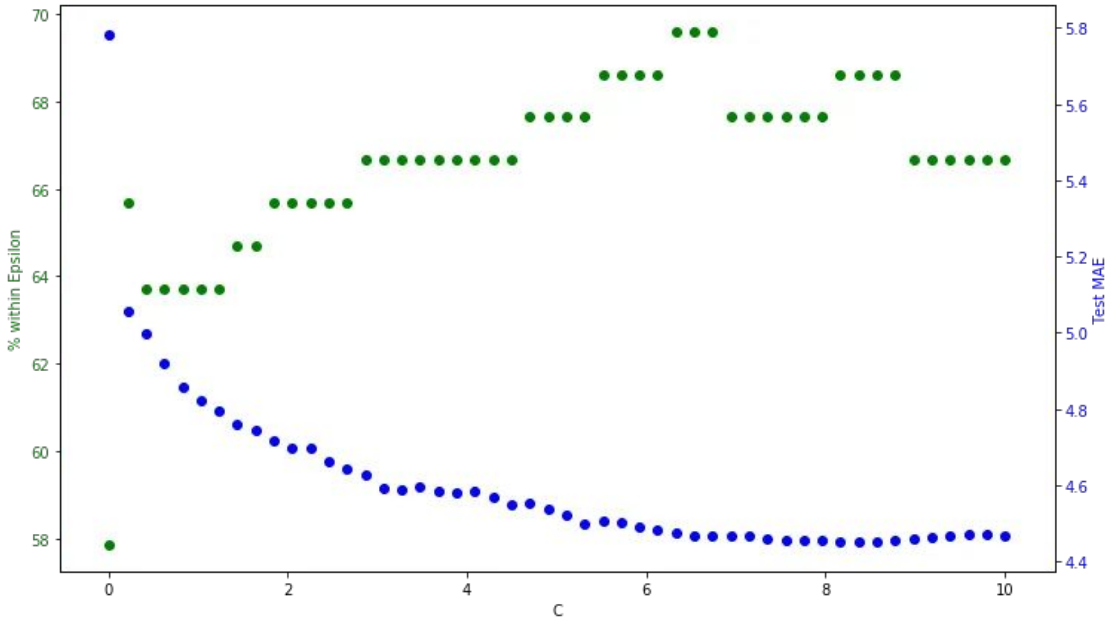
Margin of error

Loss function for the SVR

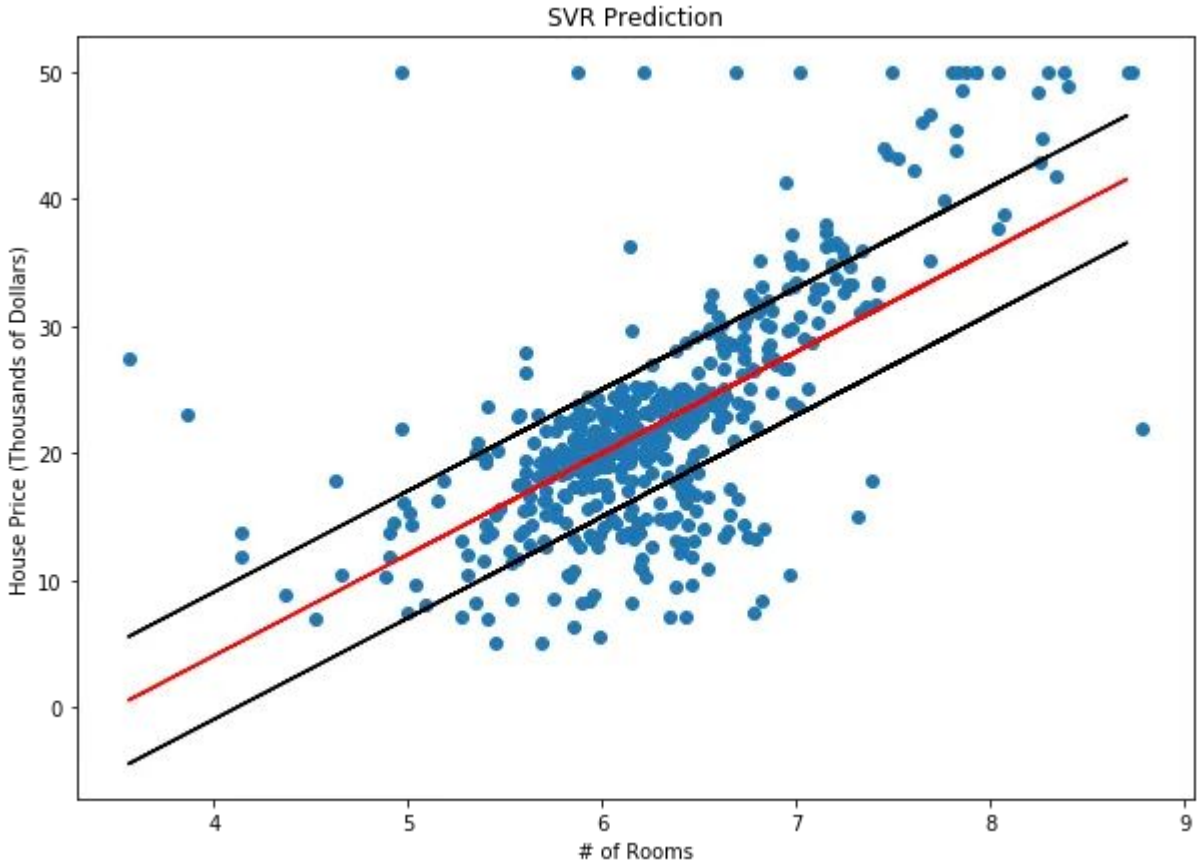
$$\min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} |\xi_i|$$

Example: House price in Boston

- We can use grid search over *C* to find the ideal amount of slack (more points within margin).
- Since our original objective of this model was to maximize the prediction within our margin of error ($5,000), we want to find the value of *C* that maximizes *% within Epsilon*. Thus, *C*=6.13.



