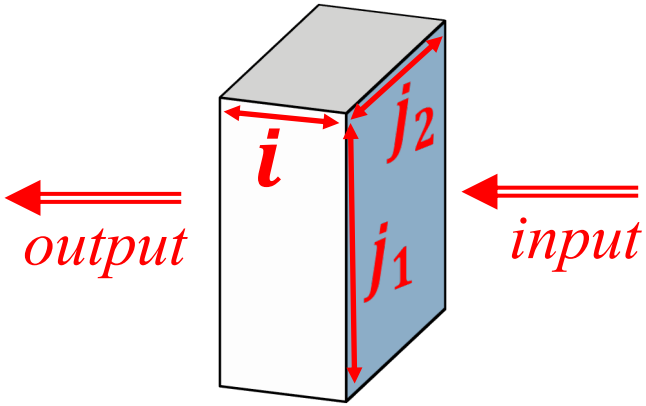
Tensors

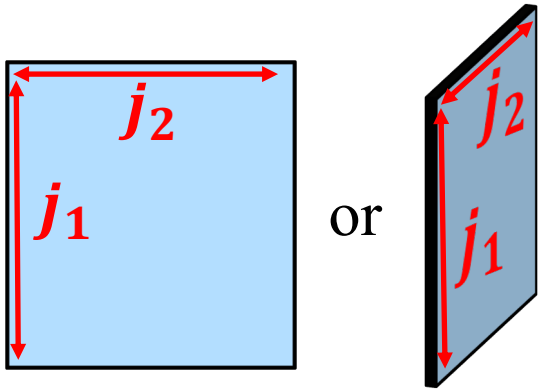
- Contravariant and covariant tensors
- Einstein notation
- Interpretation

What is a Tensor?

- It is the generalization of a:
  - Matrix operator for >2 dimensions



  - Vector data to >1 dimension



- Why do we need it?
  - Came out of differential geometry.
  - Was used by Albert Einstein to formulate the theory of General Relativity.
  - It's needed for many problems, including Deep NNs.

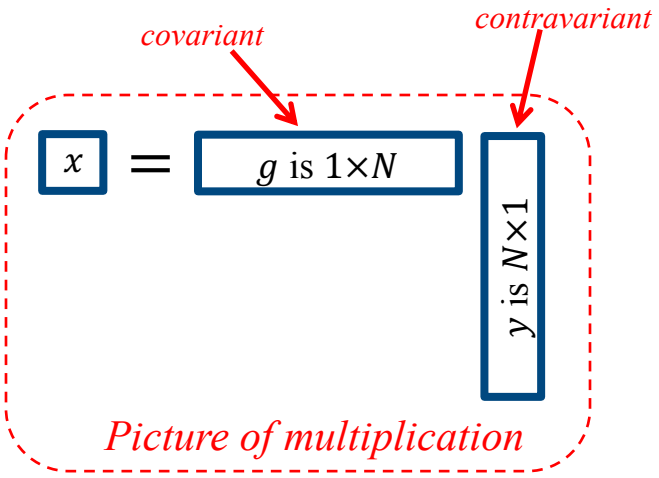
Contravariant and Covariant Vectors

$$x = gy$$

- Contravariant vectors:
  - "Column Vectors" that represent data
  - Vectors that describe the position of something
  - y^j for $0 \leq j < N$ and $x \in \Re$
- Covariant vector:
  - "Row vectors" that operate on data
  - Gradient vectors
  - g_j for $0 \leq j < N$
- Einstein notation

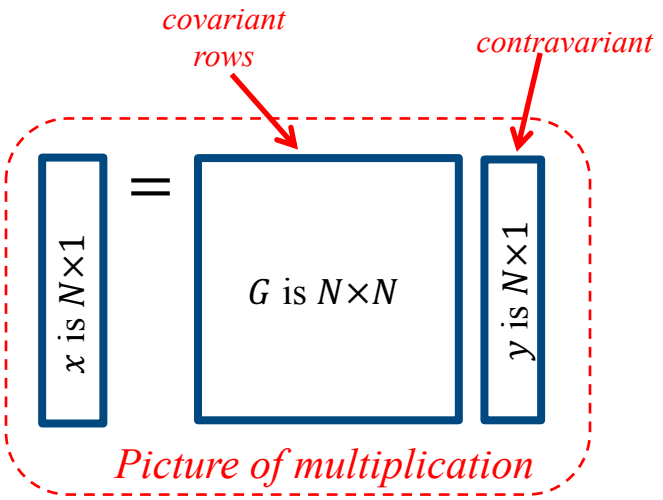
$$x = g_j\, y^j = \sum_{j=0}^{N-1} g_j\, y^j$$

  - Leave out the sum to make notation cleaner
  - Always sum over any two indices that appear twice



Vector-Matrix Products as Tensors

$$x = Gy$$



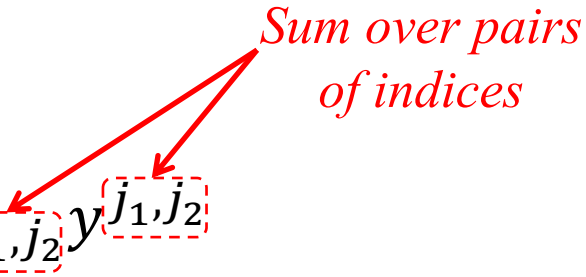
- 1D Contravariant vectors:
  - y^j for $0 \le j < N_y$
  - x^j for $0 \le j < N_x$
- 2D tensor (i.e., matrix):
  - $G^i{}_j$ for $0 \le i < N_x$ and $0 \le j < N_y$
  - There is a gradient for each component x^i
- Einstein notation

$$x^i = G^i{}_j\, y^j = \sum_{j=0}^{N_y-1} G^i{}_j\, y^j$$

  - Leave out the sum to make notation cleaner
  - Always sum over any two indices that appear twice

Tensor Products

- Einstein notation



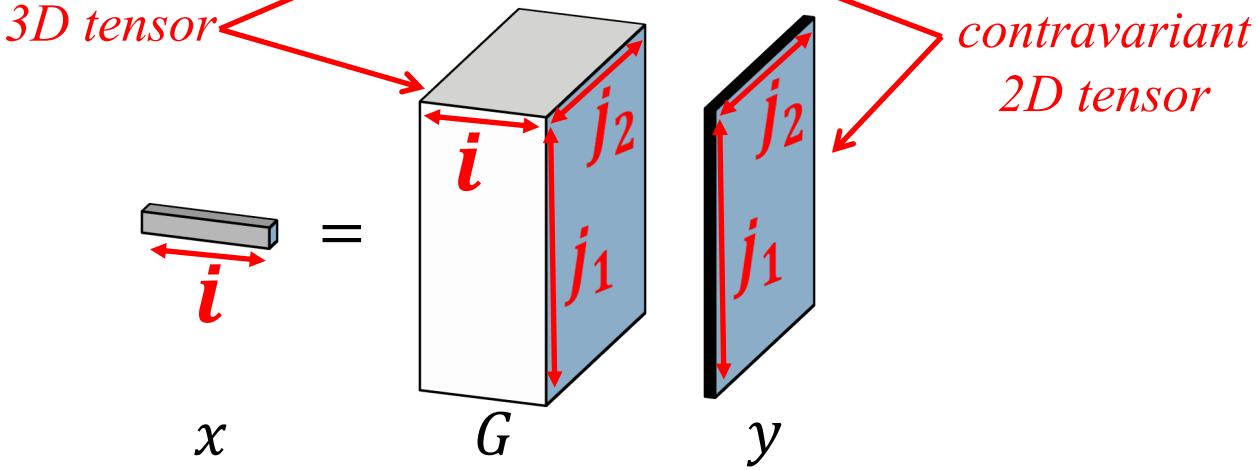
$$x^{i_1,i_2} = G^{i_1,i_2}{}_{j_1,j_2} y^{j_1,j_2}$$

- 2-D Contravariant vectors:
  - y^{j_1,j_2} for $0 \leq j_1, j_2 < N_y$
  - x^{i_1,i_2} for $0 \leq i_1, i_2 < N_x$
- 4-D Tensor:
  - $G^{i_1,i_2}{}_{j_1,j_2}$ for $0 \leq i_1, i_2 < N_x$ and $0 \leq j_1, j_2 < N_y$
  - G is known as a tensor
    - 2D covariant input
    - 2D contravariant output

Picture of a Tensor Product

- For example, if we have

$$x^i = G^i{}_{j_1,j_2}\, y^{j_1,j_2}$$



  - Tensor 3-D tensor G has

    Input: 2D image indexed by (j_1, j_2)

