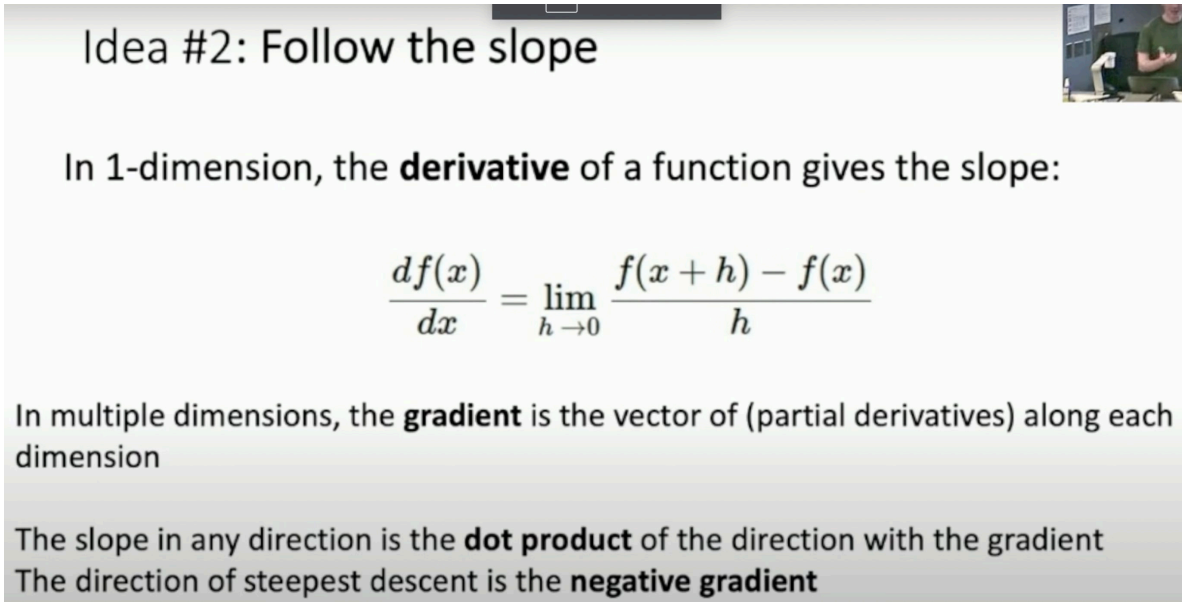
卡片 1 正面

课程笔记--优化算法

$$w^* = \arg\min_w L(w)$$

这个方程式 w*=arg minwL(w) 是一个优化方程，它可以找到 L(w) 的最小值，而 L(w) 是成本或误差函数。 参数 w 是函数的参数，*表示我们正在寻找能够使得成本 / 误差函数最小的 w 的值



partial derivative 偏导数

任何方向的斜率都是方向与梯度的点积

点积 dot product

（也称向量积、内积、数量积）是广义上多个实数或向量之间的乘积，是将两个向量进行乘积操作，使得一个向量被另一个向量线性变换而得到新向量。它和外积有区别，点积只考虑空间中的模长大小，而不关心正负，而外积考虑正负的关系。

