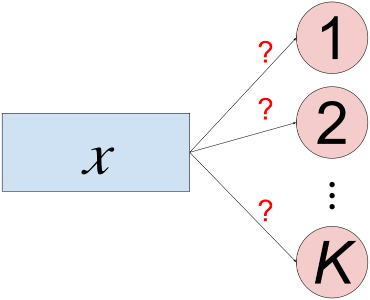
Lectures on Natural Language Processing

2. Classification

Karl Stratos

The Classification Problem

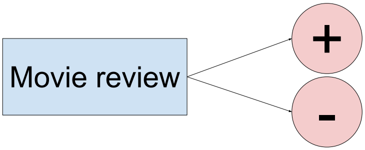
- Input space $\mathcal{X}$, label space $\mathcal{Y} = \{1 \dots K\}$
- Given input $x \in \mathcal{X}$, predict corresponding label/class $y \in \mathcal{Y}$



- This is an abstraction: nature of x, y depends on the task
  - In NLP, x is often text.
  - In vision, x is an image; in speech, x is a signal. In robotics, x is a sensory value.

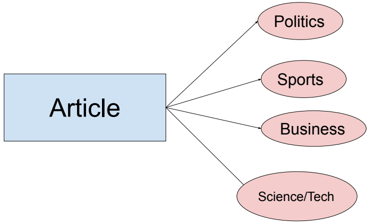
Examples in NLP

Sentiment analysis (e.g., $K = 2$)



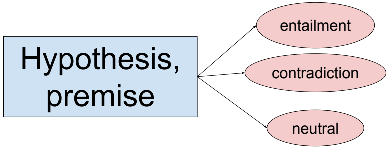
(positive or negative?)

Topic classification (e.g., $K = 4$)



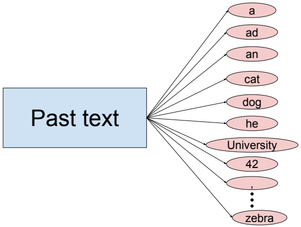
(what's the article about?)

Natural language inference (e.g., $K = 3$)



(does premise follow from hypothesis?)

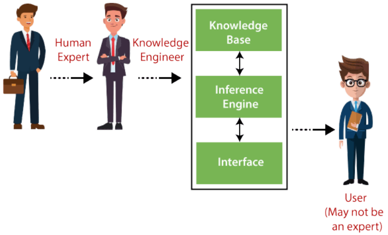
Language modeling (e.g., $K = 120,000$)



(given past text, what's the next word?)

Old Days: Expert Systems

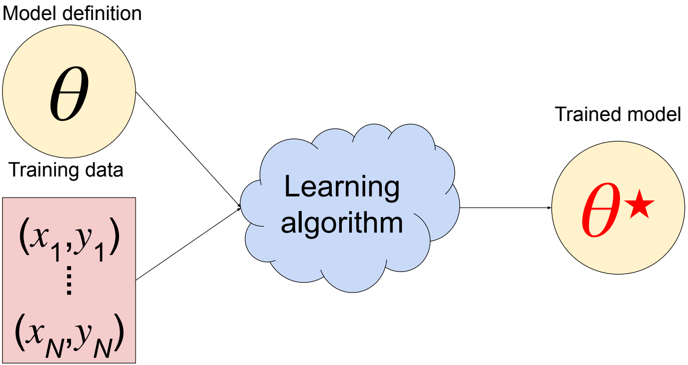
- Human experts/engineers come up with a bunch of **if-then rules** ("knowledge base")
- An "inference engine" applies logical rules to answer questions
- No data, no learning