    Output: 1D vector indexed by i

  - General idea: $G \in \Re^{N_o \times N_i}$

Some Useful Definitions

- Delta functions:

$$\delta^i{}_j = \begin{cases} 1 & \text{if } i = j \\ 1 & \text{if } i \neq j \end{cases}$$

  - So then we have that

$$G^k{}_j = G^k{}_i \, \delta^i{}_j$$

- Gradient w.r.t. vector

$$\nabla_y (Ay) = A$$

  - In tensor notation, we have that

$$[\nabla_A (Ay)]^i{}_j = A^i{}_j$$

- Gradient w.r.t. matrix

$$\nabla_A (Ay) = ??$$

  - First, it must have 1 output dimension and 2 input dimensions. So we have that

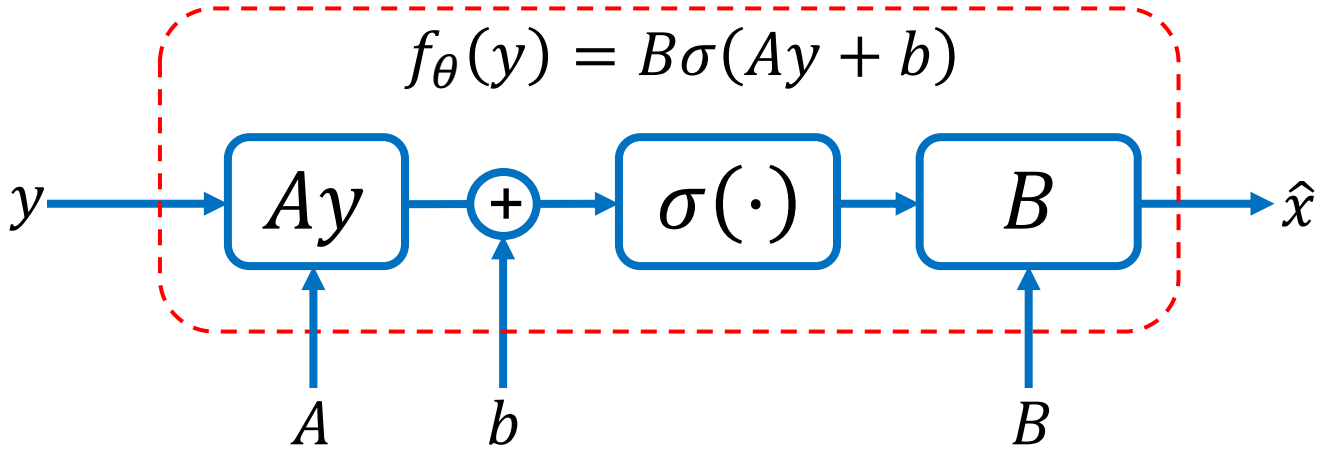
$$[\nabla_A (Ay)]^i{}_{j_1,j_2} = \delta^i{}_{j_1} y^{j_2}$$

GD for Single Layer NN

- Structure of the Gradient
- Gradients for NN parameters
- Updates for NN parameters

Gradient Direction for Single Layer NN

- Single layer NN:



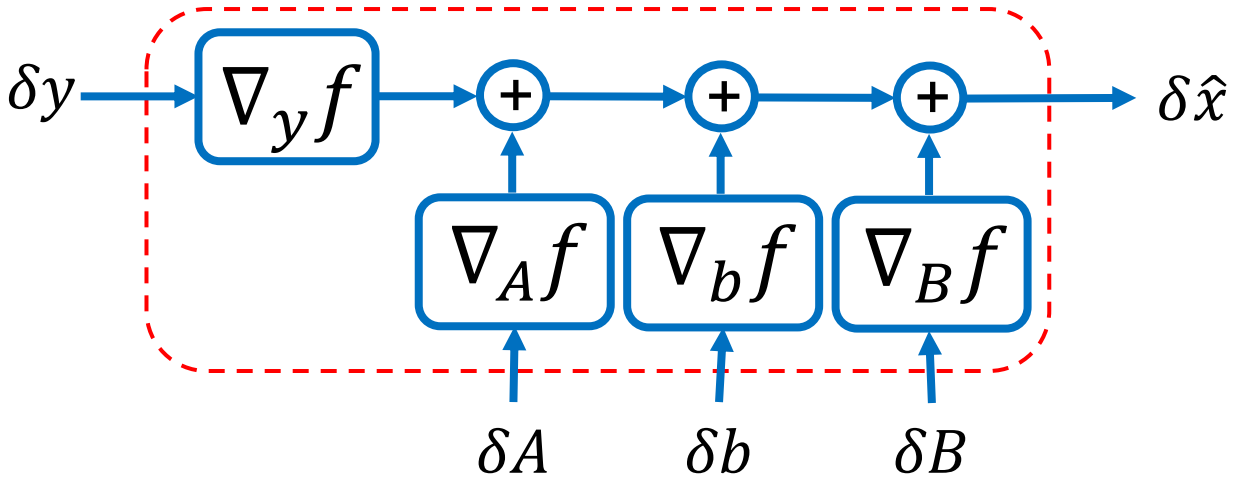
  - We will need gradient w.r.t the parameters $\theta = (A, B, b)$:

$$\nabla_\theta f_\theta(y) = \left[\nabla_A f_{(A,b,B)}(y), \nabla_b f_{(A,b,B)}(y), \nabla_B f_{(A,b,B)}(y)\right]$$

  - Later, we will also need:

$$\nabla_y f_\theta(y)$$

Gradient Structure for Single Layer NN



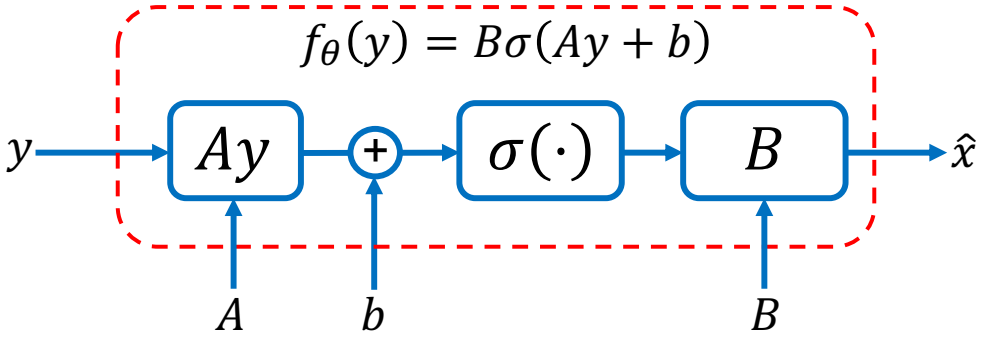
- Single layer NN:
  - Parameters are $\theta = (A, B, b)$
  - We will need the parameter gradients:

$$\nabla_\theta f_\theta(y) = \left[\nabla_A f_{(A,b,B)}(y), \nabla_b f_{(A,b,B)}(y), \nabla_B f_{(A,b,B)}(y)\right]$$

  - And the input gradient:

$$\nabla_y f_\theta(y)$$

Gradient w.r.t. y



- For this case,

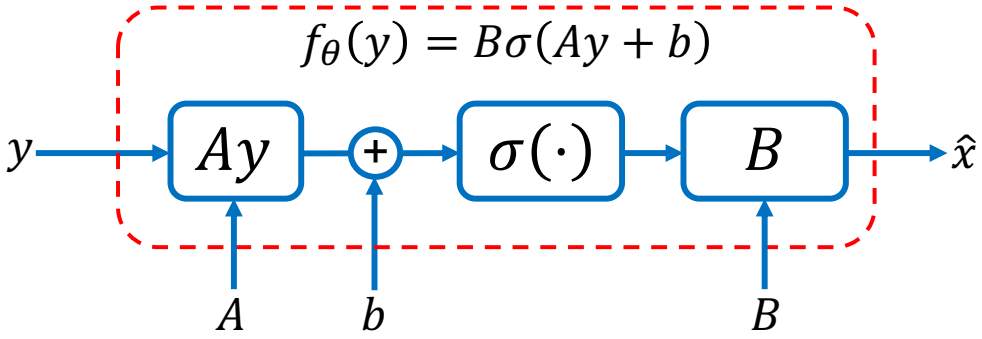
$$\nabla_y f = \nabla_y f_\theta(y) = B[\nabla\sigma(Ay + b)]A$$

  - Using Einstein notation

$$\left[\nabla_y f\right]^i{}_j = B^i{}_{i_1}[\nabla\sigma]^{i_1}{}_{i_2}A^{i_2}{}_j$$

This is a matrix or 2-D tensor

Gradient w.r.t. A



- For this case,

$$\nabla_A f = B\,[\nabla\sigma(Ay + b)]\,\nabla_A(Ay)$$

This is a tensor!

