**Báo cáo tiến trình học tập tuần 4**

1. **Entropy, Cross Entropy và KL Divergence**
2. **Entropy**

* Entropy như là một độ đo để định lượng thông tin hay độ bất định/không chắc chắn của thông tin.

**Ví dụ:**

* Vào mùa đông, xác suất ngày nắng (nhẹ) là 100%, mưa là 0%. Như vậy dựa vào xác suất đó, mùa đông ta hoàn toàn tự tin đoán được ngày mai mưa hay nắng. Trong trường hợp này, độ chắc chắn cao, sáng mai ngủ dậy, trời không có mưa thì ta cũng không thấy bất ngờ, tức thông tin tiếp nhận đó không có nhiều giá trị.
* Vào mùa hè, xác suất một ngày không mưa là 50%, có mưa là 50%. Như vậy xác suất mưa/nắng là như nhau. Ta rất khó đoán xem ngày mai mưa hay nắng, độ chắc chắn là không cao. Nói cách khác, ngày mai có mưa hay nắng đều làm ta bất ngờ hơn trong trường hợp 1.

1. Mã hóa thông tin

* Để tối ưu trong việc truyền tin người ta mã hóa bằng các bit 0, 1.
* Thông tin được mã hóa trước khi truyền đi, người nhận sẽ nhận được các thông tin đã bị mã hóa, cần tiến hành giải mã để được thông tin gốc ban đầu.

1. Công thức tính entropy

**Kích cỡ mã hóa trung bình**:

Với là số bit cần dùng để mã hóa tín hiệu có xác suất .

* Gọi N là số loại tin nhắn khác nhau, n là số bít cần dùng để mã hóa, ta có:
* Giả sử các tin nhắn này đều có tần suất xuất hiện như nhau là p = 1/N.
* Kích cỡ mã hóa trung bình tối thiểu:

1. **Cross Entropy**

* Entropy của phân phối này có phân phối P nhưng lại được mã hóa dựa vào phân phối Q:

Tính chất:

* Đạt giá trị minimum nếu P=Q
* Rất nhạy cảm với sự sai khác giữa *pi*​ và *qi*​. Khi *pi*​ và *qi*​ càng khác nhau, giá trị cross\_entropy càng tăng nhanh.

