11-695: Competitive Engineering
Supervised Learning

Spring 2018

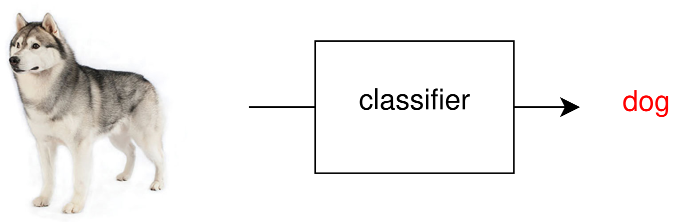
Outline

Carnegie Mellon

1. The Learning Problem
2. Data and Label
3. Modeling
4. Error and Loss Function
5. Learning
6. Overfitting and Regularization

Image Classification

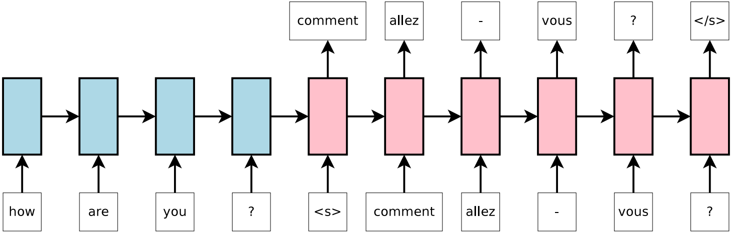
- Find a function $\mathbf{y} = \mathbf{f}(\mathbf{x})$
  - $\mathbf{x}$: an image
  - $\mathbf{y}$: dog, cat, bird, car, etc.



Machine Translation

Carnegie Mellon

- Find a function $\mathbf{y} = \mathbf{f}(\mathbf{x})$
  - $\mathbf{x}$: an English sentence
  - $\mathbf{y}$: an French sentence



How to Find the Function f?

How to Find the Function f?

Learn from data

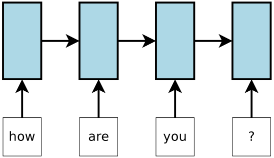
Outline

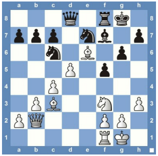
CarnegieMellon

1. The Learning Problem
2. Data and Label
3. Modeling
4. Error and Loss Function
5. Learning
6. Overfitting and Regularization

Data and Label

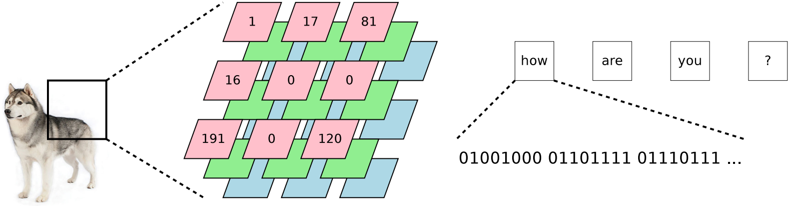
- Pairs of $(\mathbf{x}, \mathbf{y})$: $\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), ..., (\mathbf{x}^{(N)}, \mathbf{y}^{(N)})\}$
- Each $\mathbf{x}^{(i)}$ is called a *data point*
- Each $\mathbf{y}^{(i)}$ is called a *label*
- $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$ can be *anything*





Data Representation

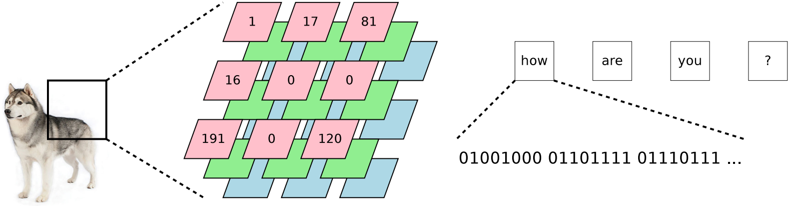
- Computers don't "see" things like we do



Data Representation

Carnegie Mellon

- Computers don't "see" things like we do



- ... so it's hard to make them think like we do

Some Types of Labels

CarnegieMellon

- Label $\mathbf{y}$ is in a finite set $\mathcal{C} = \{1, 2, ..., |\mathcal{C}|\}$
  - *Classification* problems

Some Types of Labels

CarnegieMellon

- Label $\mathbf{y}$ is in a finite set $\mathcal{C} = \{1, 2, ..., |\mathcal{C}|\}$
  - *Classification* problems
- Label $\mathbf{y}$ is in a "continuous" set, e.g. $\mathbf{y} \in [0, 1]$
  - *Regression* problems

Some Types of Labels

- Label $\mathbf{y}$ is in a finite set $\mathcal{C} = \{1, 2, ..., |\mathcal{C}|\}$
  - *Classification* problems
- Label $\mathbf{y}$ is in a "continuous" set, e.g. $\mathbf{y} \in [0, 1]$
  - *Regression* problems
- Label $\mathbf{y}$ is has some self-dependencies, e.g. a sentence in French
  - *Structured prediction* problems

Some Types of Labels

- Label $\mathbf{y}$ is in a finite set $\mathcal{C} = \{1, 2, ..., |\mathcal{C}|\}$
  - *Classification* problems
- Label $\mathbf{y}$ is in a "continuous" set, e.g. $\mathbf{y} \in [0, 1]$
  - *Regression* problems
- Label $\mathbf{y}$ is has some self-dependencies, e.g. a sentence in French
  - *Structured prediction* problems
- Why learn these?
  - The types of problems you tackle (loosely) tell you how to design the learning models.

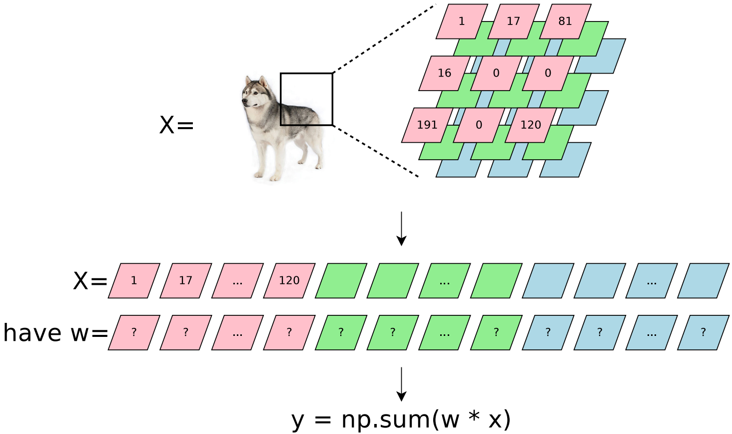
Outline

Carnegie Mellon

1. The Learning Problem
2. Data and Label
3. Modeling
4. Error and Loss Function
5. Learning
6. Overfitting and Regularization

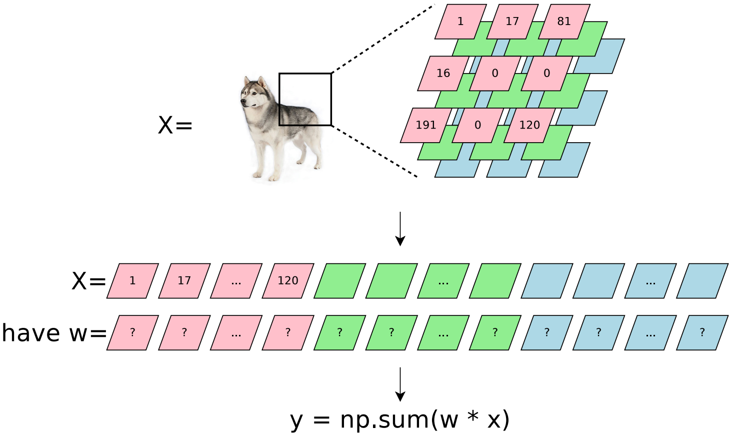
An Example

- Modeling is the process of coming up with *class of candidates*
  - Each value of w gives us an $\mathbf{f}$