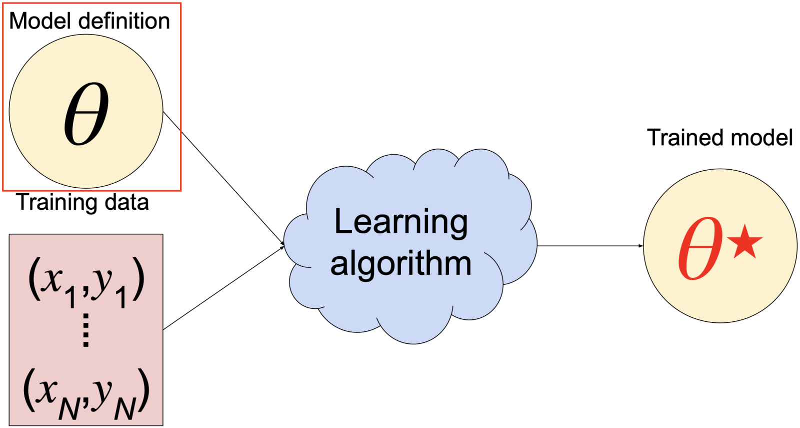
First practical AI applications (e.g., translation, ELIZA), but obvious limitations ⇒ AI winter (early 1990s)

- Never enough rules, hard to go beyond "shallow" behavior
- Heavy dependence on human expertise
- Can't handle truly complex tasks (e.g., autonomous driving)
- No generalization between tasks: one-trick ponies

Principles of Machine Learning



- **No explicit rule construction**: relationship between input x and output y inferred from training data
- **Task-agnostic**: no matter what the task is, the principles stay the same


Model definition
θ
Training data
(x_1, y_1)
$\vdots$
(x_N, y_N)
Learning algorithm
Trained model
$\theta^\star$

Basic ML Concepts: Model and Parameter

- A **model** with **parameter** θ is a mapping f_θ ("computational recipe") whose behavior is controlled by the choice of $\theta \in \Theta$.
- This defines a **model class** (or **hypothesis class**):

$$\mathcal{F} = \{f_\theta : \theta \in \Theta\}$$

- Example: linear models $f_\theta : \mathbb{R}^3 \to \mathbb{R}^2$

$$\Theta := \left\{(W, b) \in \mathbb{R}^{3\times 2} \times \mathbb{R}^2\right\} \qquad \text{(parameter space)}$$
$$f_\theta(x) := W^\top x + b \qquad \text{(model definition)}$$

Every choice of (W, b) yields a particular (linear) mapping

$$W = \begin{bmatrix} 3.14 & -9 \\ -2 & 7 \\ 1.1 & 5 \end{bmatrix} \quad b = \begin{bmatrix} 10 \\ -3 \end{bmatrix} \quad \Longrightarrow \quad f_\theta\left(\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}\right) = \begin{bmatrix} 3.14 z_1 - 2 z_2 + 1.1 z_3 + 10 \\ -9 z_1 + 7 z_2 + 5 z_3 - 3 \end{bmatrix}$$

The Classification Model

General classifier with parameter θ

$$\textbf{score}_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$$

Given any $x \in \mathcal{X}$, the model predicts

$$\boxed{\hat{y}_\theta(x) = \arg\max_{y \in \mathcal{Y}} \ \textbf{score}_\theta(x, y)}$$

$\textbf{score}_\theta(x, y)$ (also called **logit**) represents how likely x has label y.

No matter how complicated the model becomes,
it will always take this form!

Example: Topic Classification with Linear Classifiers

- Input document represented as a d-dimensional vector $x \in \mathbb{R}^d$
  - E.g., "bag-of-words" $x = (1, 0, 0, 1, \ldots, 0) \in \{0, 1\}^d$ indicating the presence/absence of words
- Model parameter: for each topic $y = 1 \ldots K$, have a vector w_y and a scalar b_y