ϵ=5, C=6.13



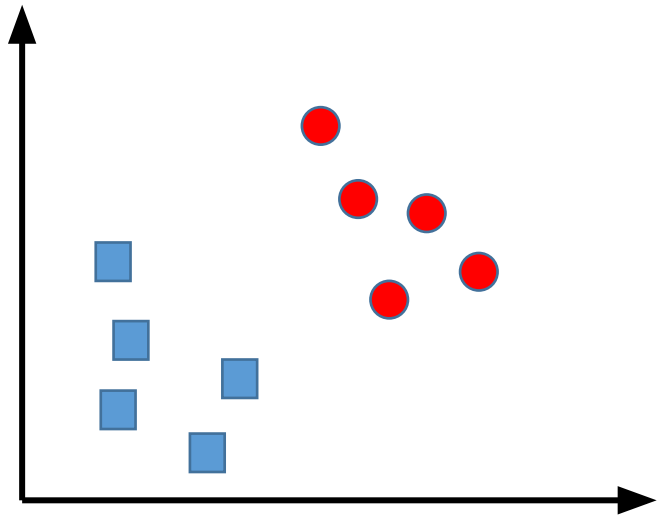
Support Vector Machine for Regression

- 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.

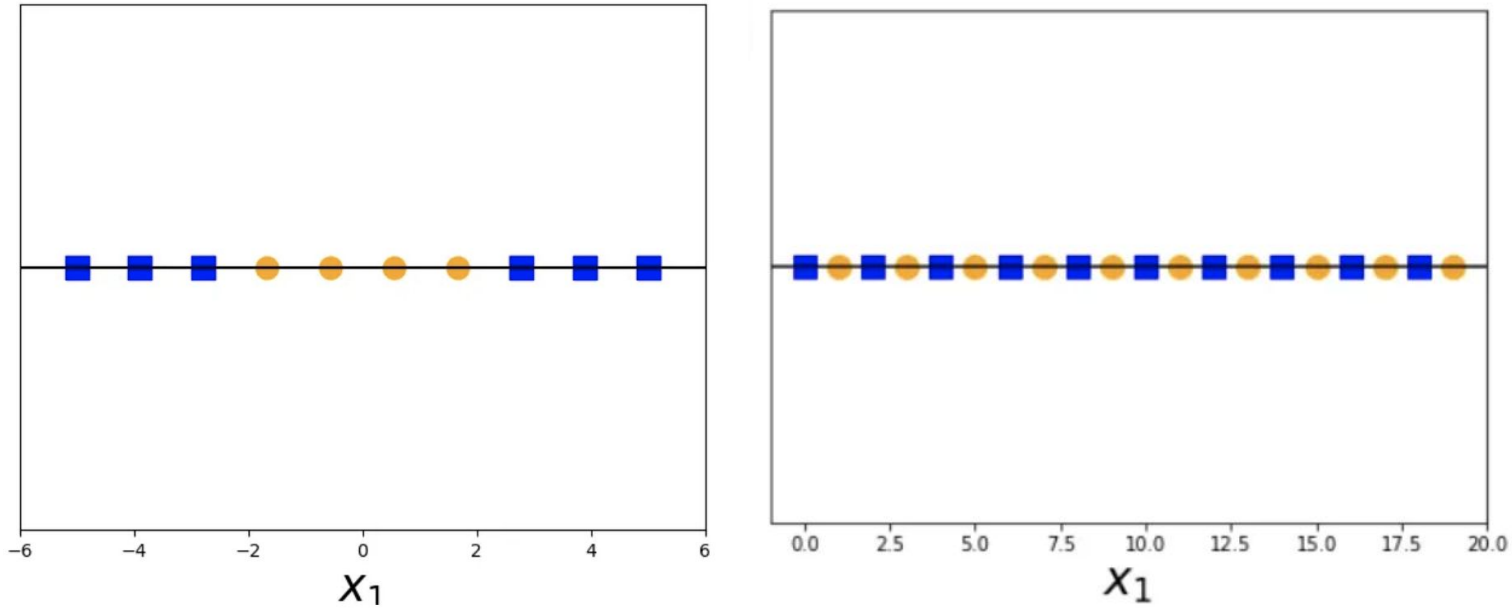
What if…

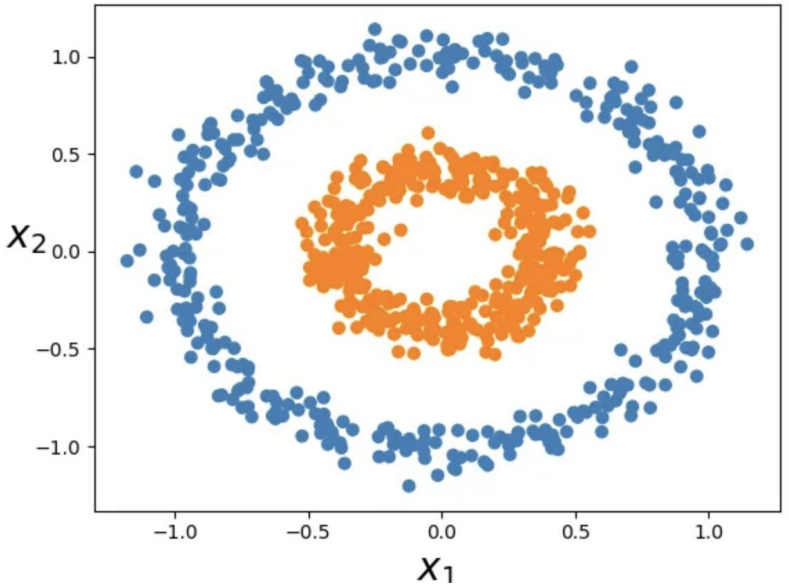
Non-linear spaces

Linearly separable

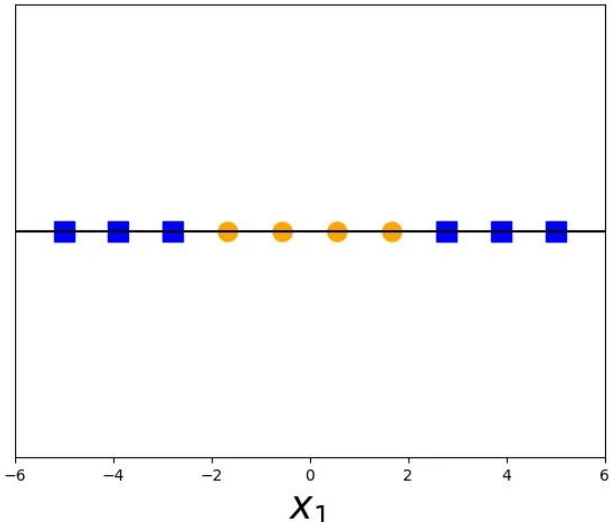


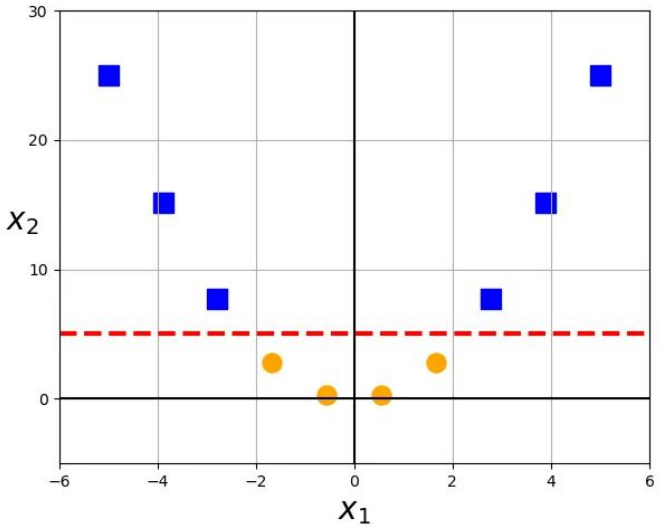
Not linearly separable





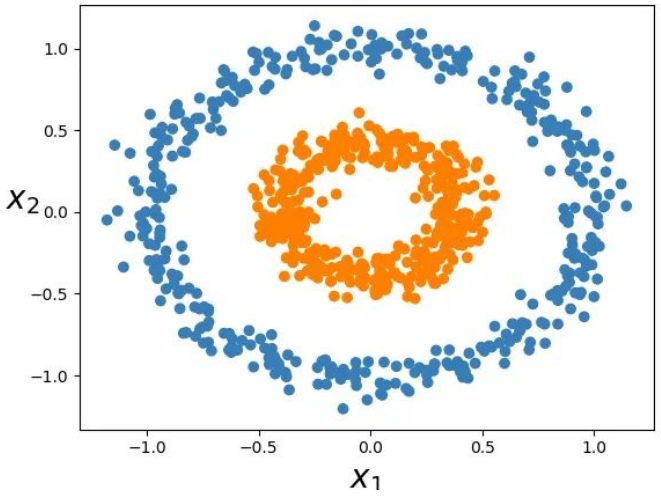
Kernel tricks

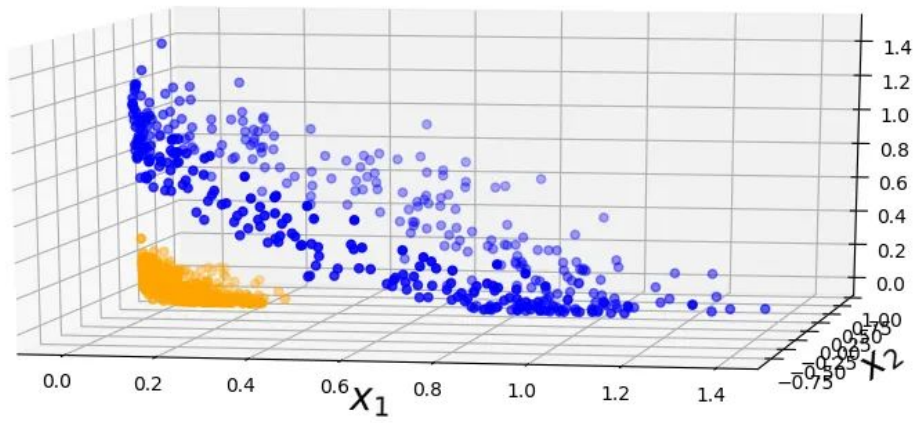




“Give me enough dimensions and I will classify the whole world”.

Zucker, Steve





Time for a quiz and tutorial!

https://tinyurl.com/geocomp2025

Intro to optimization

Review on Linear Regression

Task (T)

Input $x \in \mathbb{R}^n$

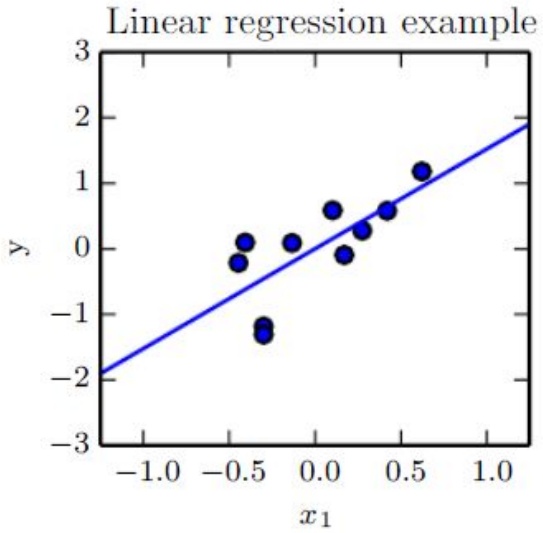