An Example

- It's okay to come up with very bad classes of **f**



Some Maths

- Class of *hypotheses* $\mathcal{H} = \{\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_N\}$
  - singular: *hypothesis*

Some Maths

Carnegie Mellon

- Class of *hypotheses* $\mathcal{H} = \{\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_N\}$
  - singular: *hypothesis*
- It is possible that $|\mathcal{H}| = \infty$
  - $\mathcal{H} = \{\mathbf{f}(\mathbf{x}; \mathbf{w}) = \sum_i w_i x_i | \text{ for all } \mathbf{w}\}$
  - Each $\mathbf{w}$ gives an $\mathbf{f}$, so we write $\mathbf{y} = f(\mathbf{x}; \mathbf{w})$
  - People also write $\mathbf{y} = \mathbf{f}_{\mathbf{w}}(\mathbf{x})$ or $\mathbf{y} = \mathbf{f}^{\mathbf{w}}(\mathbf{x})$

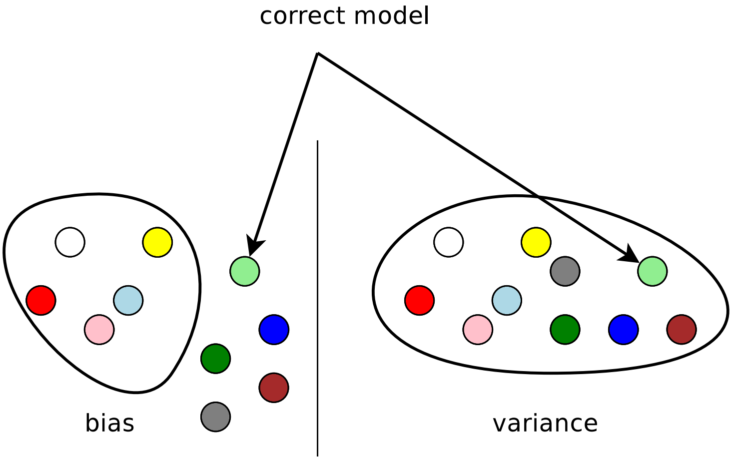
Some Maths

Carnegie Mellon

- Class of *hypotheses* $\mathcal{H} = \{\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_N\}$
  - singular: *hypothesis*
- It is possible that $|\mathcal{H}| = \infty$
  - $\mathcal{H} = \{\mathbf{f}(\mathbf{x}; \mathbf{w}) = \sum_i w_i x_i | \text{ for all } \mathbf{w}\}$
  - Each $\mathbf{w}$ gives an $\mathbf{f}$, so we write $\mathbf{y} = f(\mathbf{x}; \mathbf{w})$
  - People also write $\mathbf{y} = \mathbf{f}_{\mathbf{w}}(\mathbf{x})$ or $\mathbf{y} = \mathbf{f}^{\mathbf{w}}(\mathbf{x})$
- Some times you have more than one $\mathbf{w}$. For example:
  - $\mathbf{x} \in \mathbb{R}^{1\times 100}$
  - $\mathbf{y}_1 = (\mathbf{x} \cdot \mathbf{w}_1)^2$ where $\mathbf{w}_1 \in \mathbb{R}^{100\times 200}$
  - $\mathbf{y}_2 = 1/(\mathbf{y}_1 \cdot \mathbf{w}_2)$ where $\mathbf{w}_2 \in \mathbb{R}^{100\times 1}$
  - We use θ to denote all $\mathbf{w}$'s. $\mathbf{y} = \mathbf{f}(\mathbf{x}; \mathbf{w}_1, \mathbf{w}_2) = \mathbf{f}(\mathbf{x}; \theta)$. θ is called *parameters.*

The Bias-Variance Tradeoff

- Intuitions to design a good class of hypotheses $\mathcal{H}$
  - Bias: small $\mathcal{H}$ may leave out the correct model
  - Variance: large $\mathcal{H}$ is hard to navigate and find the correct model



Case Study 1: Car prize prediction

- Data: image of a car $\mathbf{x} \in \mathbb{R}^{3\times 512\times 512}$
- Label: cost of the car $\mathbf{x}$, namely $\mathbf{y} \in \mathbb{R}$

Case Study 1: Car prize prediction

CarnegieMellon

- Data: image of a car $\mathbf{x} \in \mathbb{R}^{3\times 512\times 512}$
- Label: cost of the car $\mathbf{x}$, namely $\mathbf{y} \in \mathbb{R}$
- Linear regression:

Case Study 1: Car prize prediction

CarnegieMellon

- Data: image of a car $\mathbf{x} \in \mathbb{R}^{3\times512\times512}$
- Label: cost of the car $\mathbf{x}$, namely $\mathbf{y} \in \mathbb{R}$
- Linear regression:
  - Parameters: $\theta = \{\mathbf{w} \in \mathbb{R}^{786432\times1}\}$, where $786432 = 3 \times 512 \times 512$.

Case Study 1: Car prize prediction

CarnegieMellon

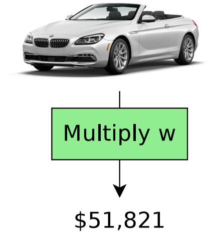
- Data: image of a car $\mathbf{x} \in \mathbb{R}^{3\times512\times512}$
- Label: cost of the car $\mathbf{x}$, namely $\mathbf{y} \in \mathbb{R}$
- Linear regression:
  - Parameters: $\theta = \{\mathbf{w} \in \mathbb{R}^{786432\times1}\}$, where $786432 = 3 \times 512 \times 512$.
  - $\mathbf{x}_1 = \text{reshape}(\mathbf{x}, [1, 786432])$

Case Study 1: Car prize prediction

- Data: image of a car $\mathbf{x} \in \mathbb{R}^{3\times 512\times 512}$
- Label: cost of the car $\mathbf{x}$, namely $\mathbf{y} \in \mathbb{R}$
- Linear regression:
  - Parameters: $\theta = \{\mathbf{w} \in \mathbb{R}^{786432\times 1}\}$, where $786432 = 3 \times 512 \times 512$.
  - $\mathbf{x}_1 = \text{reshape}(\mathbf{x}, [1, 786432])$
  - $\mathbf{y} = \mathbf{x}_1 \cdot \mathbf{w}$

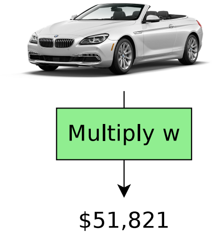
Case Study 1: Car prize prediction

- Data: image of a car $\mathbf{x} \in \mathbb{R}^{3\times512\times512}$
- Label: cost of the car $\mathbf{x}$, namely $\mathbf{y} \in \mathbb{R}$
- Linear regression:
  - Parameters: $\theta = \{\mathbf{w} \in \mathbb{R}^{786432\times1}\}$, where $786432 = 3 \times 512 \times 512$.
  - $\mathbf{x}_1 = \text{reshape}(\mathbf{x}, [1, 786432])$
  - $\mathbf{y} = \mathbf{x}_1 \cdot \mathbf{w}$



Case Study 1: Car prize prediction

- Data: image of a car $\mathbf{x} \in \mathbb{R}^{3\times512\times512}$
- Label: cost of the car $\mathbf{x}$, namely $\mathbf{y} \in \mathbb{R}$
- Linear regression:
  - Parameters: $\theta = \{\mathbf{w} \in \mathbb{R}^{786432\times1}\}$, where $786432 = 3 \times 512 \times 512$.
  - $\mathbf{x}_1 = \text{reshape}(\mathbf{x}, [1, 786432])$
  - $\mathbf{y} = \mathbf{x}_1 \cdot \mathbf{w}$
  - $\mathbf{y} = \mathbf{x} \cdot \mathbf{w}$ is called a *linear transformation.*