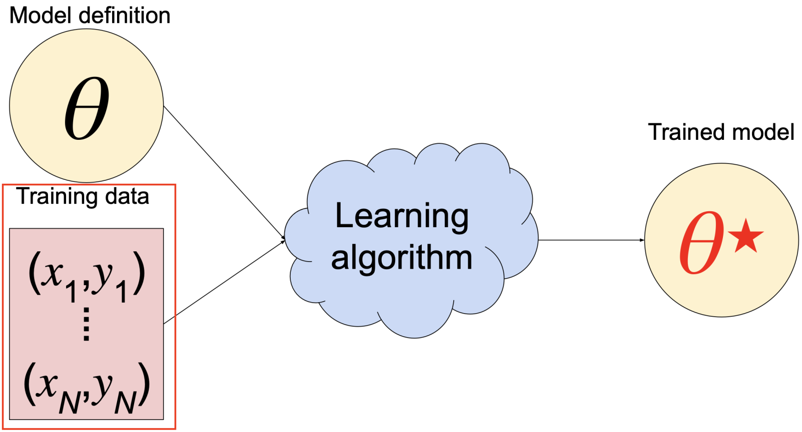
$$\theta = \left\{ w_y \in \mathbb{R}^d,\ b_y \in \mathbb{R} : y = 1 \ldots K \right\}$$

- **Model definition.** Score of document x and topic y computed as

$$\textbf{score}_\theta(x, y) = w_y^\top x + b_y \in \mathbb{R}$$

- **Model prediction.** Given any $x_{\text{new}} \in \mathbb{R}^d$, predict

$$\hat{y}_\theta(x_{\text{new}}) = \operatorname*{arg\,max}_{y=1\ldots K}\ w_y^\top x_{\text{new}} + b_y \in \{1 \ldots K\}$$


Model definition
θ
Training data
(x_1, y_1)
$\vdots$
(x_N, y_N)
Learning algorithm
Trained model
$\theta^\star$

Naively Fitting the Training Data

Input.

- Functional definition of $\textbf{score}_{\theta}$ with parameter space $\theta \in \Theta$ defining $\hat{y}_{\theta}(x) = \arg\max_{y \in \mathcal{Y}} \textbf{score}_{\theta}(x, y)$
- Input-label pairs $(x_1, y_1) \ldots (x_N, y_N) \in \mathcal{X} \times \mathcal{Y}$

Training. Find a model that makes the smallest number of classification mistakes

$$\tilde{\theta} = \arg\min_{\theta \in \Theta} \sum_{i=1}^{N} \underbrace{\mathbb{1}(\hat{y}_{\theta}(x_i) \neq y_i)}_{\text{“0-1 loss”: 1 if incorrect, 0 if correct}}$$

What is $\tilde{\theta}$? Is it a “good model”?

Memorization vs Generalization

Memorization: Any $\tilde{\theta}$ such that

$$\hat{y}_{\tilde{\theta}}(x_i) = y_i \quad \forall i = 1 \ldots N$$

has zero training loss, but that alone does not imply generalization.

Generalization: accurately classify future data

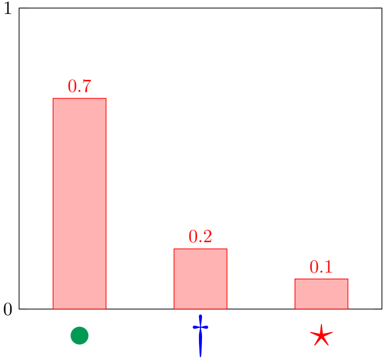
$$x_{\text{new}} \mapsto \;?$$

- This is our goal, but is it even feasible? (E.g., consider an adversarial setting)
- What sort of assumptions do we need?