Weights $w \in \mathbb{R}^n$

$$\hat{y} = w^T x$$

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

Performance (P)

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

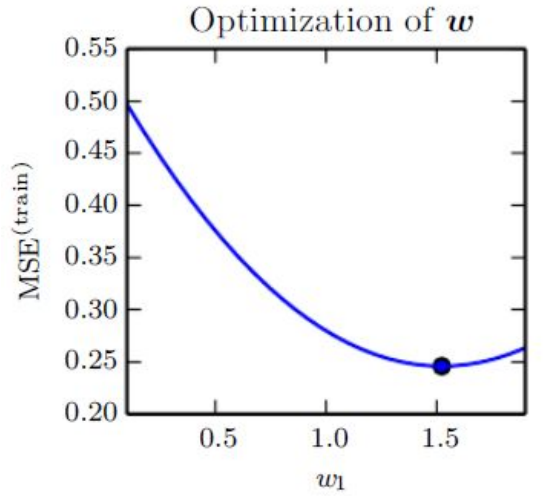


Dataset

$$(X, y) \longrightarrow \begin{cases} (X_{train}, y_{train}) \\ (X_{test}, y_{test}) \end{cases}$$

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)

Linear Regression Optimization

- Add an offset w_0: $f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x} + \mathrm{w}_0$, $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_n, y_n)\}$

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

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

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

Mean squared error loss

Rewrite:

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

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

Regularization

- Ridge regression: penalize with L2 norm

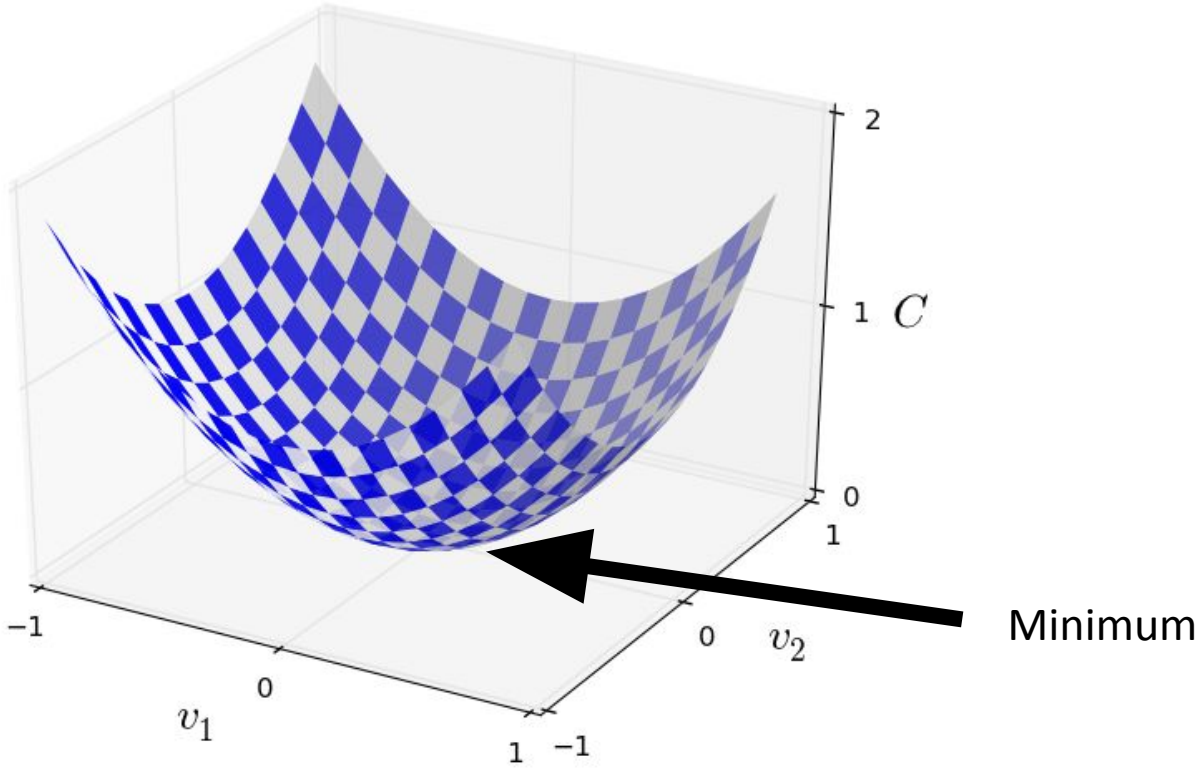
$$\boldsymbol{w}^* = \arg\min \sum_i L(f(\boldsymbol{x}_i; \boldsymbol{w}), y_i) + \lambda \sum_{j=1}^{m} w_j^2$$

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

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

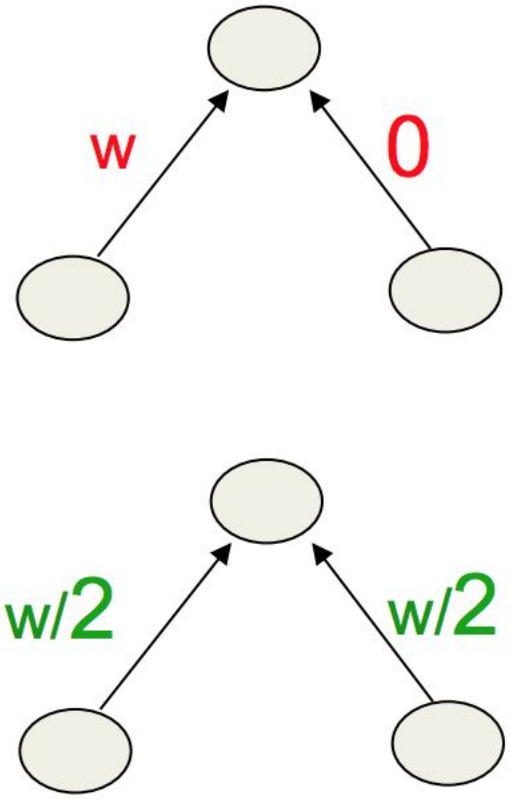
  - No closed form solution but still convex (optimal solution can be found)

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!