Case Study 2: Image classification

CarnegieMellon

- Data: $\mathbf{x} \in \mathbb{R}^{3\times 32\times 32}$
- Label: $\mathbf{y} \in \{\text{dog}, \text{cat}, \text{house}, \text{car}, \text{flower}\}$
  - For ease: $\mathbf{y} \in \{1, 2, 3, 4, 5\}$

Case Study 2: Image classification

Carnegie Mellon

- Data: $\mathbf{x} \in \mathbb{R}^{3\times32\times32}$
- Label: $\mathbf{y} \in \{\text{dog}, \text{cat}, \text{house}, \text{car}, \text{flower}\}$
  - For ease: $\mathbf{y} \in \{1, 2, 3, 4, 5\}$
- Softmax classification:

Case Study 2: Image classification

CarnegieMellon

- Data: $\mathbf{x} \in \mathbb{R}^{3\times 32\times 32}$
- Label: $\mathbf{y} \in \{\text{dog}, \text{cat}, \text{house}, \text{car}, \text{flower}\}$
  - For ease: $\mathbf{y} \in \{1, 2, 3, 4, 5\}$
- Softmax classification:
  - Parameters: $\theta = \{\mathbf{w} \in \mathbb{R}^{3072\times 10}\}$.

Case Study 2: Image classification

Carnegie Mellon

- Data: $\mathbf{x} \in \mathbb{R}^{3 \times 32 \times 32}$
- Label: $\mathbf{y} \in \{\text{dog}, \text{cat}, \text{house}, \text{car}, \text{flower}\}$
  - For ease: $\mathbf{y} \in \{1, 2, 3, 4, 5\}$
- Softmax classification:
  - Parameters: $\theta = \{\mathbf{w} \in \mathbb{R}^{3072 \times 10}\}$.
  - $\mathbf{x}_1 = \text{reshape}(\mathbf{x}, [1, 3072])$

Case Study 2: Image classification

CarnegieMellon

- Data: $\mathbf{x} \in \mathbb{R}^{3\times 32\times 32}$
- Label: $\mathbf{y} \in \{\text{dog}, \text{cat}, \text{house}, \text{car}, \text{flower}\}$
  - For ease: $\mathbf{y} \in \{1, 2, 3, 4, 5\}$
- Softmax classification:
  - Parameters: $\theta = \{\mathbf{w} \in \mathbb{R}^{3072\times 10}\}$.
  - $\mathbf{x}_1 = \text{reshape}(\mathbf{x}, [1, 3072])$
  - $\ell = \mathbf{x}_1 \cdot \mathbf{w}$

Case Study 2: Image classification

CarnegieMellon

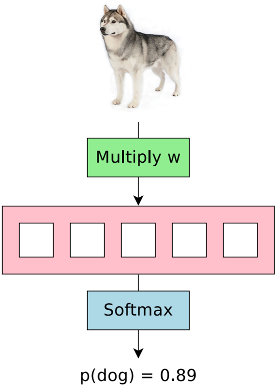
- Data: $\mathbf{x} \in \mathbb{R}^{3\times 32\times 32}$
- Label: $\mathbf{y} \in \{\text{dog}, \text{cat}, \text{house}, \text{car}, \text{flower}\}$
  - For ease: $\mathbf{y} \in \{1, 2, 3, 4, 5\}$
- Softmax classification:
  - Parameters: $\theta = \{\mathbf{w} \in \mathbb{R}^{3072\times 10}\}$.
  - $\mathbf{x}_1 = \text{reshape}(\mathbf{x}, [1, 3072])$
  - $\ell = \mathbf{x}_1 \cdot \mathbf{w}$
    - What is ℓ's dimension?

Case Study 2: Image classification

- Data: $\mathbf{x} \in \mathbb{R}^{3 \times 32 \times 32}$
- Label: $\mathbf{y} \in \{\text{dog}, \text{cat}, \text{house}, \text{car}, \text{flower}\}$
  - For ease: $\mathbf{y} \in \{1, 2, 3, 4, 5\}$
- Softmax classification:
  - Parameters: $\theta = \{\mathbf{w} \in \mathbb{R}^{3072 \times 10}\}$.
  - $\mathbf{x}_1 = \text{reshape}(\mathbf{x}, [1, 3072])$
  - $\ell = \mathbf{x}_1 \cdot \mathbf{w}$
    - What is ℓ's dimension?
  - $\mathbf{Prob}\left[\mathbf{y} = i\right] = \frac{\exp\{\ell_i\}}{\sum_{j=1}^{5} \exp\{\ell_j\}}$

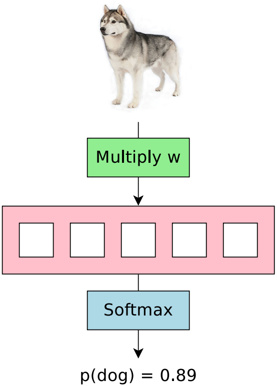
Case Study 2: Image classification

- Data: $\mathbf{x} \in \mathbb{R}^{3\times 32\times 32}$
- Label: $\mathbf{y} \in \{\text{dog}, \text{cat}, \text{house}, \text{car}, \text{flower}\}$
  - For ease: $\mathbf{y} \in \{1, 2, 3, 4, 5\}$
- Softmax classification:
  - Parameters: $\theta = \{\mathbf{w} \in \mathbb{R}^{3072\times 10}\}$.
  - $\mathbf{x}_1 = \text{reshape}(\mathbf{x}, [1, 3072])$
  - $\ell = \mathbf{x}_1 \cdot \mathbf{w}$
    - What is ℓ's dimension?
  - $\mathbf{Prob}\,[\mathbf{y} = i] = \dfrac{\exp\{\ell_i\}}{\sum_{j=1}^{5} \exp\{\ell_j\}}$



Case Study 2: Image classification

- Data: $\mathbf{x} \in \mathbb{R}^{3\times 32\times 32}$
- Label: $\mathbf{y} \in \{\text{dog}, \text{cat}, \text{house}, \text{car}, \text{flower}\}$
    - For ease: $\mathbf{y} \in \{1, 2, 3, 4, 5\}$
- Softmax classification:
    - Parameters: $\theta = \{\mathbf{w} \in \mathbb{R}^{3072\times 10}\}$.
    - $\mathbf{x}_1 = \text{reshape}(\mathbf{x}, [1, 3072])$
    - $\ell = \mathbf{x}_1 \cdot \mathbf{w}$
        - What is ℓ's dimension?
    - $\textbf{Prob}\,[\mathbf{y} = i] = \dfrac{\exp\{\ell_i\}}{\sum_{j=1}^{5} \exp\{\ell_j\}}$
    - The function $s(\ell) = \exp\{\ell_i\} / \sum_j \exp\{\ell_j\}$ is called the *Softmax function*.



Outline

How to Tell Good f from Bad f?

- Supposed you have designed the class of hypotheses
  - $\mathcal{H} = \{\mathbf{f}(\cdot;\theta) | \theta \in \mathbb{R}^D\}$

How to Tell Good f from Bad f?

- Supposed you have designed the class of hypotheses
  - $\mathcal{H} = \{\mathbf{f}(\cdot;\theta) | \theta \in \mathbb{R}^D\}$
- How do you tell if $\mathbf{f}(\cdot;\theta_1)$ is better than $\mathbf{f}(\cdot;\theta_2)$?

How to Tell Good f from Bad f?

Carnegie Mellon

- Supposed you have designed the class of hypotheses
  - $\mathcal{H} = \{\mathbf{f}(\cdot;\theta) | \theta \in \mathbb{R}^D\}$
- How do you tell if $\mathbf{f}(\cdot;\theta_1)$ is better than $\mathbf{f}(\cdot;\theta_2)$?
  - Idea 1: take a pair $(\mathbf{x}, \mathbf{y})$, look how different is $\mathbf{y}$ from $\mathbf{f}(\mathbf{x};\theta)$

How to Tell Good f from Bad f?