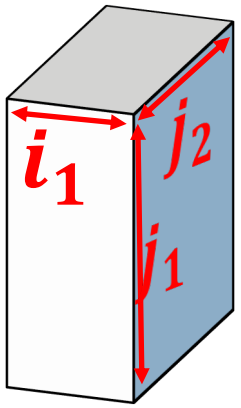
  - Using Einstein notation, since

$$[\nabla_A(Ay)]^{i}{}_{j_1,j_2} = \delta^{i}{}_{j_1} y^{j_2}$$

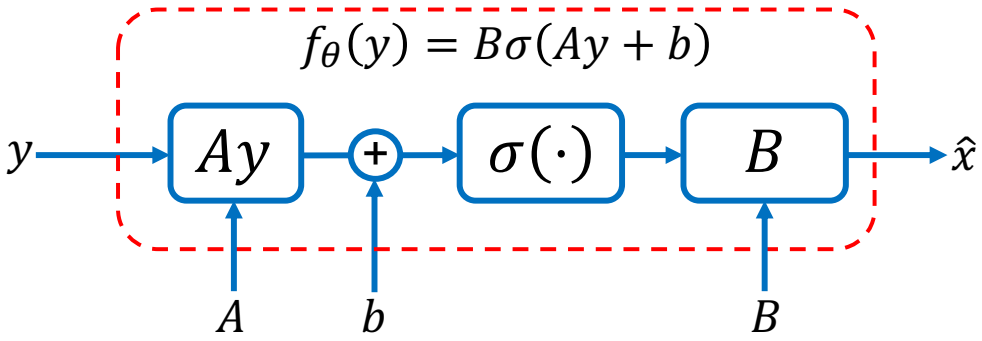
  - Then

$$\begin{aligned}[\nabla_A f]^{i_1}{}_{j_1,j_2} &= B^{i_1}{}_{i_2}[\nabla\sigma]^{i_2}{}_{i_3}[\nabla_A(Ay)]^{i_3}{}_{j_1,j_2} \\ &= B^{i_1}{}_{i_2}[\nabla\sigma]^{i_2}{}_{i_3}\delta^{i_3}{}_{j_1} y^{j_2}\end{aligned}$$

$$[\nabla_A f]^{i_1}{}_{j_1,j_2} = B^{i_1}{}_{i_2}[\nabla\sigma]^{i_2}{}_{j_1}\, y^{j_2}$$



Gradient w.r.t. b



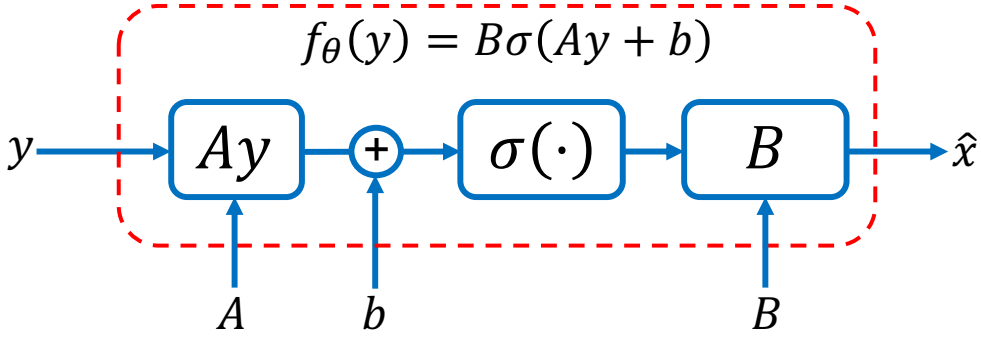
- For this case,

$$\nabla_b f = B\,[\nabla\sigma(Ay + b)]\,\nabla_b(Ay + b)$$
$$= B\,[\nabla\sigma]I = B\,\nabla\sigma$$

  - Using Einstein notation,

$$[\nabla_b f]^{i_1}{}_{j_1} = B^{i_1}{}_{i_2}[\nabla\sigma]^{i_2}{}_{j_1}$$

Gradient w.r.t. B



- For this case,

$$\nabla_B f = \nabla_B (B\sigma)$$

This is a tensor!

  - Using Einstein notation, since

$$[\nabla_B (B\sigma)]^i{}_{j_1,j_2} = \delta^i{}_{j_1}\sigma^{j_2}$$

  - We have that

$$[\nabla_b f]^{i_1}{}_{j_1,j_2} = \delta^{i_1}{}_{j_1}\sigma^{j_2}$$

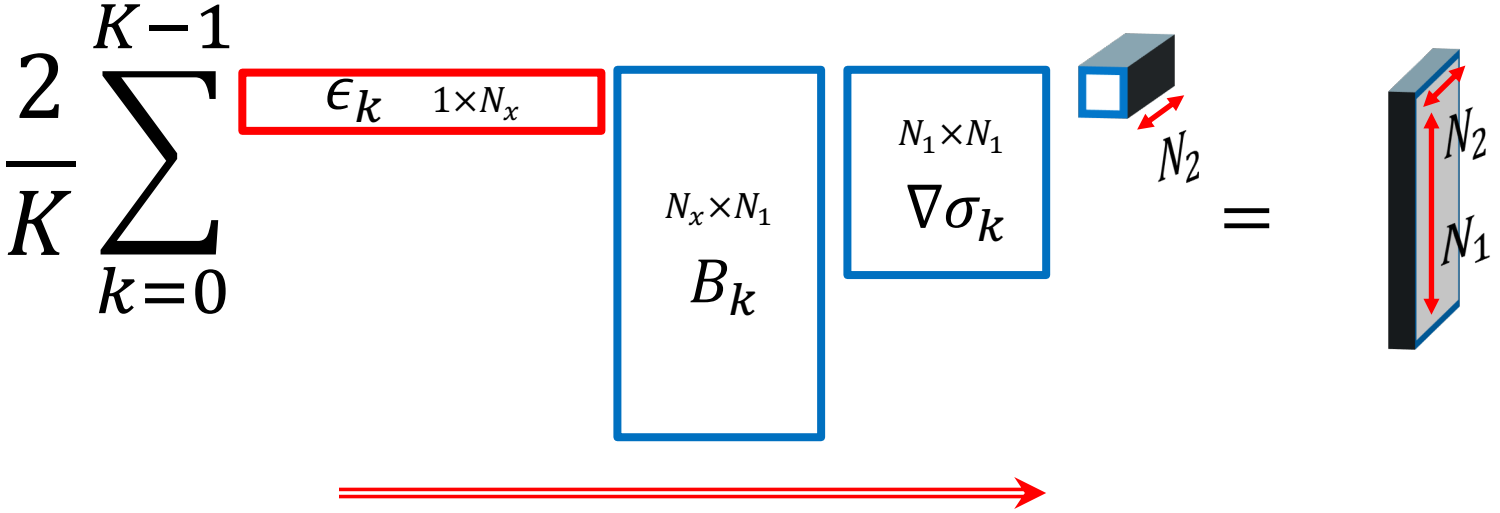
Update Direction for A

Gradient step:

$$A \leftarrow A + \alpha d^t$$

- d is given by

$$d_{j_1,j_2} = \frac{2}{K} \sum_{k=0}^{K-1} \epsilon_{k,i_1} \, {B_k}^{i_1}{}_{i_2} [\nabla \sigma_k]^{i_2}{}_{j_1} \, {y_k}^{j_2}$$



For efficiency, computation goes this way!

Update Direction for b

Gradient step:

$$b \leftarrow b + \alpha d^t$$

- d is given by

$$d_{j_1} = \frac{2}{K} \sum_{k=0}^{K-1} \epsilon_{k,i_1} \, B_k{}^{i_1}{}_{i_2} [\nabla \sigma_k]^{i_2}{}_{j_1}$$

$$\frac{2}{K} \sum_{k=0}^{K-1} \underbrace{\epsilon_k}_{1 \times N_x} \underbrace{B_k}_{N_x \times N_1} = \underbrace{}_{N_1 \times 1}$$

For efficiency, computation goes this way!