Review: Probability Distribution

A **distribution** p_X over random variable $X \in \{\bullet, \dagger, \star\}$ satisfies

1. $p_X(\bullet),\ p_X(\dagger),\ p_X(\star) \geq 0$
2. $p_X(\bullet) + p_X(\dagger) + p_X(\star) = 1$

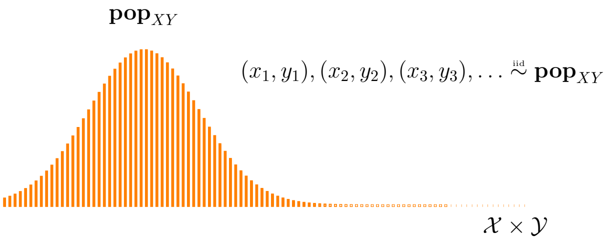


Independently and identically distributed (iid) samples

$\bullet\ \star\ \bullet\ \bullet\ \star\ \bullet\ \bullet\ \dagger\ \bullet\ \star\ \bullet\ \dagger\ \bullet\ \bullet\ \star\ \dagger\ \dagger\ \bullet\ \dagger\ \bullet\ \bullet\ \bullet\ \dagger\ \bullet\ \dagger\ \cdots$

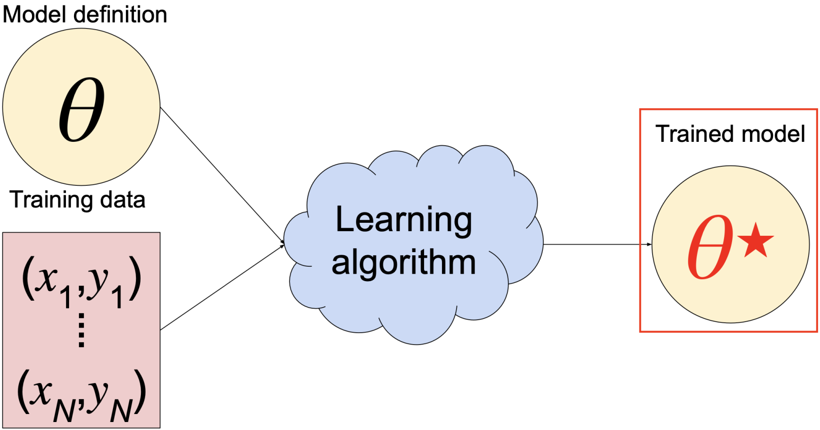
Population Distribution

Foundational assumption in ML: data comes randomly from a **population distribution** $\mathbf{pop}_{XY}$ over $(X, Y) \in \mathcal{X} \times \mathcal{Y}$



(Implies marginal $\mathbf{pop}_X, \mathbf{pop}_Y$ and conditional $\mathbf{pop}_{Y|X}, \mathbf{pop}_{X|Y}$ distributions.) Generalization of any classifier $f : \mathcal{X} \to \mathcal{Y}$ now measured by the *expected* loss (aka. "risk")

$$\Pr_{(x,y)\sim\mathbf{pop}_{XY}} \left(f(x) \neq y\right)$$


Model definition
θ
Training data
(x1,y1)
(xN,yN)
Learning algorithm
Trained model
θ★

Optimal Classification by Density Estimation

Goal. Estimate the **(Bayes) optimal classifier**

$$f^\star := \arg\min_{f\colon \mathcal{X}\to\mathcal{Y}} \Pr_{(x,y)\sim\mathbf{pop}_{XY}} (f(x) \neq y)$$

Optimal Classification by Density Estimation

Goal. Estimate the **(Bayes) optimal classifier**

$$f^\star := \operatorname*{arg\,min}_{f:\,\mathcal{X}\to\mathcal{Y}} \Pr_{(x,y)\sim\mathbf{pop}_{XY}} (f(x) \neq y)$$

Claim. For any $x \in \mathcal{X}$

$$f^\star(x) = \operatorname*{arg\,max}_{y\in\mathcal{Y}} \; \mathbf{pop}_{Y|X}(y|x)$$

Optimal Classification by Density Estimation

Goal. Estimate the **(Bayes) optimal classifier**

$$f^\star := \arg\min_{f\colon \mathcal{X} \to \mathcal{Y}} \Pr_{(x,y) \sim \mathbf{pop}_{XY}} (f(x) \neq y)$$