- Supposed you have designed the class of hypotheses
  - $\mathcal{H} = \{\mathbf{f}(\cdot;\theta) | \theta \in \mathbb{R}^D\}$
- How do you tell if $\mathbf{f}(\cdot;\theta_1)$ is better than $\mathbf{f}(\cdot;\theta_2)$?
  - Idea 1: take a pair $(\mathbf{x}, \mathbf{y})$, look how different is $\mathbf{y}$ from $\mathbf{f}(\mathbf{x};\theta)$
    - ▷ sample size is too small
    - ▷ may just get lucky (or unlucky)

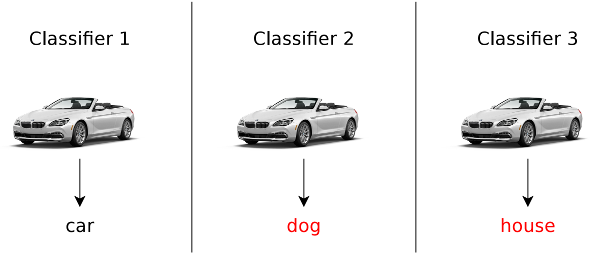
How to Tell Good f from Bad f?

Carnegie Mellon

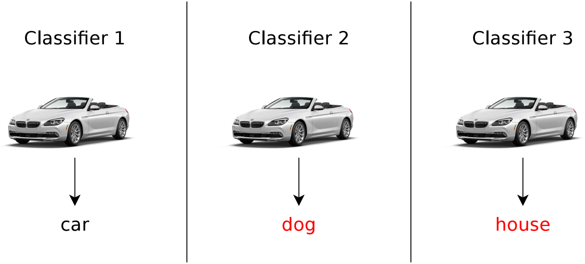
- Supposed you have designed the class of hypotheses
  - $\mathcal{H} = \{\mathbf{f}(\cdot;\theta) | \theta \in \mathbb{R}^D\}$
- How do you tell if $\mathbf{f}(\cdot;\theta_1)$ is better than $\mathbf{f}(\cdot;\theta_2)$?
  - Idea 1: take a pair $(\mathbf{x}, \mathbf{y})$, look how different is $\mathbf{y}$ from $\mathbf{f}(\mathbf{x};\theta)$
    - ▷ sample size is too small
    - ▷ may just get lucky (or unlucky)
  - Idea 2: take many pairs $(\mathbf{x}, \mathbf{y})$, look how different is $\mathbf{y}$ from $\mathbf{f}(\mathbf{x};\theta)$ *on average*

How to Tell Good f from Bad f?

- Supposed you have designed the class of hypotheses
  - $\mathcal{H} = \{\mathbf{f}(\cdot;\theta) | \theta \in \mathbb{R}^D\}$
- How do you tell if $\mathbf{f}(\cdot;\theta_1)$ is better than $\mathbf{f}(\cdot;\theta_2)$?
  - Idea 1: take a pair $(\mathbf{x}, \mathbf{y})$, look how different is $\mathbf{y}$ from $\mathbf{f}(\mathbf{x};\theta)$
    - sample size is too small
    - may just get lucky (or unlucky)
  - Idea 2: take many pairs $(\mathbf{x}, \mathbf{y})$, look how different is $\mathbf{y}$ from $\mathbf{f}(\mathbf{x};\theta)$ *on average*
    - How about this case? Which is worse?

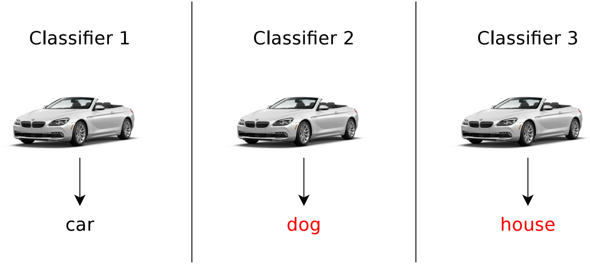


Loss Function



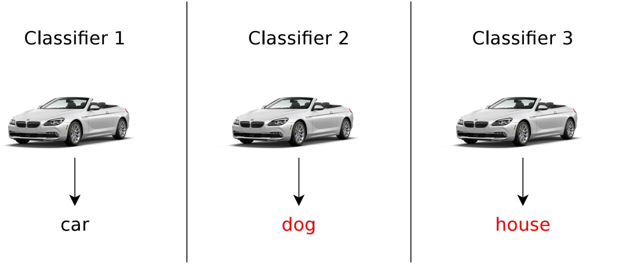
- Notations:
  - $\mathbf{x}$ is your data; $\mathbf{y}$ is your label;
  - $\mathbf{f}$ is your model; θ is your parameter;
  - $\hat{\mathbf{y}} = \mathbf{f}(\mathbf{x}; \theta)$ is your *empirical prediction.*

Loss Function



- Notations:
  - $\mathbf{x}$ is your data; $\mathbf{y}$ is your label;
  - $\mathbf{f}$ is your model; θ is your parameter;
  - $\hat{\mathbf{y}} = \mathbf{f}(\mathbf{x};\theta)$ is your *empirical prediction.*
- Loss function: $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \mathcal{L}(\mathbf{y}, \mathbf{f}(\mathbf{x}, \theta))$

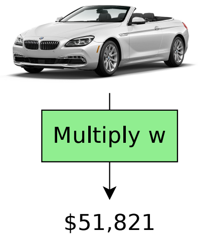
Loss Function



- Notations:
  - $\mathbf{x}$ is your data; $\mathbf{y}$ is your label;
  - $\mathbf{f}$ is your model; θ is your parameter;
  - $\hat{\mathbf{y}} = \mathbf{f}(\mathbf{x};\theta)$ is your *empirical prediction*.
- Loss function: $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \mathcal{L}(\mathbf{y}, \mathbf{f}(\mathbf{x}, \theta))$
  - How "off" is $\mathbf{y}$ from $\hat{\mathbf{y}} = \mathbf{f}(\mathbf{x};\theta)$

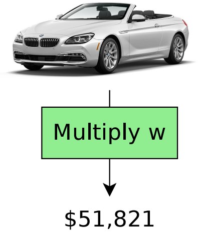
Case Study 1: Car prize prediction

- Data: image of a car $\mathbf{x} \in \mathbb{R}^{3 \times 512 \times 512}$
- Label: cost of the car $\mathbf{x}$, namely $\mathbf{y} \in \mathbb{R}$
- Linear regression:
  - Parameters: $\theta = \{\mathbf{w} \in \mathbb{R}^{786432 \times 1}\}$, where $786432 = 3 \times 512 \times 512$.
  - $\mathbf{x}_1 = \text{reshape}(\mathbf{x}, [1, 786432])$
  - $\hat{\mathbf{y}} = \mathbf{x}_1 \cdot \mathbf{w}$



Case Study 1: Car prize prediction

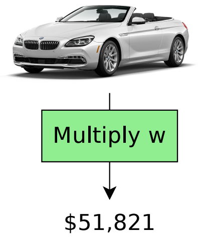
- Data: image of a car $\mathbf{x} \in \mathbb{R}^{3 \times 512 \times 512}$
- Label: cost of the car $\mathbf{x}$, namely $\mathbf{y} \in \mathbb{R}$
- Linear regression:
  - Parameters: $\theta = \{\mathbf{w} \in \mathbb{R}^{786432 \times 1}\}$, where $786432 = 3 \times 512 \times 512$.
  - $\mathbf{x}_1 = \text{reshape}(\mathbf{x}, [1, 786432])$
  - $\hat{\mathbf{y}} = \mathbf{x}_1 \cdot \mathbf{w}$
  - $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = (\hat{\mathbf{y}} - \mathbf{y})^2$ is called the ℓ_2-*loss*