Update Direction for B

Gradient step:

$$B \leftarrow B + \alpha d^t$$

- d is given by

$$d_{j_1,j_2} = \frac{2}{K} \sum_{k=0}^{K-1} \epsilon_{k,j_1} \sigma_k{}^{j_2}$$

$$\frac{2}{K} \sum_{k=0}^{K-1} \epsilon_k \quad 1 \times N_x \qquad \sigma_k \quad N_2 \qquad = \qquad N_x \quad N_2$$

For efficiency, computation goes this way!

Local and Global Minima

- Open and Closed Sets
- Convex Sets and Functions
- Properties of Convex Functions
- Local Minimum, Saddle Points, and Global Minima
- Optimization Theorems

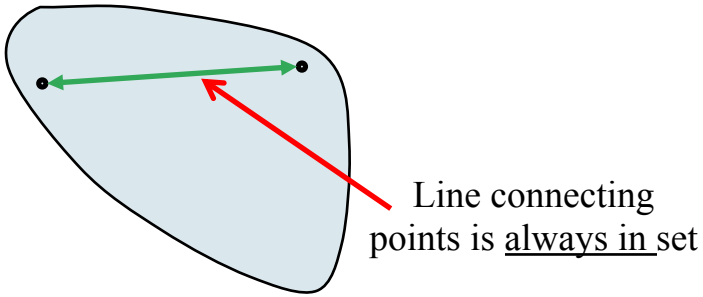
Open and Closed Sets

- Define:
  - $\mathcal{A} \subset \Re^N$
  - Open ball of diameter ϵ is $B(\epsilon) = \{r \in \Re^N : \|r - r_o\| < \epsilon\}$.
- A set $\mathcal{A}$ is open if
  - At every point, there is an open ball contained in $\mathcal{A}$.
  - $\forall r \in \mathcal{A},\ \exists \epsilon > 0$ s.t. $B(\epsilon) \subset \mathcal{A}$.
- A set $\mathcal{A}$ is closed if $\mathcal{A}^c = \Re^N - \mathcal{A}$ is open.
- A set $\mathcal{A}$ is compact if it is closed and bounded.
- Facts:
  - $\Re^N$ is both open and closed, but it is not compact.
  - If $\mathcal{A}$ is compact, than every sequence in $\mathcal{A}$ has a limit point in $\mathcal{A}$.

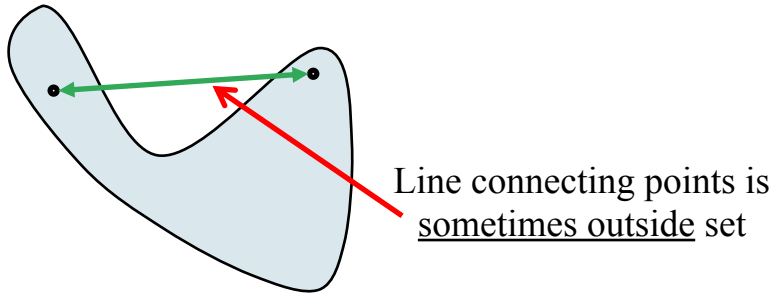
Convexity Sets

- A set $\mathcal{A}$ is convex if

  $\forall \lambda \in (0,1), \forall s, r \in \mathcal{A}$, then $\lambda r + (1-\lambda)s \in \mathcal{A}$



Convex Set



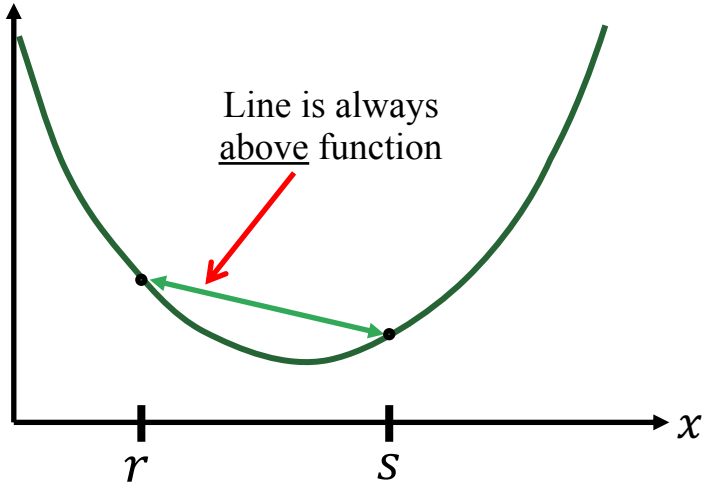
Nonconvex Set

- Properties:
  - The intersection of convex sets is convex

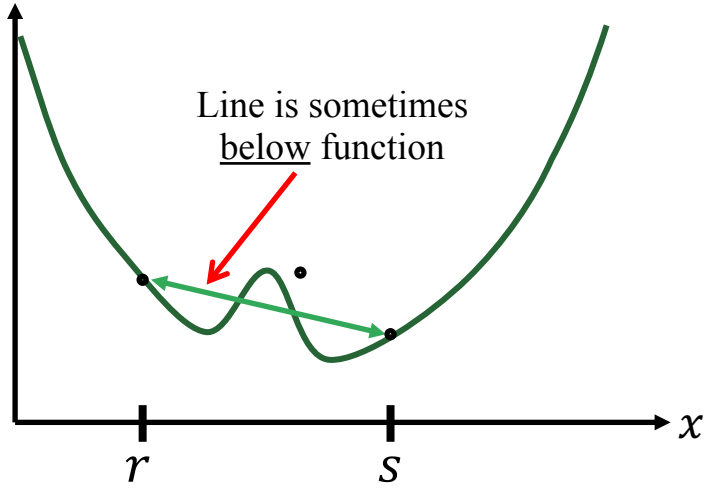
Convexity Functions

- Let $f: \mathcal{A} \rightarrow \Re$ where $\mathcal{A}$ is a convex set. Then we say that f is a convex function if

$$\forall \lambda \in (0,1), \forall s, r \in \mathcal{A}, \text{ then } f(\lambda r + (1-\lambda)s) \geq \lambda f(r) + (1-\lambda) f(s)$$



Convex Function



Nonconvex Function

- Properties:
  - The sums of convex functions is convex
  - The maximum of a set of convex functions is convex
  - If $f(y)$ is convex, then $f(Ay)$ is also convex.
  - $f(y)$ is concave if $-f(y)$ is convex.
  - $f(y) = Ay + b$ is both convex and concave

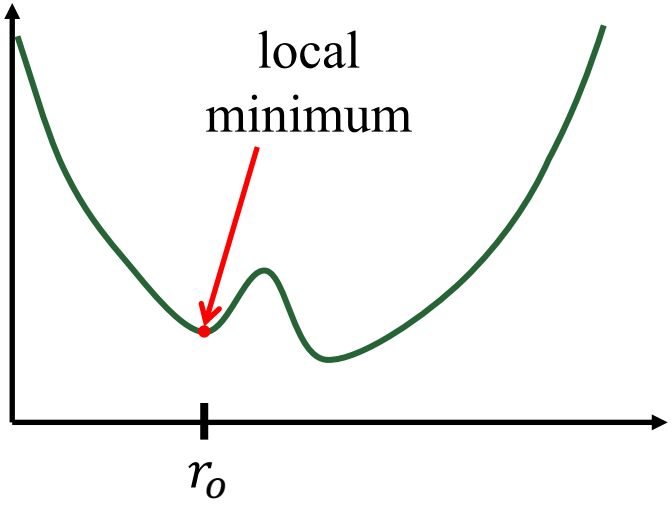
Properties of Convex Functions

- Valuable properties of convex functions

- The sums of convex functions is convex.
  - Let $f(x) = \sum_n f_n(x)$. If $f_n(x)$ are convex, then $f(x)$ is convex.
- The maximum of a set of convex functions is convex.
  - Let $f(x) = \max_n f_n(x)$. If $f_n(x)$ are convex, then $f(x)$ is convex.
- The second derivative of a convex function is positive.
  - Let $f(y)$ have two continuous derivatives, and let $H_{i,j}(y) = \frac{\partial f_i}{\partial y_i}$ be the Jacobian of f at y. Then $f(y)$ is convex if and only if $H(y)$ is non-negative definite for all y.
- A convex function of a linear transform is convex.
  - If $f(y)$ is convex, then $f(Ay)$ is also convex.
- An affine transform is both convex and concave.
  - $f(y) = Ay + b$ is both convex and concave.
- A function is concave if its negative is convex.
  - $f(y)$ is concave if $-f(y)$ is convex.

Local Minimum

- Let $f: \mathcal{A} \rightarrow \Re$ where $\mathcal{A} \subset \Re^N$.
  - We say that $r_o \in \mathcal{A}$ is a local minimum of f if there $\exists \epsilon > 0$ s.t. $\forall\, r \in \mathcal{A}$ s.t. $\|r - r_o\| < \epsilon$, then $f(r) \geq f(r_o)$.