Regularization

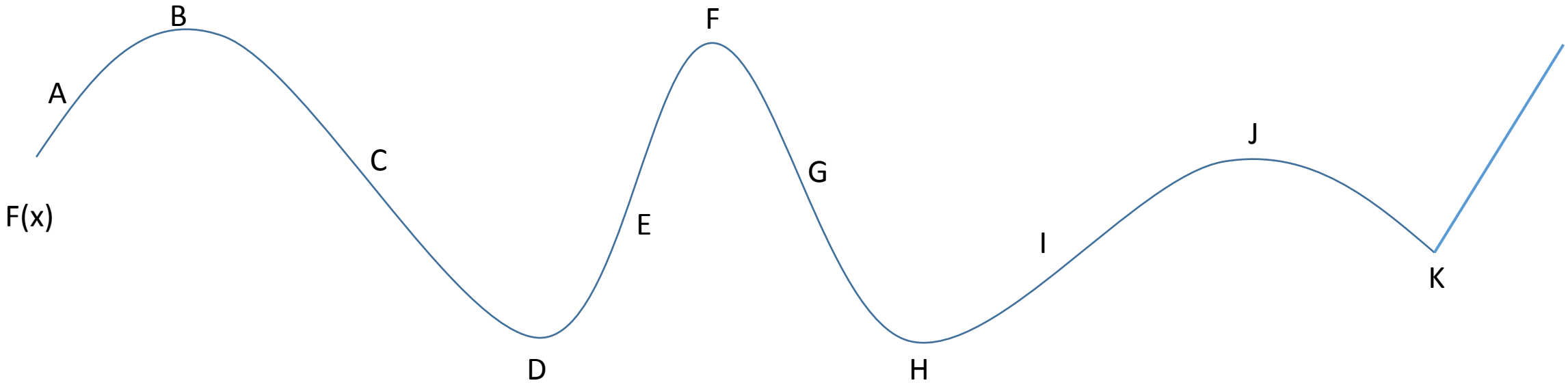


- Prefers to share smaller weights
- Makes model smoother
- More Convex

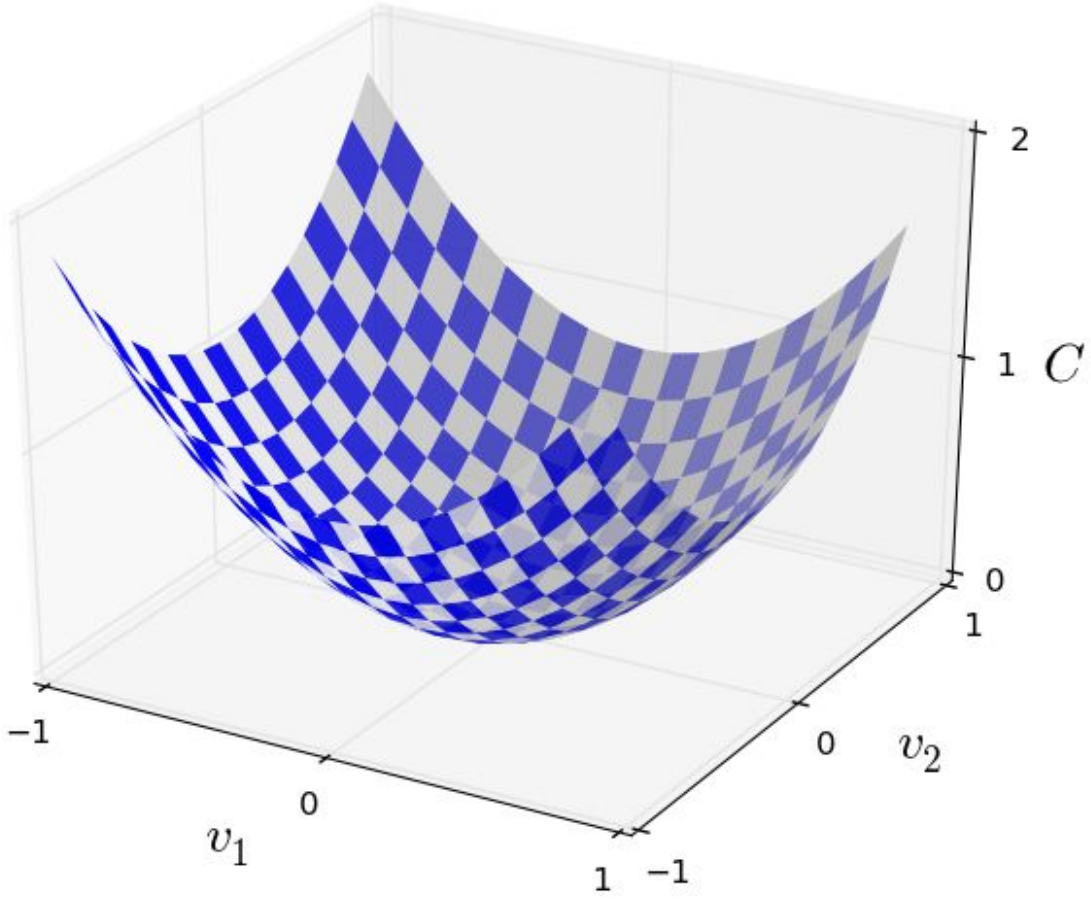
More on the derivatives

Join by QR code
Scan with your camera app

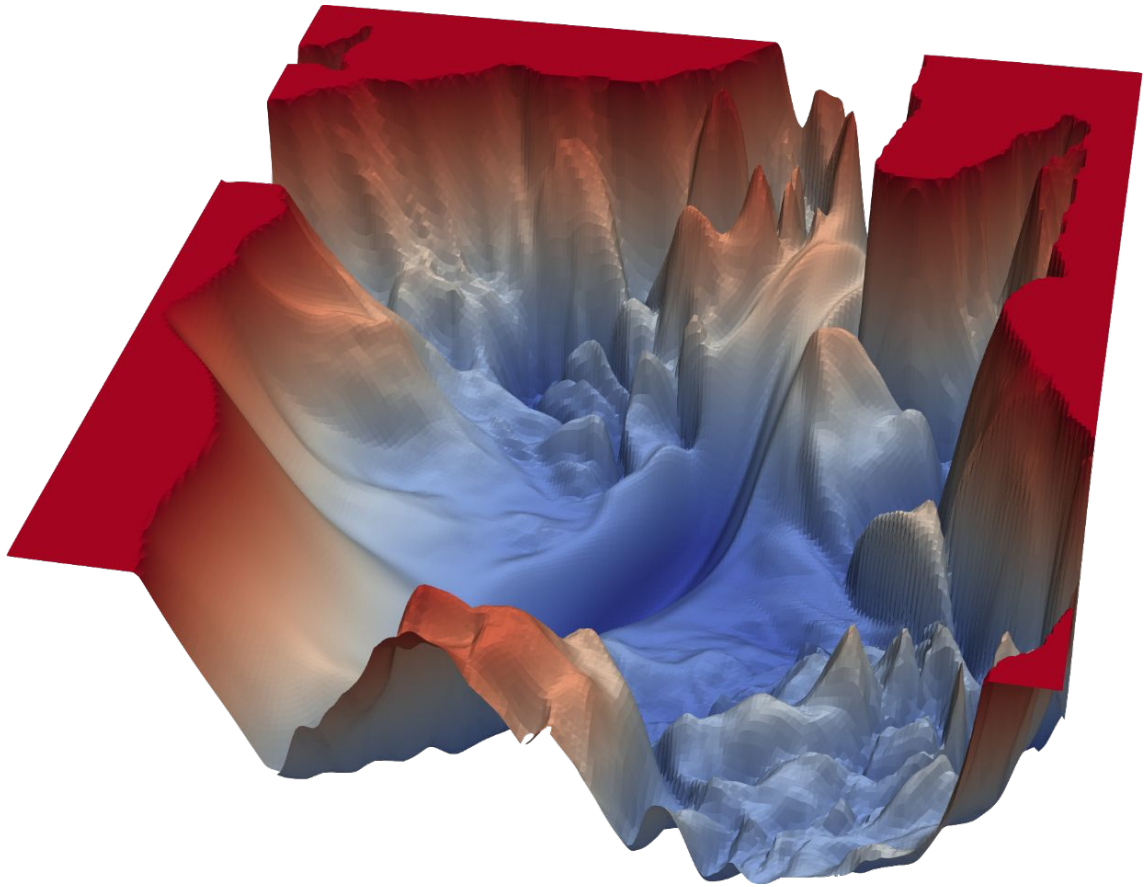
https://tinyurl.com/geocomp2025



Expectation

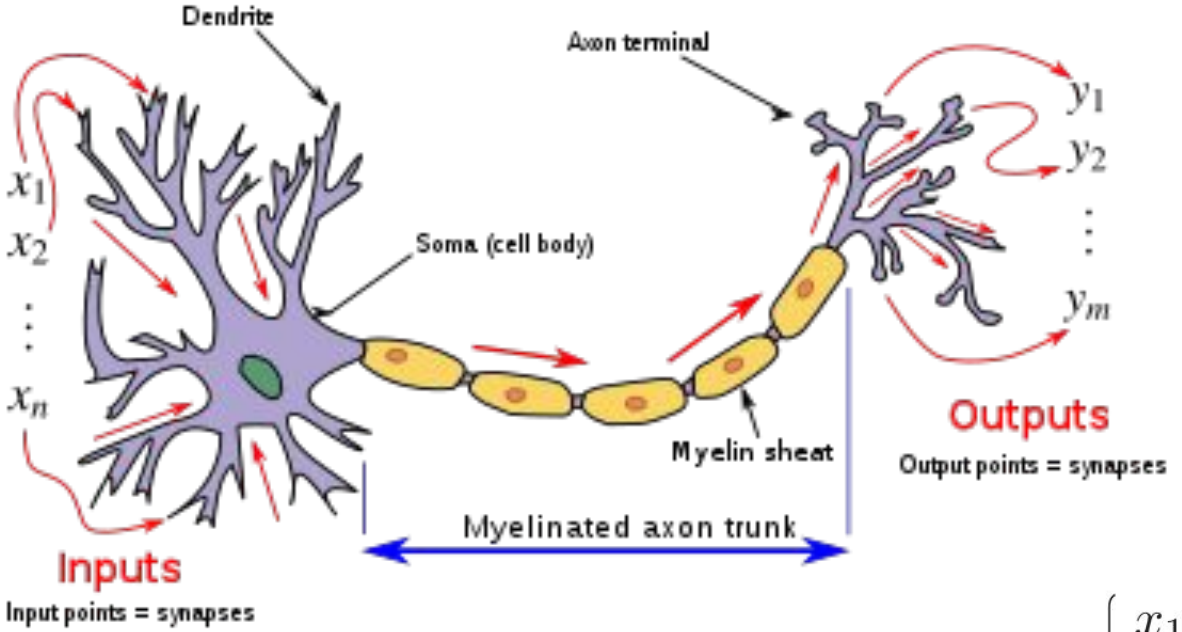


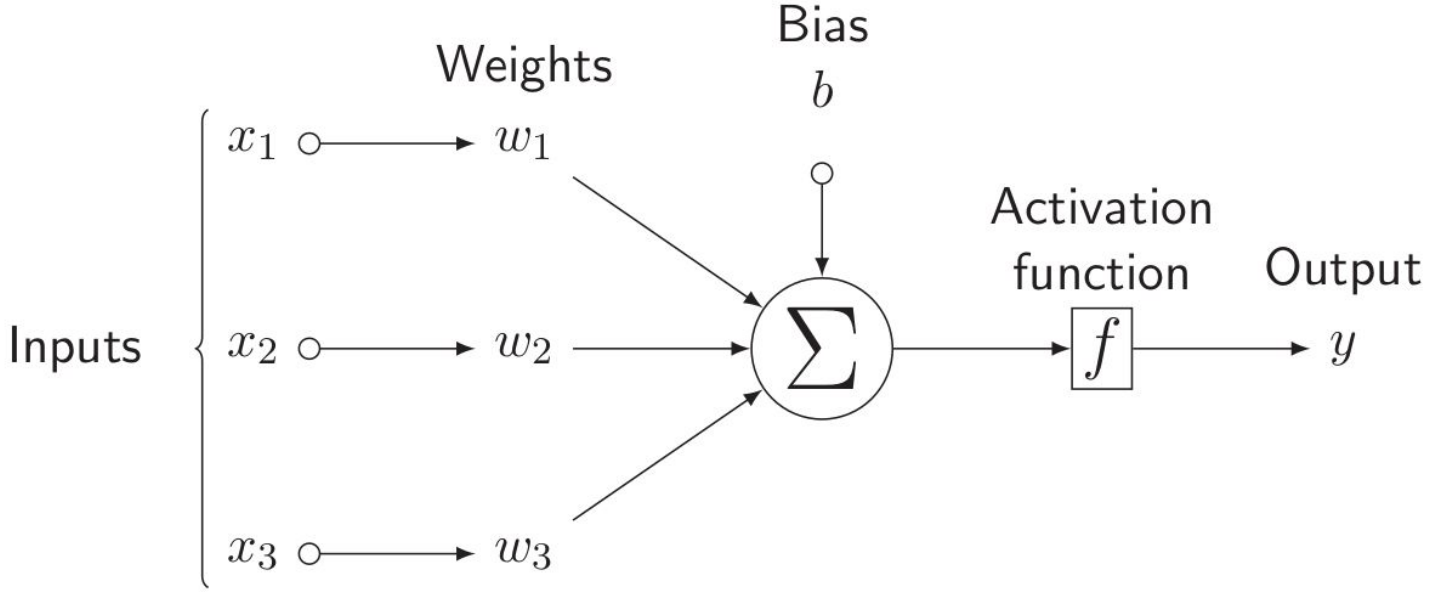
Reality