Case Study 1: Car prize prediction

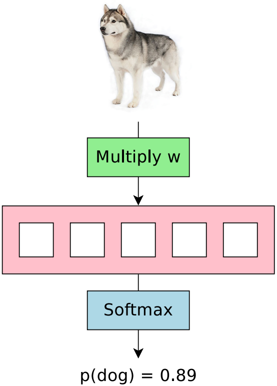
Carnegie Mellon

- Data: image of a car $\mathbf{x} \in \mathbb{R}^{3 \times 512 \times 512}$
- Label: cost of the car $\mathbf{x}$, namely $\mathbf{y} \in \mathbb{R}$
- Linear regression:
  - Parameters: $\theta = \{\mathbf{w} \in \mathbb{R}^{786432 \times 1}\}$, where $786432 = 3 \times 512 \times 512$.
  - $\mathbf{x}_1 = \text{reshape}(\mathbf{x}, [1, 786432])$
  - $\hat{\mathbf{y}} = \mathbf{x}_1 \cdot \mathbf{w}$
  - $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = (\hat{\mathbf{y}} - \mathbf{y})^2$ is called the ℓ_2-*loss*
  - $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = |\hat{\mathbf{y}} - \mathbf{y}|$ is called the ℓ_1-*loss*



Case Study 2: Image classification

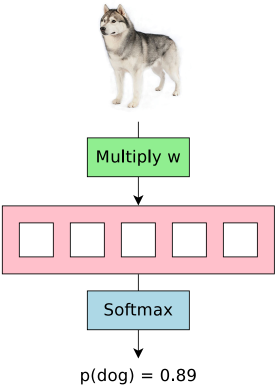
- Data: $\mathbf{x} \in \mathbb{R}^{3\times32\times32}$
- Label: $\mathbf{y} \in \{\text{dog}, \text{cat}, \text{house}, \text{car}, \text{flower}\}$
  - For ease: $\mathbf{y} \in \{1, 2, 3, 4, 5\}$
- Softmax classification:
  - Parameters: $\theta = \{\mathbf{w} \in \mathbb{R}^{3072\times10}\}$.
  - $\mathbf{x}_1 = \text{reshape}(\mathbf{x}, [1, 3072])$
  - $\ell = \mathbf{x}_1 \cdot \mathbf{w}$
  - $\hat{p} = \textbf{Prob}\left[\mathbf{y} = i\right] = \dfrac{\exp\{\ell_i\}}{\sum_{j=1}^{5} \exp\{\ell_j\}}$



Case Study 2: Image classification

Carnegie Mellon

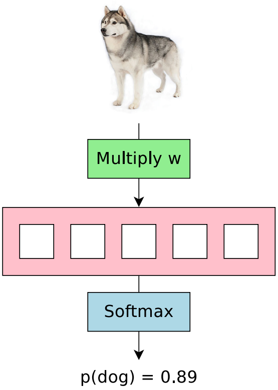
- Data: $\mathbf{x} \in \mathbb{R}^{3\times32\times32}$
- Label: $\mathbf{y} \in \{\text{dog}, \text{cat}, \text{house}, \text{car}, \text{flower}\}$
  - For ease: $\mathbf{y} \in \{1, 2, 3, 4, 5\}$
- Softmax classification:
  - Parameters: $\theta = \{\mathbf{w} \in \mathbb{R}^{3072\times10}\}$.
  - $\mathbf{x}_1 = \text{reshape}(\mathbf{x}, [1, 3072])$
  - $\ell = \mathbf{x}_1 \cdot \mathbf{w}$
  - $\hat{p} = \mathbf{Prob}\left[\mathbf{y} = i\right] = \dfrac{\exp\{\ell_i\}}{\sum_{j=1}^{5} \exp\{\ell_j\}}$
  - Cross-entropy loss: $\mathcal{L}(\hat{p}, \mathbf{y}) = -\log \hat{p}_{\mathbf{y}}$.



Cross-Entropy Loss

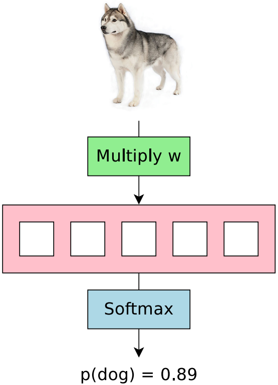
Carnegie Mellon

- $\mathcal{L}(\hat{p}, \mathbf{y}) = -\log \hat{p}_{\mathbf{y}}$
  - One of the most important loss functions of deep learning.



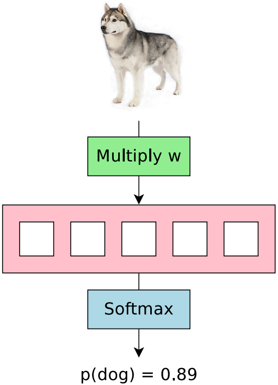
Cross-Entropy Loss

- $\mathcal{L}(\hat{p}, \mathbf{y}) = -\log \hat{p}_{\mathbf{y}}$
  - One of the most important loss functions of deep learning.
  - $\mathcal{L}$ is mall when $\hat{p}_{\mathbf{y}}$ is large, i.e. model is more confident



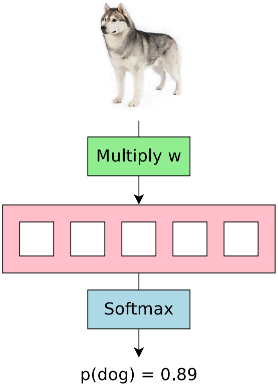
Cross-Entropy Loss

- $\mathcal{L}(\hat{p}, \mathbf{y}) = -\log \hat{p}_{\mathbf{y}}$
  - One of the most important loss functions of deep learning.
  - $\mathcal{L}$ is mall when $\hat{p}_{\mathbf{y}}$ is large, i.e. model is more confident
  - But $\mathcal{L}$ is *always positive*



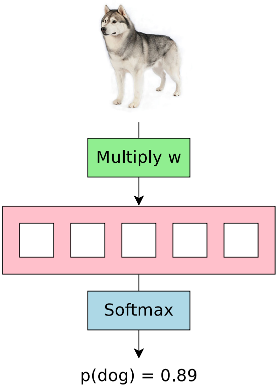
Cross-Entropy Loss

- $\mathcal{L}(\hat{p}, \mathbf{y}) = -\log \hat{p}_{\mathbf{y}}$
  - One of the most important loss functions of deep learning.
  - $\mathcal{L}$ is mall when $\hat{p}_{\mathbf{y}}$ is large, i.e. model is more confident
  - But $\mathcal{L}$ is *always positive*
  - When $\hat{p}_{\mathbf{y}}$ is large, $\hat{p}_{\neq \mathbf{y}}$ are small



Cross-Entropy Loss

- $\mathcal{L}(\hat{p}, \mathbf{y}) = -\log \hat{p}_{\mathbf{y}}$
  - One of the most important loss functions of deep learning.
  - $\mathcal{L}$ is mall when $\hat{p}_{\mathbf{y}}$ is large, i.e. model is more confident
  - But $\mathcal{L}$ is *always positive*
  - When $\hat{p}_{\mathbf{y}}$ is large, $\hat{p}_{\neq \mathbf{y}}$ are small
  - Differentiable. Recall $\hat{p}_i = \exp\{\ell_i\} / \sum_j \exp\{\ell_j\}$
    - ▷ Important for learning.



