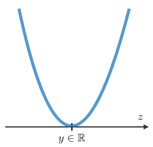
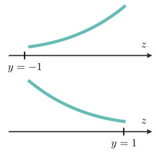
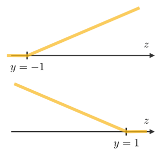
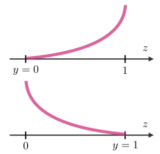
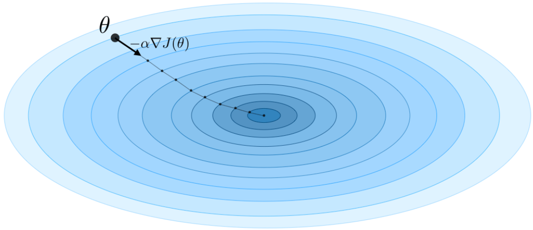
Moindres carrés	**Logistique**	**Hinge loss**	**Cross-entropie**
$\frac{1}{2}(y-z)^2$	$\log(1+\exp(-yz))$	$\max(0,1-yz)$	$-\Big[y\log(z)+(1-y)\log(1-z)\Big]$
			
Régression linéaire	Régression logistique	SVM	Réseau de neurones

❒ Fonction de coût – La fonction de coût J est communément utilisée pour évaluer la performance d'un modèle, et est définie avec la fonction de loss L par :

$$\boxed{J(\theta)=\sum_{i=1}^m L(h_\theta(x^{(i)}),y^{(i)})}$$

❒ Algorithme du gradient – En notant $\alpha\in\mathbb{R}$ le taux d'apprentissage (en anglais *learning rate*), la règle de mise à jour de l'algorithme est exprimée en fonction du taux d'apprentissage et de la fonction de cost J de la manière suivante :

$$\boxed{\theta\longleftarrow\theta-\alpha\nabla J(\theta)}$$



Remarque : L'algorithme du gradient stochastique (en anglais SGD - Stochastic Gradient Descent) met à jour le paramètre à partir de chaque élément du jeu d'entrainement, tandis que l'algorithme du gradient de batch le fait sur chaque lot d'exemples.

❒ Vraisemblance – La vraisemblance d'un modèle $L(\theta)$ de paramètre θ est utilisée pour trouver le paramètre optimal θ par le biais du maximum de vraisemblance. En pratique, on utilise la log vraisemblance $\ell(\theta)=\log(L(\theta))$ qui est plus facile à optimiser. On a :

$$\boxed{\theta^{\textrm{opt}}=\underset{\theta}{\textrm{arg max }}L(\theta)}$$

❒ Algorithme de Newton – L'algorithme de Newton est une méthode numérique qui trouve θ tel que $\ell'(\theta)=0$. La règle de mise à jour est :

$$\boxed{\theta\leftarrow\theta-\frac{\ell'(\theta)}{\ell''(\theta)}}$$