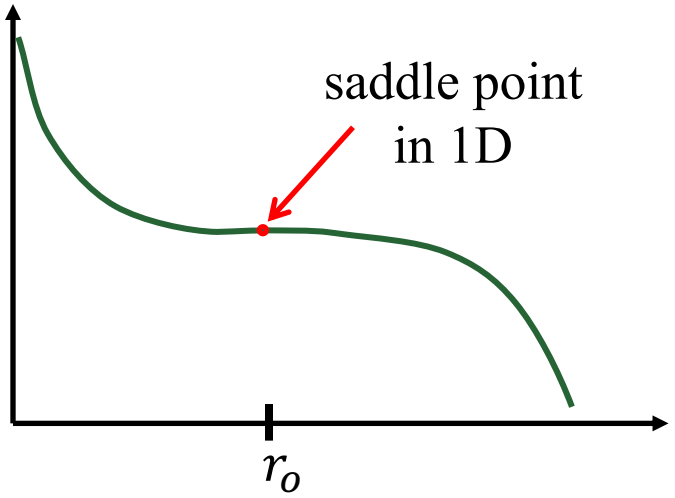


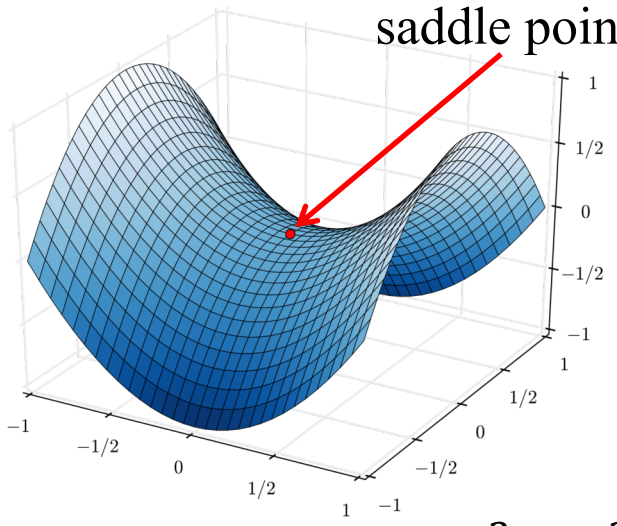
- Necessary condition for local minima:
  - Let f be continuously differentiable and let r_o be a local minimum, then $\nabla f(r_o) = 0$.

r_o a local minimum $\Rightarrow$ $\nabla f(r_o) = 0$

Saddle Point

- Let $f: \mathcal{A} \rightarrow \Re$ where $\mathcal{A} \subset \Re^N$ be a continuously differentiable function.
  - We say that $r_o \in \mathcal{A}$ is a saddle point of f if $\nabla f(r_o) = 0$ and r_o is not a local minimum.





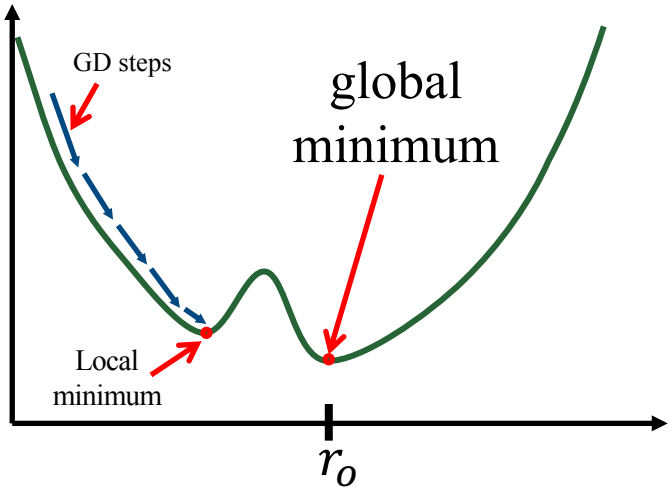
Graph of $f(x) = x_1^2 - x_2^2$

  - Saddle points can cause more problems than you might think.

*Shared from Wikipedia

Global Minimum

- Let $f: \mathcal{A} \rightarrow \Re$ where $\mathcal{A} \subset \Re^N$.
  - We say that $r_o \in \mathcal{A}$ is a global minimum of f if $\forall\, r \in \mathcal{A}, f(r) \geq f(r_o)$.



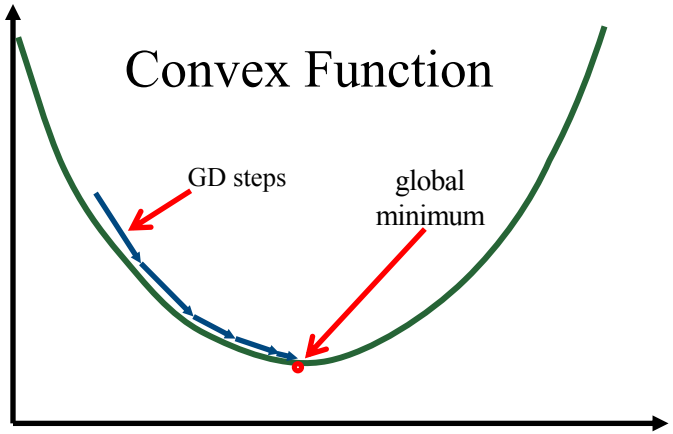
- Comments:
  - In general, finding global minimum is difficult.
  - Gradient descent optimization typically becomes trapped in local minima.

Optimization Theorems

- Let $f: \mathcal{A} \rightarrow \Re$ where $\mathcal{A} \subset \Re^N$.
  - If f is continuous and $\mathcal{A}$ is compact, then f takes on a global minimum in $\mathcal{A}$.
  - If f is convex on $\mathcal{A}$, then any local minimum is a global minimum.
  - If f is continuously differentiable and convex on $\mathcal{A}$, then $\nabla f(r_o) = 0$ implies that $r_o \in \mathcal{A}$ is a global minimum of f.

key idea, that's why we want convex loss functions

- Important facts:
  - Global minimum may not be unique.
  - If $\mathcal{A}$ is closed but not bounded, then it may not take on a global minimum.
  - Generally speaking, gradient descent algorithm converge to the global minimum of continuously differentiable convex functions.
  - Most interesting functions in ML are not convex!



Training and Generalization

- Overfitting, Underfitting, and Goldilocks Fitting
- Training, Validation, and Testing Data Sets
- Model Order, Model Capacity, Generalization Loss

Training and Generalization

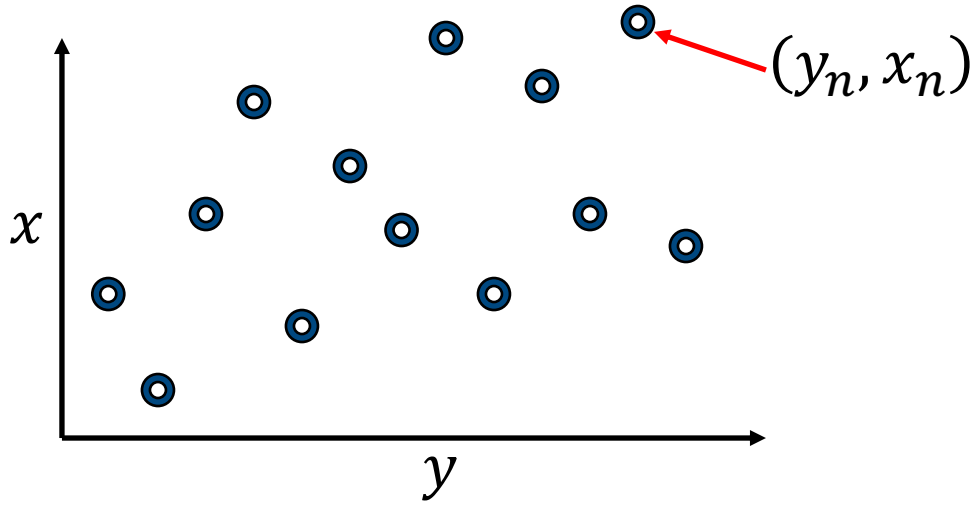
- Goal:
  - Learn the "true relationship" from training data pairs $(x_k, y_k)\big|_{k=0}^{K-1}$.