Outline

CarnegieMellon

The Story So Far

- **The problem:** want to learn a function $\mathbf{y} = \mathbf{f}(\mathbf{x})$

The Story So Far

- **The problem:** want to learn a function $\mathbf{y} = \mathbf{f}(\mathbf{x})$
- **The “data” solution:**

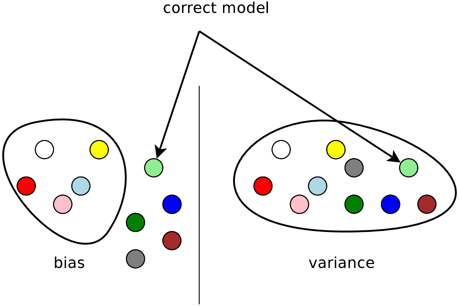
The Story So Far

Carnegie Mellon

- **The problem:** want to learn a function $\mathbf{y} = \mathbf{f}(\mathbf{x})$
- **The "data" solution:**
  - Collect *data* $\left\{\left(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}\right), \left(\mathbf{x}^{(2)}, \mathbf{y}^{(2)}\right), ..., \left(\mathbf{x}^{(N)}, \mathbf{y}^{(N)}\right)\right\}$

The Story So Far

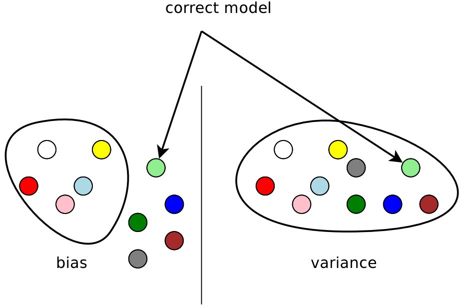
- **The problem:** want to learn a function $\mathbf{y} = \mathbf{f}(\mathbf{x})$
- **The "data" solution:**
  - Collect *data* $\left\{\left(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}\right), \left(\mathbf{x}^{(2)}, \mathbf{y}^{(2)}\right), ..., \left(\mathbf{x}^{(N)}, \mathbf{y}^{(N)}\right)\right\}$
  - Come up with a set of *hypotheses* $\mathcal{H} = \{\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_{|\mathcal{H}|}\}$



The Story So Far

Carnegie Mellon

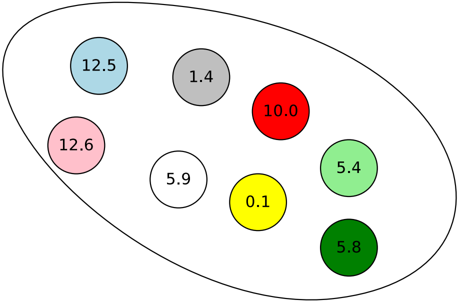
- **The problem:** want to learn a function $\mathbf{y} = \mathbf{f}(\mathbf{x})$
- **The "data" solution:**
  - Collect *data* $\left\{\left(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}\right), \left(\mathbf{x}^{(2)}, \mathbf{y}^{(2)}\right), ..., \left(\mathbf{x}^{(N)}, \mathbf{y}^{(N)}\right)\right\}$
  - Come up with a set of *hypotheses* $\mathcal{H} = \{\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_{|\mathcal{H}|}\}$



  - Come up with a *loss function* $\mathcal{L} : \mathbf{f} \in \mathcal{H} \to \mathbb{R}$, which tells you how bad is a hypothesis $\mathbf{f}$.

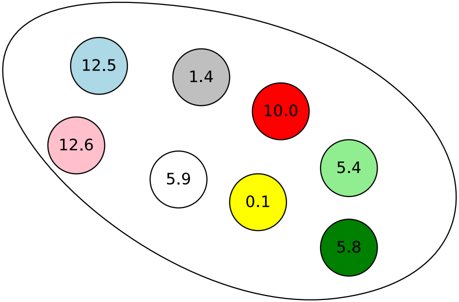
Learning

- Learning is the process of finding $\mathbf{f} \in \mathcal{H}$ that minimizes $\mathcal{L}(\mathbf{f})$



Learning

- Learning is the process of finding $\mathbf{f} \in \mathcal{H}$ that minimizes $\mathcal{L}(\mathbf{f})$



- In math: $\mathbf{f}^* = \operatorname{argmin}_{\mathbf{f} \in \mathcal{H}} \mathcal{L}(\mathbf{f})$

Case Study: Linear Regression

CarnegieMellon

- **Data:** $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$
  - For simplicity: $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}$.

Case Study: Linear Regression

CarnegieMellon

- **Data:** $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$
  - For simplicity: $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}$.
- Hypotheses: $\mathcal{H} = \{\mathbf{f}(\mathbf{x}; a, b) = a \cdot \mathbf{x} + b\}$

Case Study: Linear Regression

CarnegieMellon

- **Data:** $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$
  - For simplicity: $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}$.
- Hypotheses: $\mathcal{H} = \{\mathbf{f}(\mathbf{x}; a, b) = a \cdot \mathbf{x} + b\}$
- Loss function:

$$\mathcal{L}(\mathbf{f}) = \sum_{i=1}^{N} (\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i)^2$$

Case Study: Linear Regression

- **Data:** $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$
  - For simplicity: $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}$.
- Hypotheses: $\mathcal{H} = \{\mathbf{f}(\mathbf{x}; a, b) = a \cdot \mathbf{x} + b\}$
- Loss function:

$$\mathcal{L}(\mathbf{f}) = \sum_{i=1}^{N} (\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i)^2$$

- **Learning:** try to find

$$a^*, b^* = \underset{a,b}{\operatorname{argmin}} \sum_{i=1}^{N} (\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i)^2$$

Case Study: Linear Regression

CarnegieMellon

- **Learning:** try to find

$$a^*, b^* = \operatorname*{argmin}_{a,b} \sum_{i=1}^{N} (\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i)^2$$

Case Study: Linear Regression

CarnegieMellon

- **Learning:** try to find

$$
\begin{aligned}
a^*, b^* &= \underset{a,b}{\operatorname{argmin}} \sum_{i=1}^{N} (\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i)^2 \\
&= \underset{a,b}{\operatorname{argmin}} \sum_{i=1}^{N} (a\mathbf{x}_i + b - \mathbf{y}_i)^2
\end{aligned}
$$

Case Study: Linear Regression

CarnegieMellon

- **Learning:** try to find

$$a^*, b^* = \underset{a,b}{\operatorname{argmin}} \sum_{i=1}^{N} (\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i)^2$$
$$= \underset{a,b}{\operatorname{argmin}} \sum_{i=1}^{N} (a\mathbf{x}_i + b - \mathbf{y}_i)^2$$

- **Approach:** Find *gradients* and set to 0

$$\ell(a, b) = \sum_{i=1}^{N} (a\mathbf{x}_i + b - \mathbf{y}_i)^2$$

Case Study: Linear Regression

CarnegieMellon

- **Learning:** try to find

$$a^*, b^* = \underset{a,b}{\text{argmin}} \sum_{i=1}^{N} (\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i)^2$$
$$= \underset{a,b}{\text{argmin}} \sum_{i=1}^{N} (a\mathbf{x}_i + b - \mathbf{y}_i)^2$$

- **Approach:** Find *gradients* and set to 0

$$\ell(a, b) = \sum_{i=1}^{N} (a\mathbf{x}_i + b - \mathbf{y}_i)^2$$
$$\frac{\partial \ell}{\partial a} = \sum_{i=1}^{N} 2\mathbf{x}_i(a\mathbf{x}_i + b - \mathbf{y}_i)$$

Case Study: Linear Regression

- **Learning:** try to find

$$a^*, b^* = \underset{a,b}{\operatorname{argmin}} \sum_{i=1}^{N} (\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i)^2$$
$$= \underset{a,b}{\operatorname{argmin}} \sum_{i=1}^{N} (a\mathbf{x}_i + b - \mathbf{y}_i)^2$$

- **Approach:** Find *gradients* and set to 0

$$\ell(a,b) = \sum_{i=1}^{N} (a\mathbf{x}_i + b - \mathbf{y}_i)^2$$
$$\frac{\partial \ell}{\partial a} = \sum_{i=1}^{N} 2\mathbf{x}_i(a\mathbf{x}_i + b - \mathbf{y}_i)$$
$$\frac{\partial \ell}{\partial b} = \sum_{i=1}^{N} 2(a\mathbf{x}_i + b - \mathbf{y}_i)$$