Perceptron

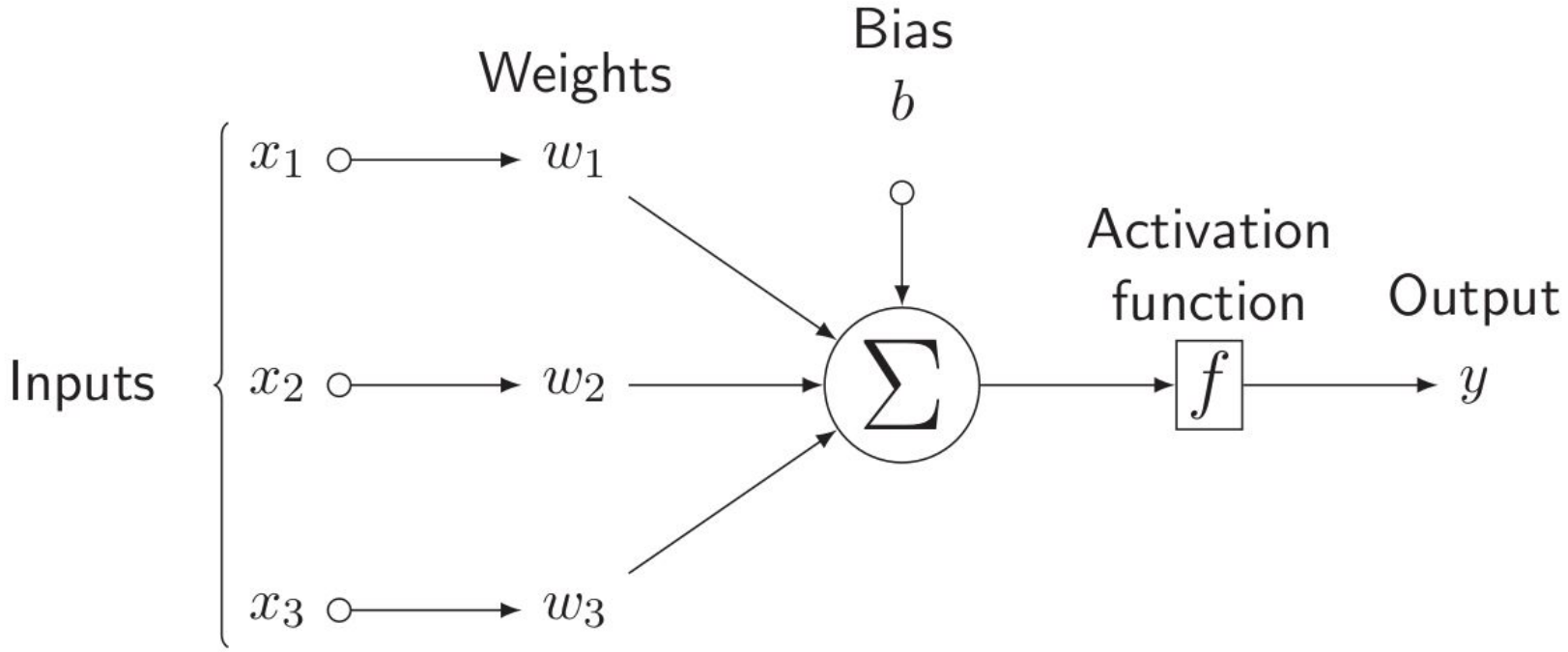
Perceptron: Threshold Logic





Perceptron: Threshold Logic

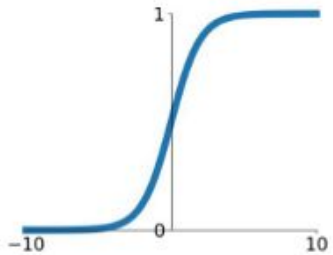
$$\mathcal{L}_{\text{perc}}(\mathbf{x}, y) = \begin{cases} 0 & \text{if } y\mathbf{w}^\top \mathbf{f}(\mathbf{x}) > 0 \\ -y\mathbf{w}^\top \mathbf{f}(\mathbf{x}) & \text{if } y\mathbf{w}^\top \mathbf{f}(\mathbf{x}) \leq 0 \end{cases}$$



Activation functions