Numeric gradient

数值梯度，它指的是函数的数值导数。它被用于计算一个变量相对于另一个变量的

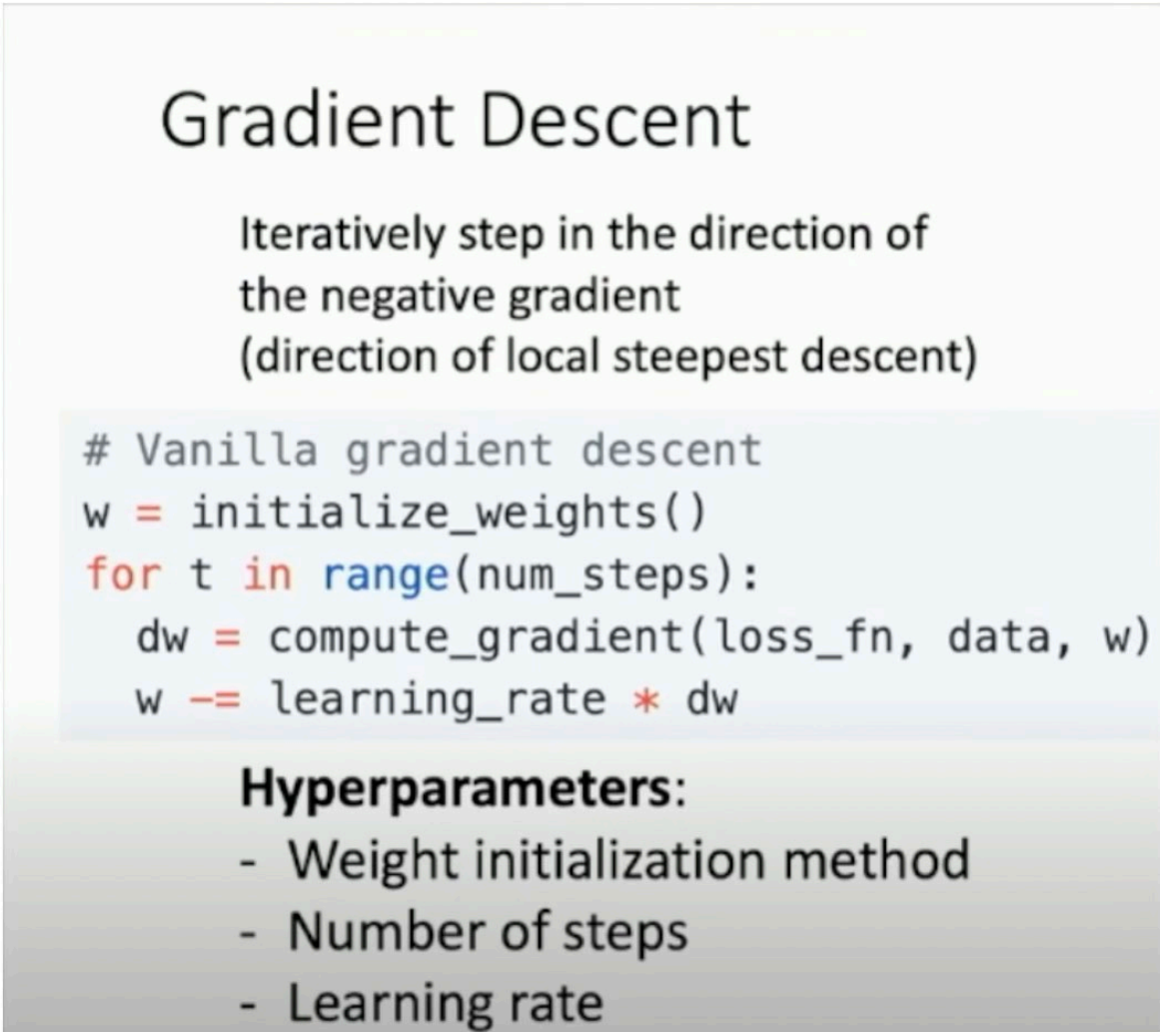
变化率。它也被用于机器学习算法来识别趋势并做出更好的预测。

Numeric Gradient:
- Slow: O(#dimensions)
- Approximate

Computing Gradients

- **Numeric gradient**: approximate, slow, easy to write
- **Analytic gradient**: exact, fast, error-prone

In practice: Always use analytic gradient, but check implementation with numerical gradient. This is called a **gradient check.**