Case Study: Linear Regression

- **Approach:** Find *gradients* and set to 0

$$\frac{\partial \ell}{\partial a} = \sum_{i=1}^{N} \mathbf{x}_i (a\mathbf{x}_i + b - \mathbf{y}_i)$$

$$\frac{\partial \ell}{\partial b} = \sum_{i=1}^{N} (a\mathbf{x}_i + b - \mathbf{y}_i)$$

Case Study: Linear Regression

CarnegieMellon

- **Approach:** Find *gradients* and set to 0

$$\frac{\partial \ell}{\partial a} = \sum_{i=1}^{N} \mathbf{x}_i(a\mathbf{x}_i + b - \mathbf{y}_i)$$
$$= a\sum_{i=1}^{N} \mathbf{x}_i^2 + b\sum_{i=1}^{N} \mathbf{x}_i - \sum_{i=1}^{N} \mathbf{x}_i\mathbf{y}_i$$
$$\frac{\partial \ell}{\partial b} = \sum_{i=1}^{N}(a\mathbf{x}_i + b - \mathbf{y}_i)$$

- This is a linear system in a, b. You can solve it!

Case Study: Linear Regression

CarnegieMellon

- **Approach:** Find *gradients* and set to 0

$$
\begin{aligned}
\frac{\partial \ell}{\partial a} &= \sum_{i=1}^{N} \mathbf{x}_i(a\mathbf{x}_i + b - \mathbf{y}_i) \\
&= a\sum_{i=1}^{N} \mathbf{x}_i^2 + b\sum_{i=1}^{N} \mathbf{x}_i - \sum_{i=1}^{N} \mathbf{x}_i\mathbf{y}_i \\
\frac{\partial \ell}{\partial b} &= \sum_{i=1}^{N} (a\mathbf{x}_i + b - \mathbf{y}_i) \\
&= a\sum_{i=1}^{N} \mathbf{x}_i + b \cdot N - \sum_{i=1}^{N} \mathbf{y}_i
\end{aligned}
$$

- This is a linear system in a, b. You can solve it!

Learning is not Always Easy!

Carnegie Mellon

- Sometimes setting gradients to 0 does not work
  - The system does not have any solution
  - Too complicated to solve
- Neural networks: millions of parameters
  - or more extreme... (Shazeer et al., 2017)

	Test Perplexity 10 epochs	Test Perplexity 100 epochs	#Parameters excluding embedding and softmax layers
Best Published Results	34.7	30.6	151 million
Low-Budget MoE Model	34.1		4303 million
Medium-Budget MoE Model	31.3		4313 million
High-Budget MoE Model	**28.0**		4371 million

- You simply *cannot* find $\mathbf{f}^* = \operatorname{argmin}_{\mathbf{f} \in \mathcal{H}} \mathcal{L}(\mathbf{f})$
- Rely on *numerical optimization algorithms.*

Gradient Descent

- Data: $\left\{\left(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}\right), \left(\mathbf{x}^{(2)}, \mathbf{y}^{(2)}\right), ..., \left(\mathbf{x}^{(N)}, \mathbf{y}^{(N)}\right)\right\}$
- Hypotheses: $\mathcal{H} = \{\mathbf{f}(\mathbf{x}; \theta) : \theta \in \mathbb{R}^D\}$
- Loss function

$$\ell(\theta) = \sum_{i=1}^{N} \mathcal{L}(\mathbf{f}(\mathbf{x}_i; \theta), \mathbf{y}_i)$$

Gradient Descent

- Data: $\left\{\left(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}\right), \left(\mathbf{x}^{(2)}, \mathbf{y}^{(2)}\right), ..., \left(\mathbf{x}^{(N)}, \mathbf{y}^{(N)}\right)\right\}$
- Hypotheses: $\mathcal{H} = \{\mathbf{f}(\mathbf{x};\theta) : \theta \in \mathbb{R}^D\}$
- Loss function

$$\ell(\theta) = \sum_{i=1}^{N} \mathcal{L}(\mathbf{f}(\mathbf{x}_i;\theta), \mathbf{y}_i)$$

- Gradient Descent (GD) algorithm

Gradient Descent

- Data: $\left\{ \left(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}\right), \left(\mathbf{x}^{(2)}, \mathbf{y}^{(2)}\right), ..., \left(\mathbf{x}^{(N)}, \mathbf{y}^{(N)}\right) \right\}$
- Hypotheses: $\mathcal{H} = \{\mathbf{f}(\mathbf{x}; \theta) : \theta \in \mathbb{R}^D\}$
- Loss function

$$\ell(\theta) = \sum_{i=1}^{N} \mathcal{L}(\mathbf{f}(\mathbf{x}_i; \theta), \mathbf{y}_i)$$

- Gradient Descent (GD) algorithm
  - Randomly *initialize* $\theta_0 \in \mathbb{R}^D$.
  - Repeat until convergence
    - Compute the gradient: $\nabla_\theta \ell(\theta^{(t)})$
    - Update: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla_\theta \ell(\theta^{(t)})$

Problems with Gradient Descent

Carnegie Mellon

- We have to compute

$$\nabla_\theta \ell(\theta^{(t)}) = \nabla_\theta \sum_{i=1}^{N} \mathcal{L}(\mathbf{f}(\mathbf{x}_i;\theta), \mathbf{y}_i)$$

Problems with Gradient Descent

CarnegieMellon

- We have to compute

$$\nabla_\theta \ell(\theta^{(t)}) = \nabla_\theta \sum_{i=1}^{N} \mathcal{L}(\mathbf{f}(\mathbf{x}_i;\theta), \mathbf{y}_i) = \sum_{i=1}^{N} \nabla_\theta \mathcal{L}(\mathbf{f}(\mathbf{x}_i;\theta), \mathbf{y}_i)$$

Problems with Gradient Descent

- We have to compute

$$\nabla_\theta \ell(\theta^{(t)}) = \nabla_\theta \sum_{i=1}^{N} \mathcal{L}(\mathbf{f}(\mathbf{x}_i;\theta), \mathbf{y}_i) = \sum_{i=1}^{N} \nabla_\theta \mathcal{L}(\mathbf{f}(\mathbf{x}_i;\theta), \mathbf{y}_i)$$

- It's *very slow* if N is large.
  - ImageNet: $N = 1,200,000$
  - English-German translation: $N = 4,500,000$
  - Google 1-billion-words data: $N = 1,000,000,000$
  - Human Genes: $N = ???$

Solution: Stochastic Gradient Descent

CarnegieMellon

- Data: $\left\{\left(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}\right), \left(\mathbf{x}^{(2)}, \mathbf{y}^{(2)}\right), ..., \left(\mathbf{x}^{(N)}, \mathbf{y}^{(N)}\right)\right\}$
- Hypotheses: $\mathcal{H} = \{\mathbf{f}(\mathbf{x}; \theta) : \theta \in \mathbb{R}^D\}$
- Loss function

$$\ell(\theta) = \sum_{i=1}^{N} \mathcal{L}(\mathbf{f}(\mathbf{x}_i; \theta), \mathbf{y}_i)$$

- Stochastic Gradient Descent (SGD) algorithm
  - Randomly *initialize* $\theta_0 \in \mathbb{R}^D$.
  - Repeat until convergence
    - Sample $(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_B, \mathbf{y}_B)$ from your data
    - Compute the stochastic gradient:

$$\hat{\nabla}_\theta = \sum_{i=1}^{B} \nabla_\theta \mathcal{L}(\mathbf{f}(\mathbf{x}_i; \theta), \mathbf{y}_i)$$

    - Update: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \hat{\nabla}_\theta$