$$x = f_\theta(y) + error$$

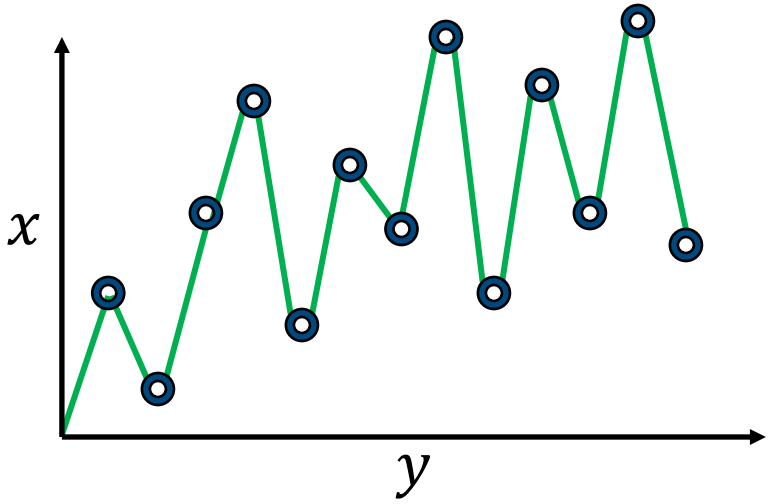
  - So what we learn needs to *generalize* beyond the training data.
- Key parameters:
  - P = Model Order = number of parameters = Dimension of $\theta \in \Re^P$
  - $N_x \times K$ = # training points = (Dimension of x)×(# of training pairs)
- Key issues
  - If $P \gg N_x \times K$: Model order is too high and there is a tendency to over fit.
  - If $P << N_x \times K$: Model order is too low, and there is a tendency to under fit

Overfitting

- Training data



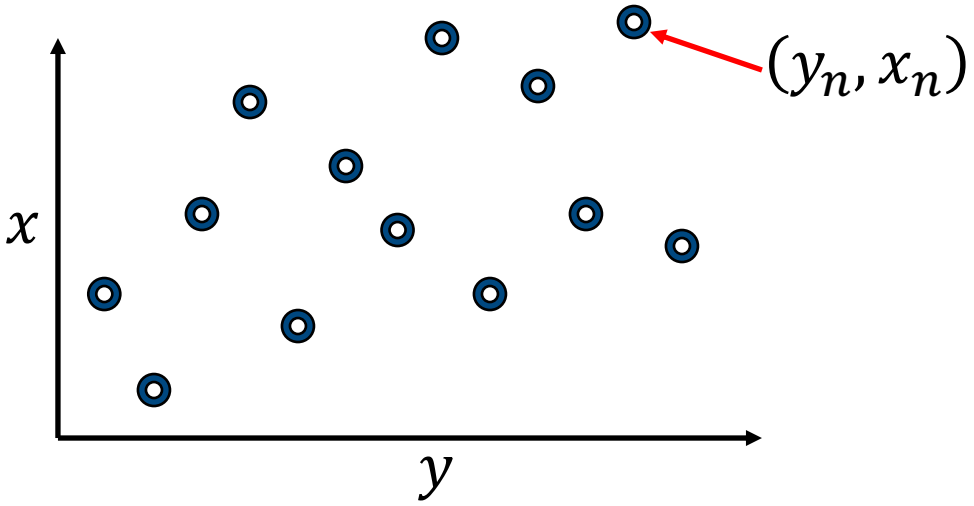
- Overfitting



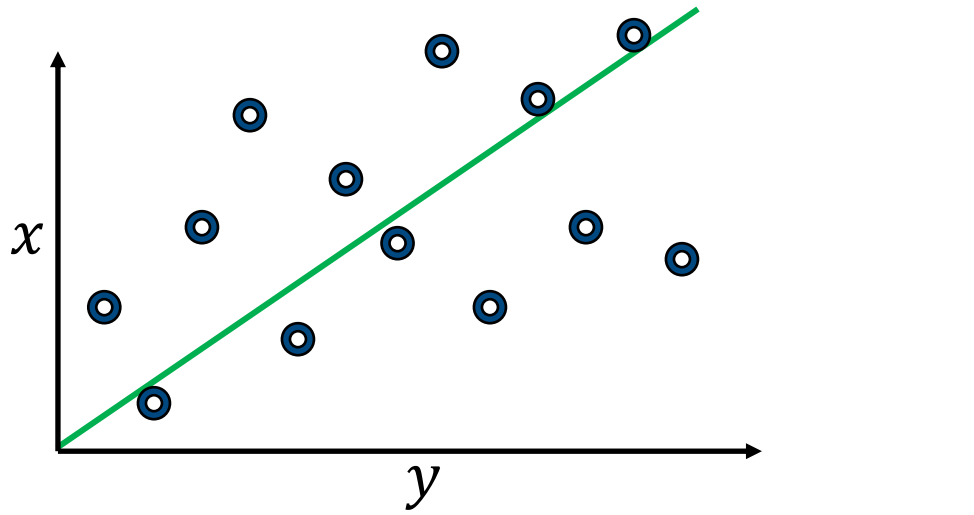
  - Model order too high
  - Doesn't generalize well

Underfitting

- Training data



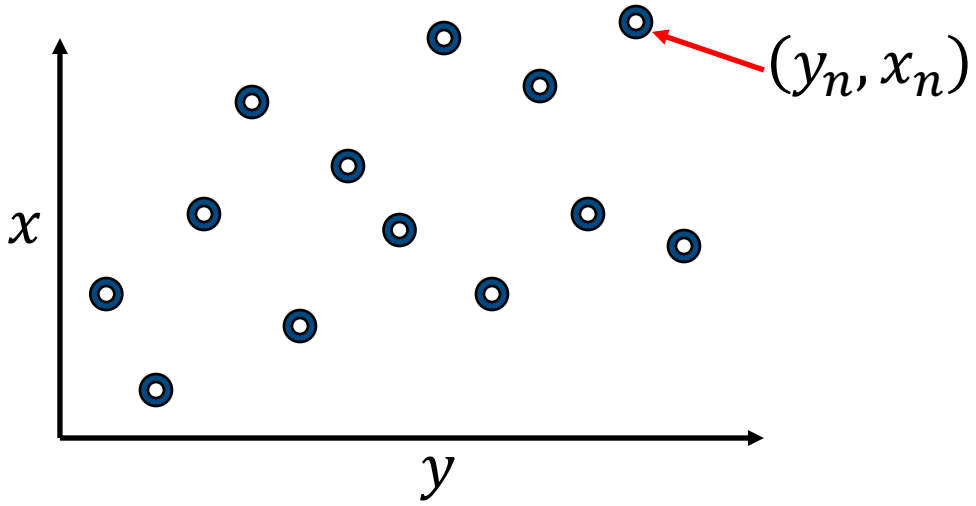
- Underfitting



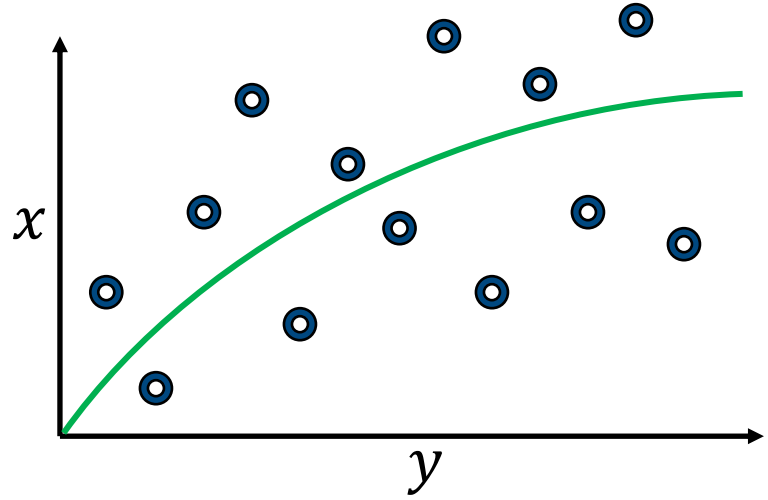
  - Model order too low
  - Doesn't generalize well