Claim. For any $x \in \mathcal{X}$

$$f^\star(x) = \arg\max_{y \in \mathcal{Y}} \mathbf{pop}_{Y|X}(y|x)$$

New goal. Estimate $\mathbf{pop}_{Y|X}$

- Optimal classifier implied by the possession of $\mathbf{pop}_{Y|X}$
- Training = conditional density estimation

Density Estimation by Cross-Entropy Minimization

Cross-entropy between distributions p, q over set S

$$H(p, q) := -\sum_{x \in S} p(x) \times \log q(x) \geq 0$$

Expected number of bits to encode the behavior of p using q

Consistency of cross-entropy

$$q^\star = \arg\min_q \; H(p, q) \quad \Leftrightarrow \quad q^\star(x) = p(x) \quad \forall x \in S$$

Game plan: convert model scores into a distribution, then estimate $\mathbf{pop}_{Y|X}$ by minimizing cross-entropy

The Softmax Function

Any K values can be converted into a distribution over $\{1 \ldots K\}$ by the **softmax function** $\mathrm{softmax} : \mathbb{R}^K \rightarrow [0,1]^K$:

$$\underbrace{\mathrm{softmax}_k(u)}_{\text{prob. of } k\text{-th item given } u \in \mathbb{R}^K} := \frac{\exp(u_k)}{\sum_{l=1}^{K} \exp(u_l)}$$

Examples

$$\mathrm{softmax}(-0.23, 1.51, -2.11) = (0.15, 0.83, 0.02)$$
$$\mathrm{softmax}(-1000, -1000, 1.00, 2.00) = (0.00, 0.00, 0.27, 0.73)$$

Shift-invariant: for any $c \in \mathbb{R}$ (elementwise addition)

$$\mathrm{softmax}(u + c) = \mathrm{softmax}(u)$$

The Probabilistic Classification Model

Any model scoring (X, Y) has the corresponding **conditional distribution** over Y given X

$$p_\theta(y|x) := \underbrace{\frac{\exp(\textbf{score}_\theta(x, y))}{\sum_{y' \in \mathcal{Y}} \exp(\textbf{score}_\theta(x, y'))}}_{\mathrm{softmax}_y(\textbf{score}_\theta(x,1), \ldots, \textbf{score}_\theta(x,K))}$$

Probabilities only needed for training, prediction unchanged

$$\arg\max_{y \in \mathcal{Y}} \; p_\theta(y|x) \equiv \arg\max_{y \in \mathcal{Y}} \; \textbf{score}_\theta(x, y)$$

The Cross-Entropy Loss

Estimate $p_\theta(y|x) \approx \mathbf{pop}_{Y|X}(y|x)$ by minimizing cross-entropy

$$\theta^\star = \operatorname*{arg\,min}_{\theta \in \Theta} \mathop{\mathbf{E}}_{(x,y) \sim \mathbf{pop}_{XY}} \left[-\log p_\theta(y|x) \right]$$

("fundamental equation of deep learning" –David McAllester)

In practice, minimize the empirical objective (**empirical risk minimization**)

$$\hat{\theta} = \operatorname*{arg\,min}_{\theta \in \Theta} \; -\frac{1}{N} \sum_{i=1}^{N} \log p_\theta(y_i|x_i)$$

based on iid samples $(x_1, y_1) \ldots (x_N, y_N) \sim \mathbf{pop}_{XY}$

Aside: Maximum Likelihood Estimation (MLE)