Notes on Stochastic Gradient Descent

Carnegie Mellon

- Data: $\left\{\left(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}\right), \left(\mathbf{x}^{(2)}, \mathbf{y}^{(2)}\right), ..., \left(\mathbf{x}^{(N)}, \mathbf{y}^{(N)}\right)\right\}$
  - Randomly *initialize* $\theta_0 \in \mathbb{R}^D$.
  - Repeat until convergence
    - Sample $(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_B, \mathbf{y}_B)$ from your data
    - Compute the stochastic gradient:

$$\hat{\nabla}_\theta = \sum_{i=1}^{B} \nabla_\theta \mathcal{L}(\mathbf{f}(\mathbf{x}_i; \theta), \mathbf{y}_i)$$

    - Update: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \hat{\nabla}_\theta$
- You **have to** *really* sample $(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_B, \mathbf{y}_B)$ from your data

Notes on Stochastic Gradient Descent

- Data: $\left\{\left(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}\right), \left(\mathbf{x}^{(2)}, \mathbf{y}^{(2)}\right), ..., \left(\mathbf{x}^{(N)}, \mathbf{y}^{(N)}\right)\right\}$
  - Randomly *initialize* $\theta_0 \in \mathbb{R}^D$.
  - Repeat until convergence
    - Sample $(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_B, \mathbf{y}_B)$ from your data
    - Compute the stochastic gradient:

$$\hat{\nabla}_\theta = \sum_{i=1}^{B} \nabla_\theta \mathcal{L}(\mathbf{f}(\mathbf{x}_i; \theta), \mathbf{y}_i)$$

    - Update: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \hat{\nabla}_\theta$
- You **have to** *really* sample $(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_B, \mathbf{y}_B)$ from your data
  - otherwise, your gradient is *biased.*
  - References: Lyapunov functions; Leon Bottou's PhD thesis.

Variations of SGD

- SGD so far:
    - Randomly *initialize* $\theta_0 \in \mathbb{R}^D$.
    - Repeat until convergence
        - Sample $(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_B, \mathbf{y}_B)$ from your data
        - Compute the stochastic gradient:

$$\hat{\nabla}_\theta = \sum_{i=1}^{B} \nabla_\theta \mathcal{L}(\mathbf{f}(\mathbf{x}_i; \theta), \mathbf{y}_i)$$

        - Update: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \hat{\nabla}_\theta$

Variations of SGD

- SGD so far:
  - Randomly *initialize* $\theta_0 \in \mathbb{R}^D$.
  - Repeat until convergence
    - Sample $(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_B, \mathbf{y}_B)$ from your data
    - Compute the stochastic gradient:

$$\hat{\nabla}_\theta = \sum_{i=1}^{B} \nabla_\theta \mathcal{L}(\mathbf{f}(\mathbf{x}_i; \theta), \mathbf{y}_i)$$

    - Update: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \hat{\nabla}_\theta$
- In TensorFlow: `tf.train.GradientDescentOptimizer`

Variations of SGD

Carnegie Mellon

- SGD so far:
  - Randomly *initialize* $\theta_0 \in \mathbb{R}^D$.
  - Repeat until convergence
    - Sample $(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_B, \mathbf{y}_B)$ from your data
    - Compute the stochastic gradient:

$$\hat{\nabla}_\theta = \sum_{i=1}^{B} \nabla_\theta \mathcal{L}(\mathbf{f}(\mathbf{x}_i; \theta), \mathbf{y}_i)$$

    - Update: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \hat{\nabla}_\theta$
- In TensorFlow: `tf.train.GradientDescentOptimizer`
- Assuming that you have a gradient ∇_θ

Variations of SGD

Carnegie Mellon

- SGD so far:
  - Randomly *initialize* $\theta_0 \in \mathbb{R}^D$.
  - Repeat until convergence
    - Sample $(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_B, \mathbf{y}_B)$ from your data
    - Compute the stochastic gradient:

$$\hat{\nabla}_\theta = \sum_{i=1}^{B} \nabla_\theta \mathcal{L}(\mathbf{f}(\mathbf{x}_i; \theta), \mathbf{y}_i)$$

    - Update: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \hat{\nabla}_\theta$
- In TensorFlow: `tf.train.GradientDescentOptimizer`
- Assuming that you have a gradient ∇_θ
- There are many other ways to update, more than just

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla_\theta$$

Momentum

- Keeps a *running average* of ∇_θ

$$v^{(t+1)} \leftarrow (1-m)v^{(t)} + m \cdot \nabla_\theta$$
$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta v^{(t+1)}$$

- You have to choose m, η
- In TensorFlow: `tf.train.MomentumOptimizer`

Adaptive Stochastic Gradient Descent (AdaGrad)

- Keeps a *running average* of ∇_θ^2

$$v^{(t+1)} \leftarrow (1-m)v^{(t)} + m \cdot \nabla_\theta^2$$
$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \cdot \frac{\nabla_\theta}{\sqrt{v^{(t+1)}}}$$

- You have to choose m, η
- In TensorFlow: `tf.train.AdagradOptimizer`

Outline

Carnegie Mellon

1. The Learning Problem
2. Data and Label
3. Modeling
4. Error and Loss Function
5. Learning
6. Overfitting and Regularization

Is it good to minimize loss function?

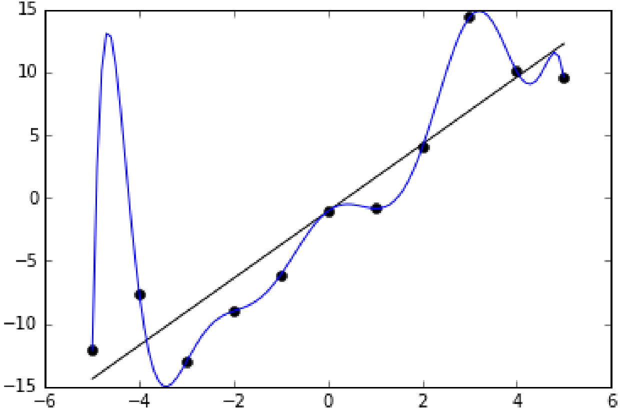


Image credit: wikipedia

Is it good to minimize loss function?

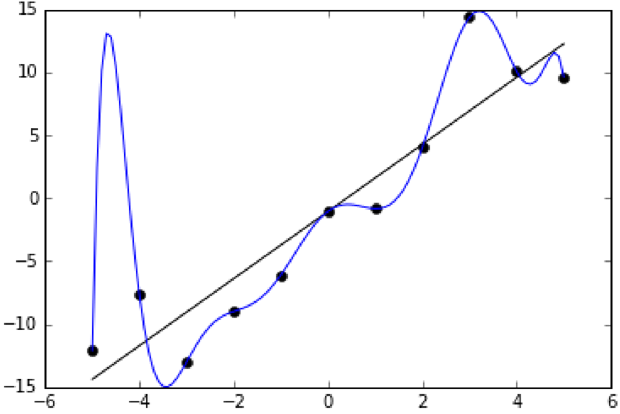


Image credit: wikipedia

- NO!

Regularization

Carnegie Mellon

- Suppose your loss function is $\ell(\theta)$
- Change it into

$$\ell^{\text{reg}}(\theta) = \ell(\theta) + \beta \left\|\theta\right\|$$

- You have to choose β
- The term $\beta \left\|\theta\right\|^2$ is called the ℓ-2 regularization.

Regularization

- There are other choices
- ℓ-1 regularization

$$\ell^{\text{reg}}(\theta) = \ell(\theta) + \beta|\theta|^2$$

- ℓ-p regularization

$$\ell^{\text{reg}}(\theta) = \ell(\theta) + \beta\left(\sum_{i=1}^{D} \theta_i^p\right)^{1/p}$$

- DropOut, DropConnect, Variational Regularization

$$\ell^{\text{reg}}(\theta) = \ell(g(\theta)),$$

where $g(\theta)$ "corrupts" θ.