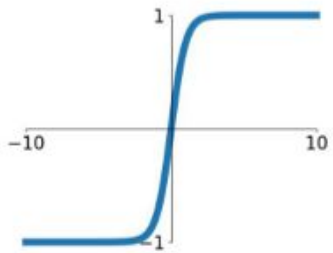
Sigmoid

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



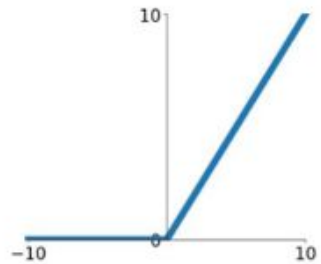
tanh

$$\tanh(x)$$



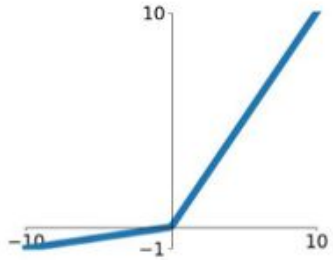
ReLU

$$\max(0, x)$$



Leaky ReLU

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

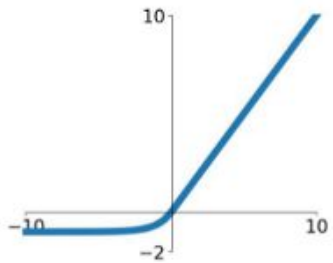


Maxout

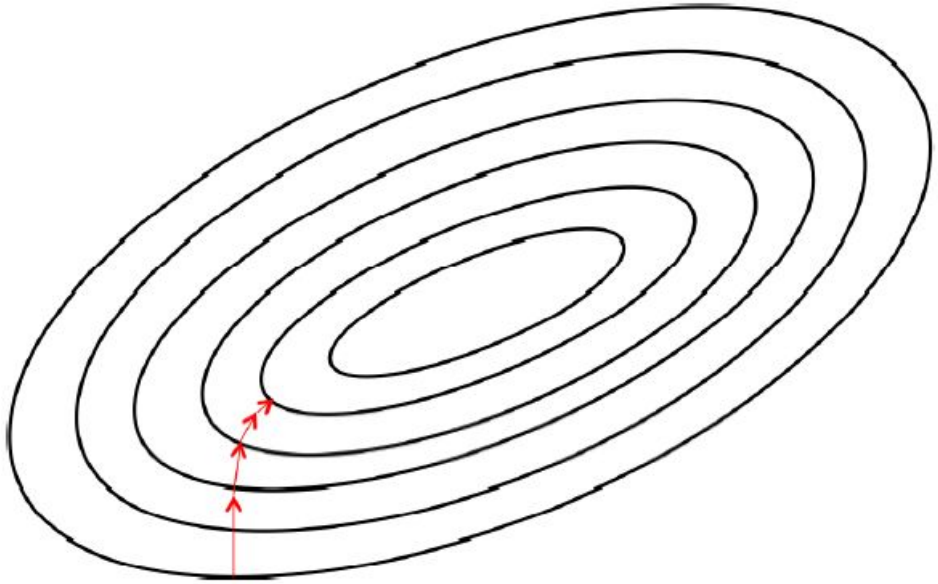
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