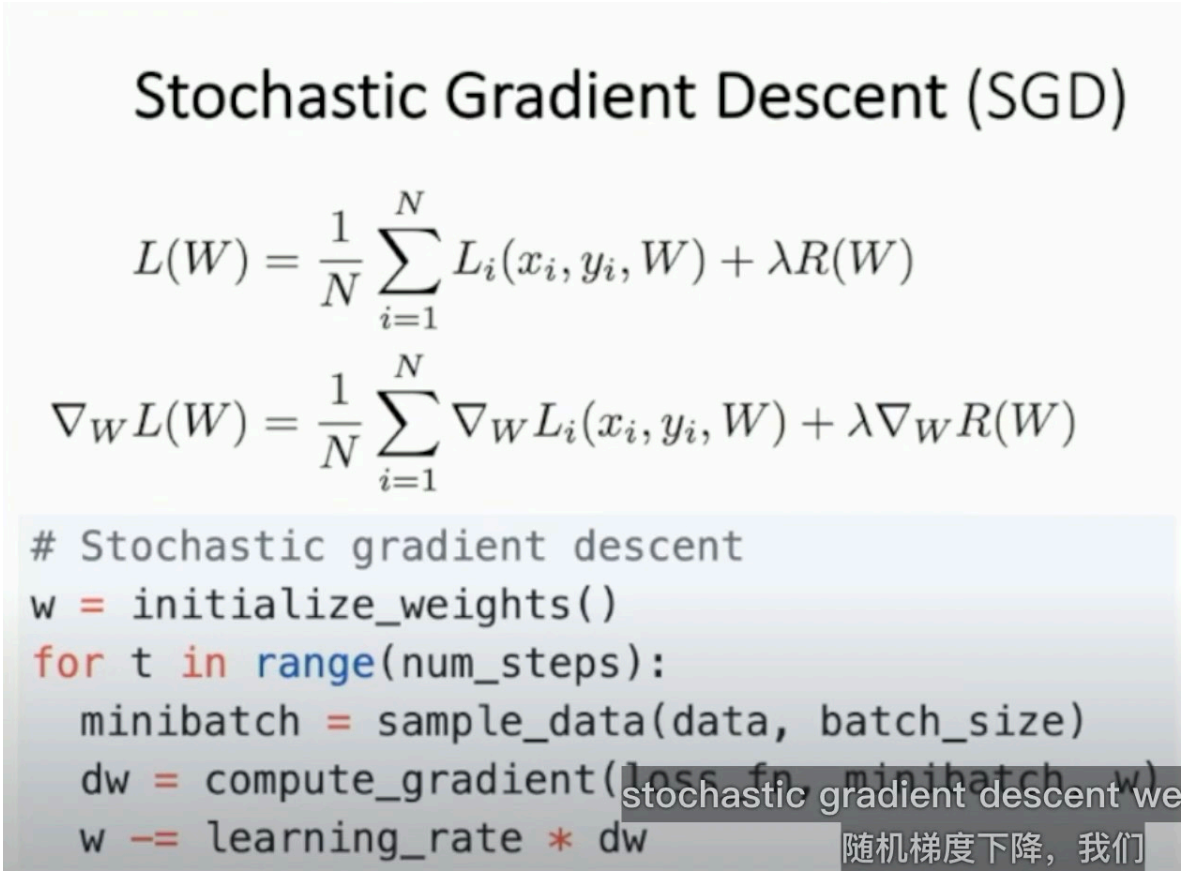


这三个超参数很重要‼️

学习率、动量和衰减率。

学习率控制修正步长的大小，动量控制对当前梯度信息的响应程度，而衰减率则决定未来步长的大小。

SGD是Stochastic Gradient Descent的缩写，它是一种迭代优化方法，通常用于深度学习中训练神经网络。它使用梯度下降法来更新每个参数，以最小化目标函数。



在这里引入Batch size这个超参数，不管参数的尺寸多大，都不影响模型效率。

另一个超参数是data sampling。对于许多图像分类问题而言，迭代数据的方式并非是一种重要的超参数。

梯度下降的两个潜在问题：

梯度下降的两个潜在问题是鞍点和逃逸局部最优解。鞍点是函数中的特殊点，其偏导数为零，梯度下降可能无法正常工作并进入局部最小值。此外，当使用梯度下降计算线性模型最佳参数时，可能会遭遇逃逸局部最 add 优解，即模型可能会进入另一个局部最小值，而不是全局最优值。 The two potential problems of gradient descent are saddle points and escaping from local optima. Saddle points are special points in a function where its partial derivatives equal to zero, and gradient descent may not work properly and get stuck in the local minimum. Additionally, when gradient descent is used to compute the optimal parameters for a linear model, one may encounter escaping from the local optima, meaning that the model may get into another local minimum instead of the global optimum.