Goldilocks Fitting

- Training data



- Best fitting



  - Model order "just right"
  - Best generalization

Partitioning of Labeled Data

- Let (x_k, y_k) for k $\in S = \{0, \cdots, K-1\}$ be the full set data.
  - y_k is the input data.
  - x_k is the label or "ground truth" data.
- Typically, we randomly partition* the data into three subsets:
  - S_T is the training data
  - S_V is the validation data
  - S_E is the testing (evaluation) data
    - Note that "partition" means $S = S_\mathrm{T} \cup S_V \cup S_E$ and $\emptyset = S_T \cap S_V = S_T \cap S_E = S_V \cap S_E$
- For each partition, we define a loss function:

$$L_T(\theta) = \frac{1}{K} \sum_{k \in S_T} \|y_k - f_\theta(x_k)\|^2$$

$$L_V(\theta) = \frac{1}{K} \sum_{k \in S_V} \|y_k - f_\theta(x_k)\|^2$$

$$L_E(\theta) = \frac{1}{K} \sum_{k \in S_E} \|y_k - f_\theta(x_k)\|^2$$

Roles of Data

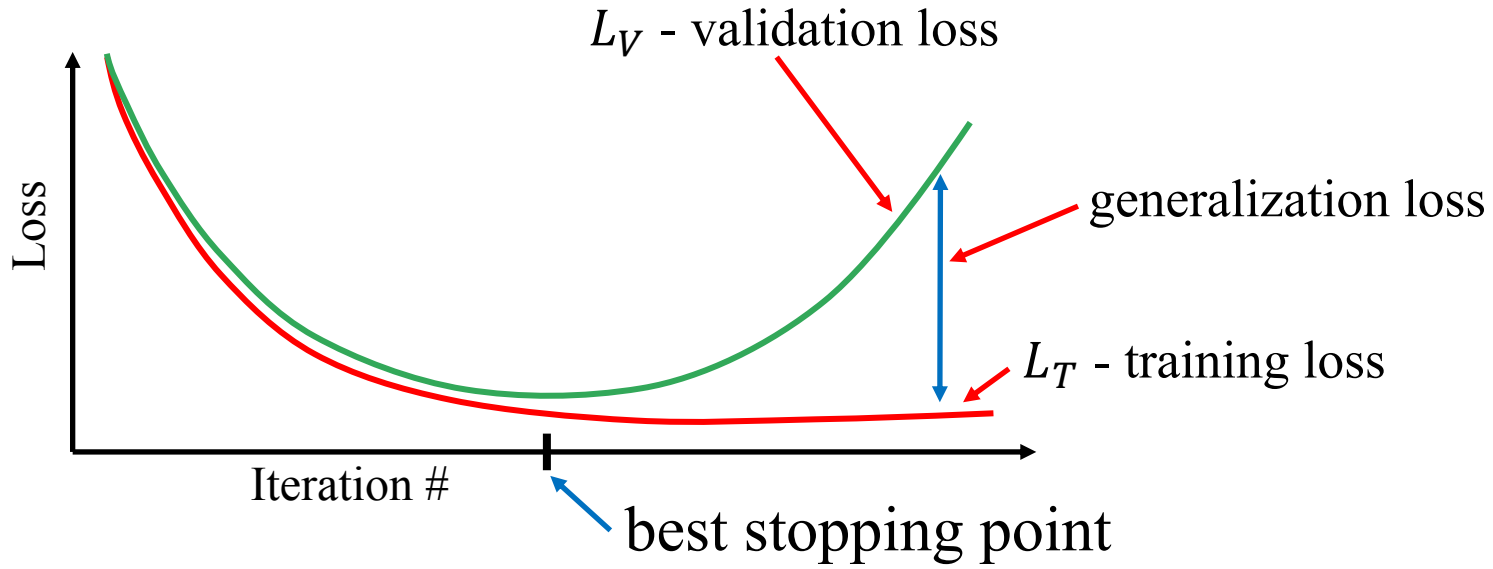
- Training data:
  - Only data used to train model

$$\theta^* = \arg\min_{\theta}\{L_T(\theta)\}$$
$$= \arg\min_{\theta}\left\{\frac{1}{K}\sum_{k\in S_T}\|y_k - f_\theta(x_k)\|^2\right\}$$

- Validation data:
  - Used to determine when model is property trained to compare models of different order.
- Testing data
  - Used for final evaluate of model performance.

Loss Function Convergence

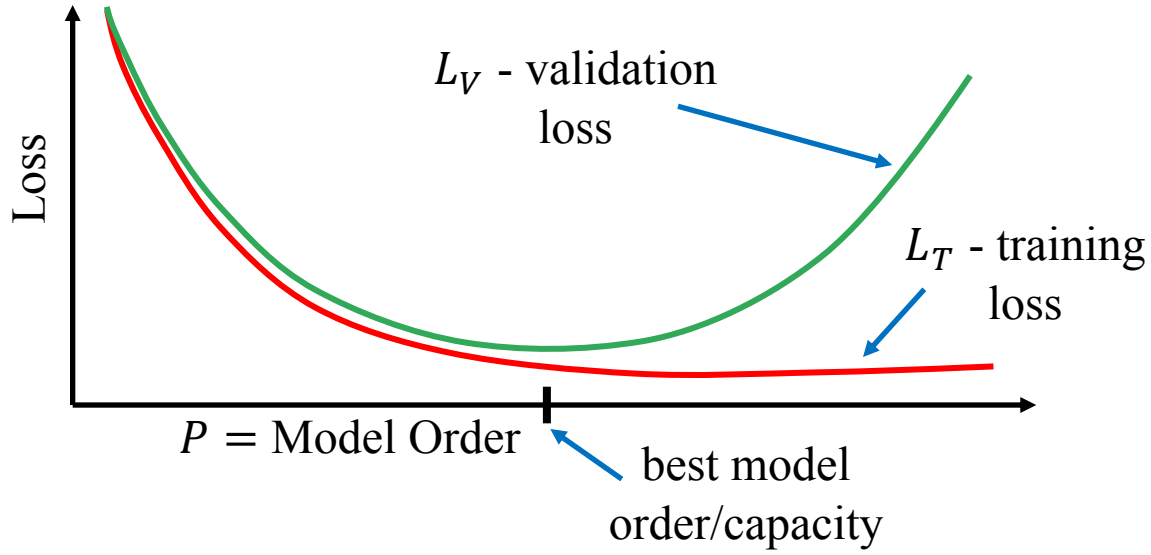
- Loss vs. iterations of gradient-based optimization



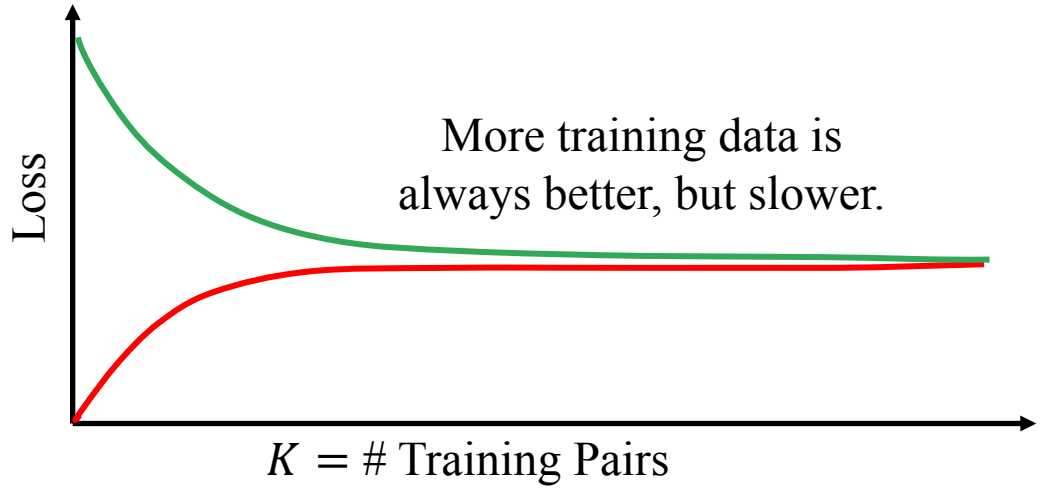
- Notice:
  - As training continues, the model is overfit to the data
  - Best to stop training when L_V is at a minimum
  - Model order is too high, but early termination of training can fix problem