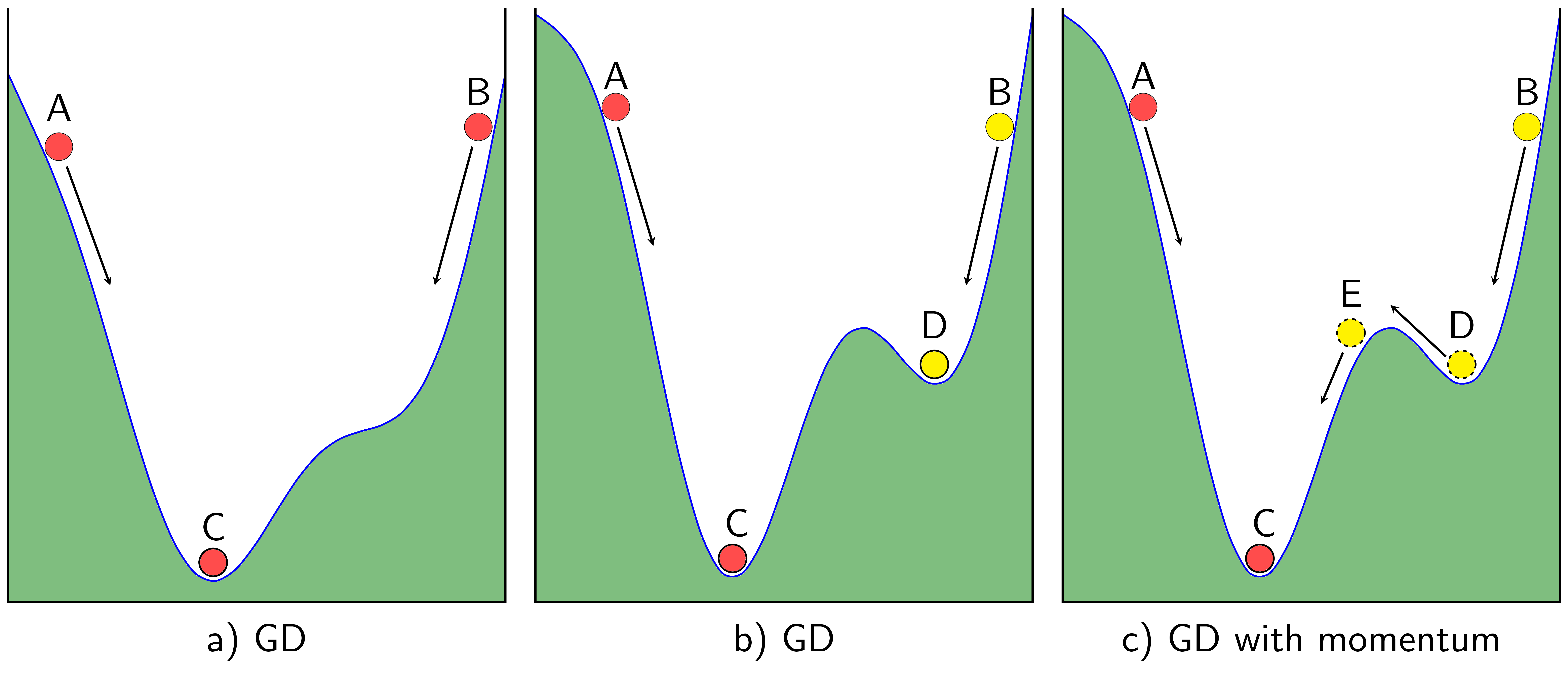
1. **KL divergence**

* Người ta muốn xác định xem họ đã *tốn*, *tổn thất* thêm bao nhiêu bit khi mã hóa dựa vào Q thay vì dùng P. Ta có:

Tính chất:

* Không có tính đối xứng, tức
* Tương tự CrossEntropy, DL cũng rất nhạy cảm với sự sai khác giữa *pi*​ và *qi.*​

1. **Các thuật toán tối ưu**
2. **Momentum**



* Ý tưởng là làm thế nào cho hòn bi đi xuống được C mà không dừng lại ở D, như trong vật lý, nếu vận tốc đủ lớn, thì khi lăn đến D, nó đang có đà và vượt qua rãnh D để tiếp tục đến C (đây là điều chúng ta mong muốn) => Momentum dựa trên nguyên lí này.
* Nếu chúng ta coi đại lượng thay đổi ở thời điểm t như là vận tốc trong vật lý, thì vị trí mới của hòn bi sẽ là :.
* vừa mang thông tin độ dốc, vừa mang thông tin của đà, tức vận tốc trước đó (chúng ta coi như vận tốc ban đầu = 0, nên ta có:
* Trong đó thường được chọn là một khoảng 0.9, vị trí mới của hòn bi được xác định như sau:

1. **Adagrad**

* Adagrad là thuật toán thay đổi learning rate, nó sẽ làm cho learning rate biến thiên sau mỗi thời gian t.

**Trong đó:**

**:** hằng số

**:** gradient tại thời điểm t

**:** hệ số tránh lỗi (chia cho mẫu bằng 0)

**:** là ma trận chéo mà mỗi phần tử trên đường chéo (i,i) là bình phương của đạo hàm vectơ tham số tại thời điểm t. **Ưu điểm:**

* Tránh việc điều chỉnh learning rate bằng tay, chỉ cần để tốc độ học default là 0.01 thì thuật toán sẽ tự động điều chỉnh.

**Nhược điểm:**

* Yếu điểm của Adagrad là tổng bình phương biến thiên sẽ lớn dần theo thời gian cho đến khi nó làm tốc độ học cực kì nhỏ, làm việc training trở nên đóng băng.

1. **RMSprop**

* RMSprop giải quyết việc learning rate giảm dần của Adagrad bằng cách chia cho tỉ lệ học trung bình của bình phương gradient:
* Khác với Adagrad là cộng dồn khiến cho việc cập nhật gradient lâu dẫn đến việc cập nhật bị đóng băng do lượng tích lũy lớn.
* RMSprop cải tiến bằng cách làm mờ dần quá khứ, như công thức là một hàm đệ quy, việc cộng dồn nhưng nhân với một hệ số gammar làm cho các hệ số được cộng vào từ lâu không còn có ảnh hưởng nữa, như thế giải quyết được vấn đề đóng băng nếu gradient xảy ra lâu.

1. **Adam**

* Adam là sự kết hợp của Momentum và RMSprop.
* Adam như 1 quả cầu rất nặng có ma sát, vì vậy nó dễ dàng vượt qua local minimum tới global minimum và khi tới global minimum nó không mất nhiều thời gian dao động qua lại quanh đích vì nó có ma sát nên dễ dừng lại hơn.
* Công thức chuẩn gồm 3 bước:

**Bước 1:** Tính trung bình động:

(giống momentum)

(giống RMSprop)

**Bước 2:** Hiệu chỉnh thiên vị: giúp đẩy giá trị cho m và v cho những vòng lặp đầu tiên lớn lên, tránh thiên vị về 0.

**Bước 3:** Cập nhật

* Các giá trị khởi tạo:

1. **Bias, variance và tradeoff**
2. **Bias (Độ chệch)**

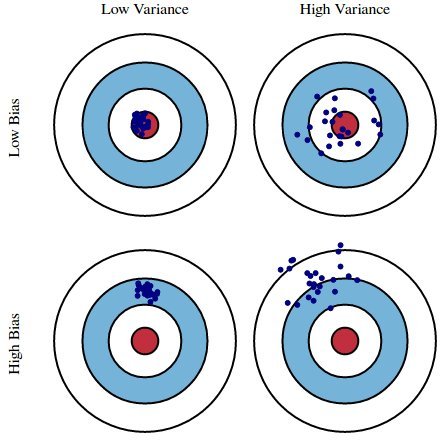
* **Độ chệch** là sai khác giữa giá trị dự báo và giá trị ground truth của một mô hình.
* Thông thường những mô hình **quá đơn giản** được huấn luyện trên những bộ dữ liệu **lớn** sẽ dẫn tới độ chệch lớn.
* Vì vậy trong tình huống này để giảm bớt độ chệch thì chúng ta thường sử dụng mô hình **phức tạp hơn** để tận dụng khả năng biểu diễn tốt hơn của chúng trên những tập dữ liệu kích thước lớn.

1. **Variance (Phương sai)**

* **Phương sai**được hiểu là hiện tượng mô hình của bạn dự báo ra giá trị có mức độ dao động lớn nhưng thiếu tổng quát.
* Yếu tố thiếu tổng quát được thể hiện qua việc giá trị dự báo có thể khớp tốt mọi điểm trên tập huấn luyện nhưng rất **dao động** xung quanh giá trị ground truth trên tập huấn luyện.
* Những lớp mô hình **phức tạp** được huấn luyện trên tập huấn luyện **nhỏ** thường xảy ra hiện tượng phương sai cao và dẫn tới việc học giả mạo thông qua bắt chước dữ liệu hơn là học qui luật tổng quát.

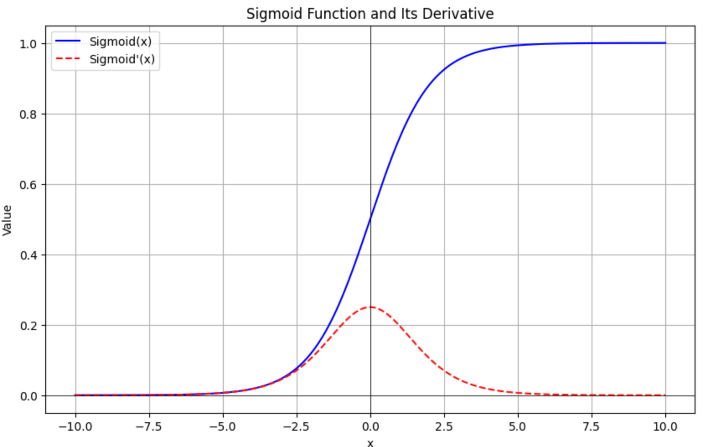
1. **Tradeoff (Đánh đổi)**

* Chúng ta luôn chấp nhận một sai số nhiễu giữa hàm f(x) và giá trị thực ground truth y, đây là phần sai số ngẫu nhiên không thể tránh khỏi.
* Công thức **Tổng bình phương sai số** giữa và giá trị thực tế y:
* Thành phần phương sai nhiễu có độ lớn không đáng kể nên ta có thể xem như **tổng bình phương sai số** chỉ phục thuộc phần lớn vào *độ chệch* và *phương sai*. Sự đánh đổi giữa *độ chệch* và *phương sai* thể hiện qua đối với các mô hình có cùng mức độ sai số, nếu muốn một mô hình dự báo ít chệch hơn thì sẽ cần phương sai lớn hơn và ngược lại.



1. Vanishing gradient problem (vấn đề biến mất độ dốc)

* Với các activation như hàm sigmoid, thì đạo hàm của nó có giá trị tối đa là 0.25:



* Khi huấn luyện mạng nhiều tầng, gradient truyền ngược sẽ chứa tích nhiều đạo hàm của hàm kích hoạt, điều này dẫn đến gradient gần như biến mất.
* Giải pháp là chúng ta sử dụng hàm ReLu, có công thức là

f(x) = max(0,x).

* Nghĩa là việc chứa nhiều tích đạo hàm của hàm kích hoạt không còn làm cho gradient bị biến mất nữa.
* Tuy nhiên, đối với các đầu vào nhỏ hơn hoặc bằng 0, đạo hàm là 0, điều này có thể dẫn đến cái được gọi là vấn đề " **ReLU chết** " - khi các tế bào thần kinh ngừng cập nhật hoàn toàn nếu chúng liên tục nhận được các đầu vào không dương.
* Để giải quyết các hạn chế của nó, các biến thể như Leaky ReLU và Parametric ReLU đưa ra các gradient nhỏ đối với các đầu vào âm, giúp duy trì hoạt động của các tế bào thần kinh trong suốt quá trình đào tạo.