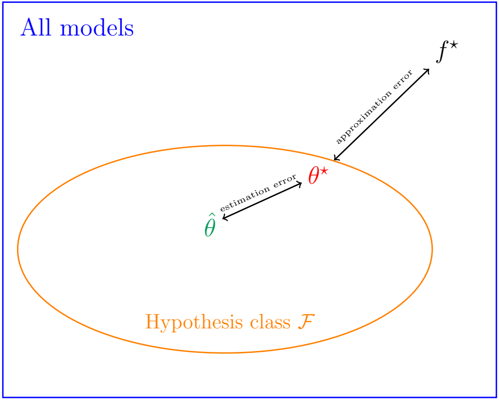
MLE: use θ that makes the data most likely

$$\hat{\theta} = \arg\max_{\theta \in \Theta} \; \frac{1}{N} \sum_{i=1}^{N} \log p_{\theta}(y_i | x_i)$$

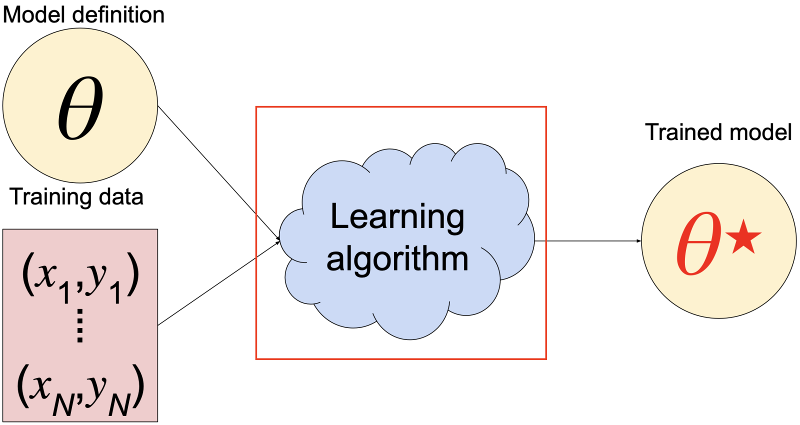
The most important estimator in statistics

- As $N \to \infty$, unbiased and efficient (i.e., minimum variance)
- Cross-entropy minimization *is* MLE!
- We will say "cross-entropy minimizer" and "maximum-likelihood estimator" interchangeably.

Approximation vs Estimation Error



- **Approximation error**: due to limited model expressiveness, remains no matter how much data we have
- **Estimation error**: due to limited data, can reduce by getting more data

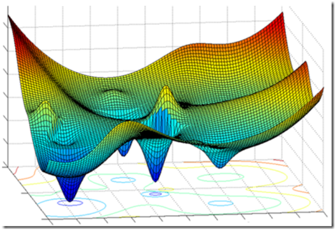

Model definition
θ
Training data
(x_1, y_1)
$\vdots$
(x_N, y_N)
Learning algorithm
Trained model
$\theta^\star$

Minimizing the Empirical Cross-Entropy Loss

Training data: $(x_1, y_1) \ldots (x_N, y_N) \overset{\text{iid}}{\sim} \mathbf{pop}_{XY}$

$$\widehat{J}_N(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log p_\theta(y_i|x_i)$$

- Goal: find θ with small $\widehat{J}_N(\theta)$
- Universal approach: **gradient descent**



Gradient of a Function

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be any differentiable function.
- The i-th **partial derivative** of f is the derivative of f when viewed as a function of the i-th variable only: $\frac{\partial f(\theta)}{\partial \theta_i} \in \mathbb{R}$.
- The **gradient** of f at $\theta \in \mathbb{R}^d$ is the vector

$$\nabla f(\theta) = \left(\frac{\partial f(\theta)}{\partial \theta_1}, \ldots, \frac{\partial f(\theta)}{\partial \theta_d} \right) \in \mathbb{R}^d$$

- Points to the direction of increase of f at location $\theta \in \mathbb{R}^d$
  - Magnitude $||\nabla f(\theta)||$: rate of change
  - $\nabla f(\theta) = 0_d$ if θ is a stationary point
- $d = 1$ example $f(\theta) = (\theta - 3)^2$, $f'(\theta) = 2\theta - 6$
  - At $\theta = 5$: Increasing to the right at a rate of 4
  - Stationary point $\theta = 3$