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



Optimizers

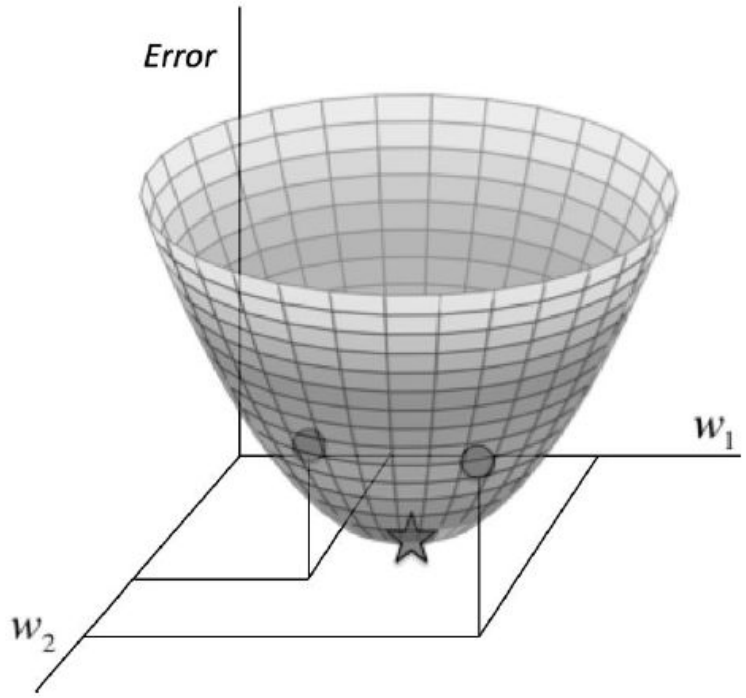


Gradient

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

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

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



Stochastic gradient descent (**SGD**)

Optimizers

Hyperparameters

- Learning rate (α)

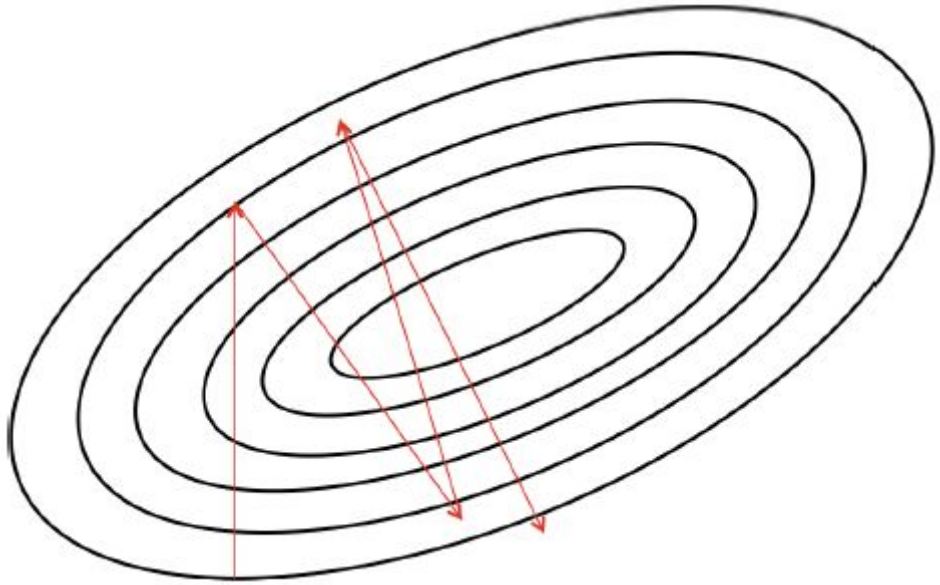
$$\Delta w_k = -\alpha \frac{\partial E}{\partial w_k}$$

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

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

Stochastic gradient descent (**SGD**)

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 α

Optimizers

Watch out for local minimal areas

Hyperparameters

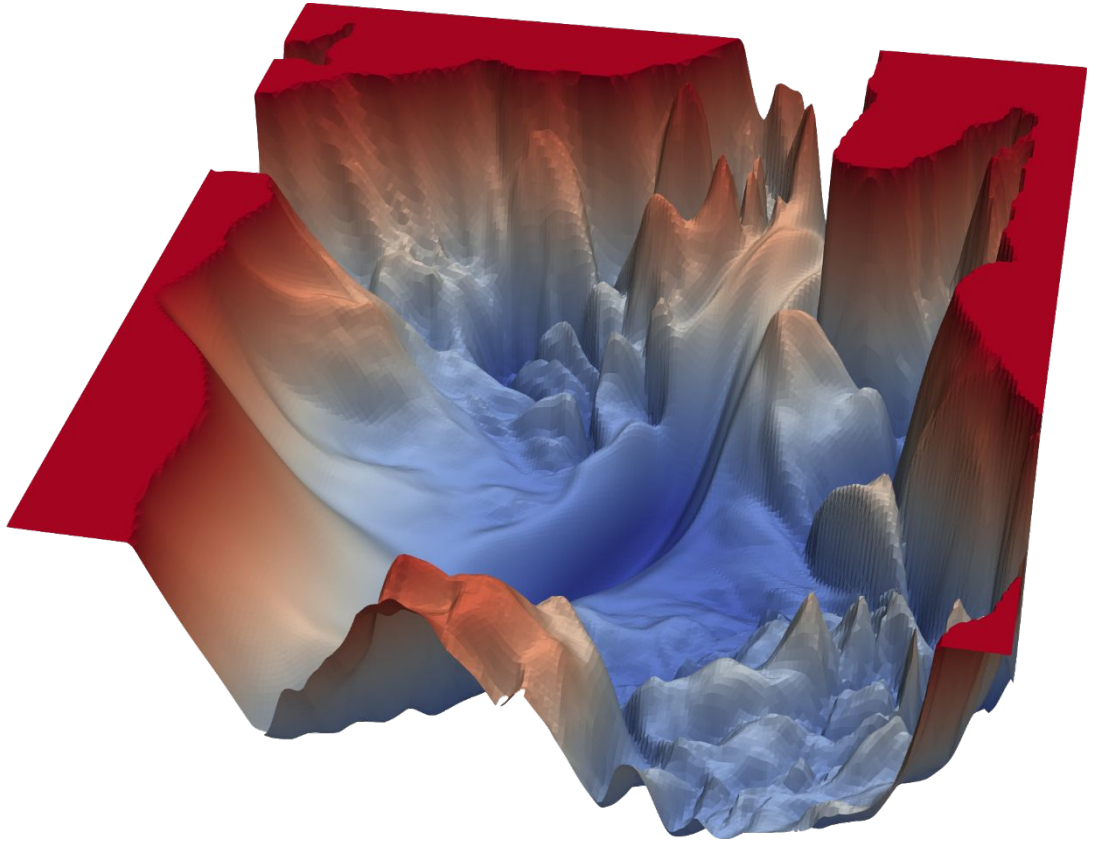
- Learning rate (α)

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

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

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

Stochastic gradient descent (**SGD**)



Gradient Descent

- Gradient descent refers to taking a step in the direction of the ***gradient (partial derivative)*** of a weight or bias with respect to the loss function
- Gradients are propagated backwards through the network in a process known as ***backpropagation***
- The size of the step taken in the direction of the gradient is called the ***learning rate***

Time for a quiz and tutorial!

https://tinyurl.com/geocomp2025