Loss vs. Model Order vs. # Training Pairs

- Loss vs. model order

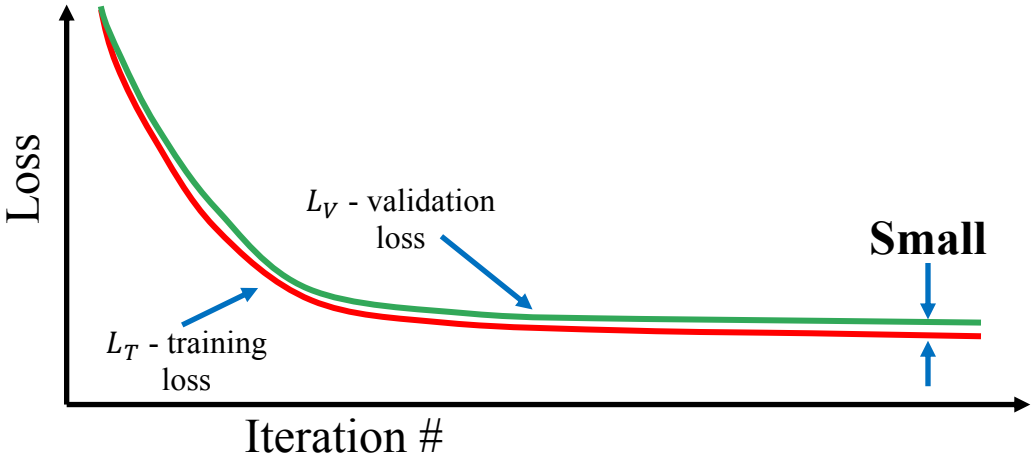


- Loss vs. # of training pairs

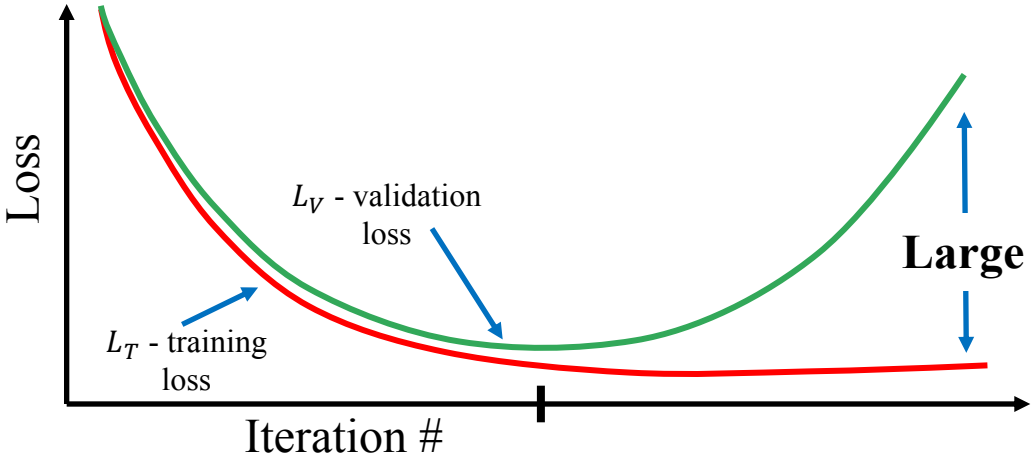


What are L_T and L_V telling you?

- Model order/model capacity may be too low…

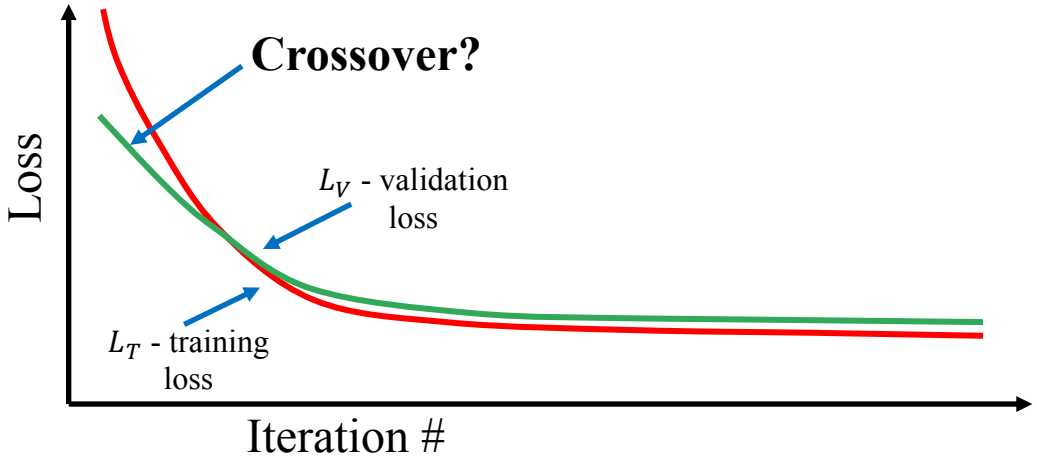


- Model order/model capacity may be too high…

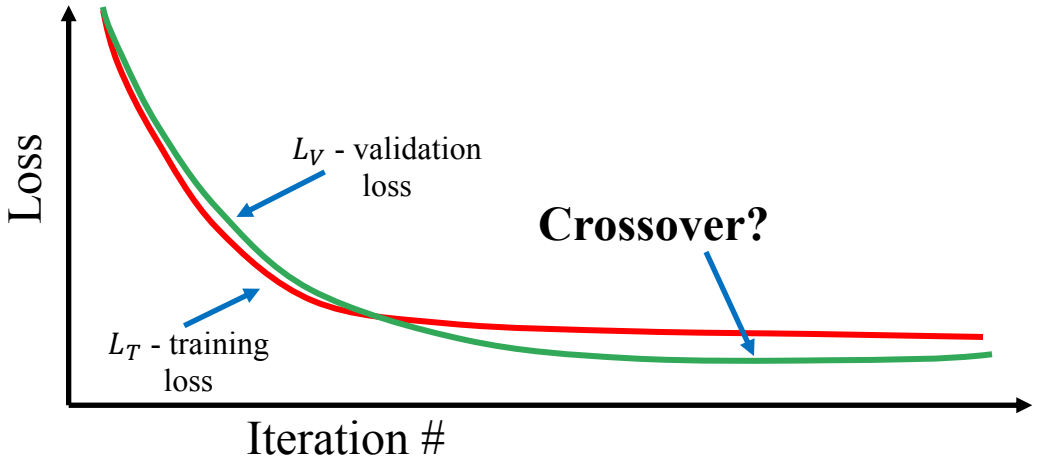


More of what are L_T and L_V are telling you?

- Something is screwed up…
  - Training sets are not randomly partitioned?

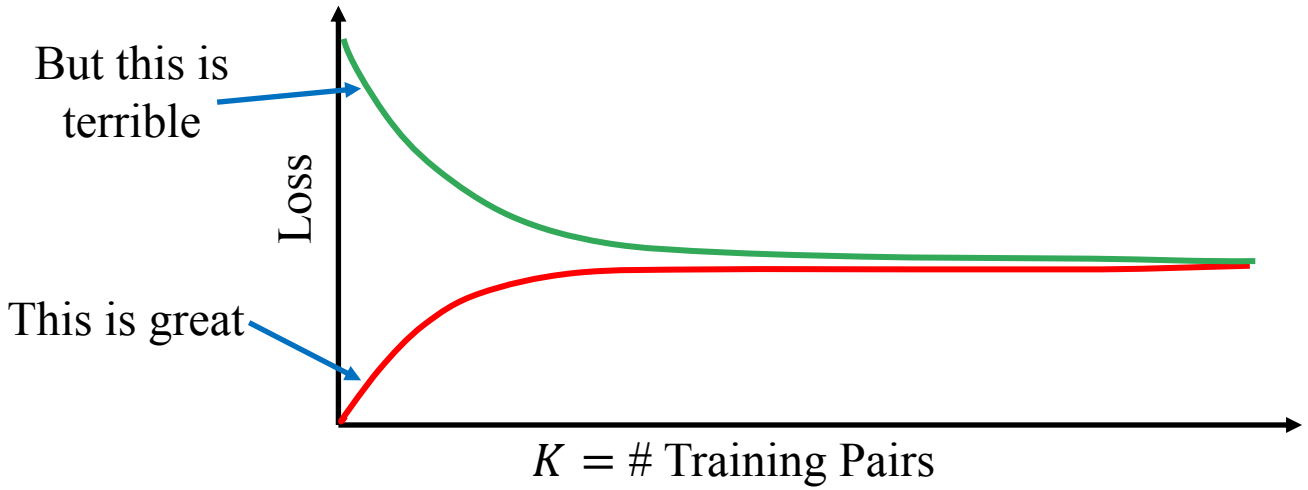


- Something is screwed up…
  - Maybe you optimization bug, or not property evaluating the validation error?



Never Test on Training Data!

- Never report training loss, L_T, as your ML system accuracy!



  - This is like doing a homework problem after you have seen the solution.
  - The network has "memorized" the answers.
- Don't even report validation loss, L_V, as your ML system accuracy.
  - This is also biased by the fact that your tuned model order parameters.
- Only report testing loss, L_E, as your ML system accuracy.
  - This data is sequestered to ensure it is an unbiased estimate of loss.