解决方法是：SGD+momentum ---->问题是怎么➕动量

SGD + Momentum

SGD

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

```
for t in range(num_steps):
  dw = compute_gradient(w)
  w -= learning_rate * dw
```

SGD+Momentum

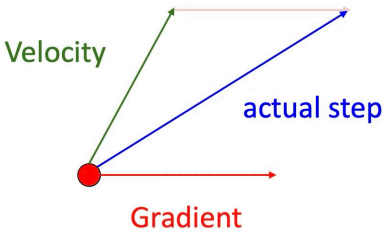
$$v_{t+1} = \rho v_t + \nabla f(x_t)$$
$$x_{t+1} = x_t - \alpha v_{t+1}$$

```
v = 0
for t in range(num_steps):
  dw = compute_gradient(w)
  v = rho * v + dw
  w -= learning_rate * v
```

- Build up “velocity” as a running mean of gradients
- Rho gives “friction”; typically rho=0.9 or 0.99

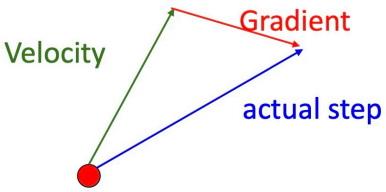
Nesterov Momentum

Momentum update:



Combine gradient at current point with velocity to get step used to update weights

Nesterov Momentum



“Look ahead” to the point where updating using velocity would take us; compute gradient there and mix it with velocity to get actual update direction

Nesterov, “A method of solving a convex programming problem with convergence rate O(1/k^2)”, 1983
Nesterov, “Introductory lectures on convex optimization: a basic course”, 2004
Sutskever et al, “On the importance of initialization and momentum in deep learning”, ICML 2013

另一种常见优化算法：这是一个深度学习常见的优化算法

Adam (almost): RMSProp + Momentum

```
moment1 = 0
moment2 = 0
for t in range(num_steps):
  dw = compute_gradient(w)
  moment1 = beta1 * moment1 + (1 - beta1) * dw
  moment2 = beta2 * moment2 + (1 - beta2) * dw * dw
  w -= learning_rate * moment1 / (moment2.sqrt() + 1e-7)
```

Optimization Algorithm Comparison

Algorithm	Tracks first moments (Momentum)	Tracks second moments (Adaptive learning rates)	Leaky second moments	Bias correction for moment estimates
SGD	✗	✗	✗	✗
SGD+Momentum	✓	✗	✗	✗
Nesterov	✓	✗	✗	✗
AdaGrad	✗	✓	✗	✗
RMSProp	✗	✓	✓	✗
Adam	✓	✓	✓	✓

这些是一阶梯度上讨论的优化算法。

卡片 1 背面

学习中的一些思考和疑问

1. 线性回归和线性分类器的区别：

线性回归和线性分类器都属于机器学习中的监督学习，具有相似的数学原理，但实际上它们的使用和应用有所不同。线性回归用于预测连续数据，如房地产销售价格或股票价格的趋势，而线性分类器用于预测离散类别，例如垃圾邮件过滤或识别图像中的对象。此外，线性回归的输出是一个连续的数值，而线性分类器的输出是一系列类别中的一个或多个，例如垃圾邮件或正常邮件。

2. 权重初始化是在开始训练神经网络模型之前，将权重分配正确的过程。常用的权重初始化方法有随机初始化、零初始化和Xavier/He初始化。其中，随机初始化是使用一定范围内的随机数来对权重进行初始化，零初始化将所有权重设置为0，而Xavier/He初始化则采用平均分布和标准差进行初始化。

3. 超参数是指机器学习算法中不能从训练数据中直接学习的参数，例如学习率、正则化参数和决策树深度等。它们的取值有助于确定算法性能何其好坏。

4. 蒙特卡洛估计Monte Carlo Estimation是一种概率统计方法，可以根据抽样的随机数据来估计不可能直接计算的量。它通常用于分析复杂的系统，并可用于模拟

和预测。

5. 全局最优值是什么

全局最优值是指函数可以达到的最大或最小值。例如，某个函数的全局最小值是函数可能达到其最低水平时所体现的浮点。在梯度下降中，全局最优解表示要查找到该函数可以实现的最大化或最小化优化值。

6. 如果梯度下降过快在某一方向而在其他方向下降过慢，会发生什么？

如果在某一方向上使用梯度下降过快，而在其他方向上慢，可能会导致最终达不到全局最优解，而仅能达到局部最优解。这是因为，如果梯度下降过快，就容易跳过优化函数全局最小值，从而只能达到该函数的局部最小值。