Minimizing a Local Approximation

Taylor's theorem: First-order approximation of $f : \mathbb{R}^d \to \mathbb{R}$ around $\theta_0 \in \mathbb{R}^d$

$$f(\theta) \approx \underbrace{f(\theta_0)}_{\text{Current value}} + \underbrace{\nabla f(\theta_0)^\top (\theta - \theta_0)}_{\text{Change in value as we move away}}$$

The approximation is only good for a local neighborhood. Add an l_2 distance penalty, for some $\eta > 0$:

$$f_{\theta_0,\eta}(\theta) := f(\theta_0) + \nabla f(\theta_0)^\top (\theta - \theta_0) + \frac{1}{2\eta} \left|\left|\theta - \theta_0\right|\right|^2$$

(Also a second-order approximation with $\nabla^2 f(\theta) \approx (1/\eta) I_{d\times d}$).
The unique minimizer of $f_{\theta_0,\eta}$ is

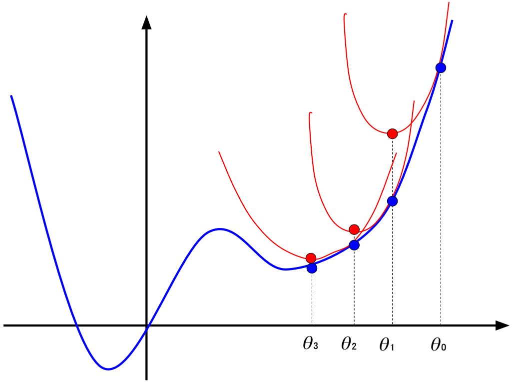
$$\theta = \theta_0 - \eta \nabla f(\theta_0)$$

Gradient Descent

Start from some $\theta_0 \in \mathbb{R}^d$, repeatedly minimize local approx. $f_{\theta_t,\eta_t}(\theta)$ around θ_t by

$$\theta_{t+1} = \theta_t - \underbrace{\eta_t}_{\text{“step size” or “learning rate”}} \nabla f(\theta_t)$$

until $\nabla f(\theta_t) \approx 0_d$



Properties of Gradient Descent

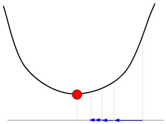
Universal: Can minimize *any* differentiable $f : \mathbb{R}^d \rightarrow \mathbb{R}$

- Only need the ability to calculate gradient $\nabla f : \mathbb{R}^d \rightarrow \mathbb{R}^d$

Local search: Only use *local* information around current location

- Initialization matters, small random values usually okay (e.g., $[\theta_0]_i \sim \mathrm{Unif}(-\alpha, \alpha)$ for $\alpha = 0.01$)
- Convergence at *some* stationary/critical point $\bar{\theta}$ (i.e., $\nabla f(\bar{\theta}) = 0_d$)
  1. Global minimum: $f(\bar{\theta}) = \min_{\theta \in \mathbb{R}^d} \; f(\theta)$
  2. Local minimum: $f(\bar{\theta}) \leq f(\bar{\theta} + u)$ for all small nonzero $u \in \mathbb{R}^d$
  3. Saddle point: Not a local minimum

  (Global minimum is also local minimum.) Depends on initialization & function shape. If f is *convex*, global convergence guaranteed for appropriate step sizes:



Quick Gradient Estimation by Sampling

Often $f : \mathbb{R}^d \to \mathbb{R}$ averages "component" functions $f_i : \mathbb{R}^d \to \mathbb{R}$

$$f(\theta) = \frac{1}{N} \sum_{i=1}^{N} f_i(\theta) \tag{1}$$

Given a partition $\mathcal{I}_1 \ldots \mathcal{I}_M$ of $\{1 \ldots N\}$ (**mini-batches**) where $|\mathcal{I}_j| \ll N$ (**batch size**)

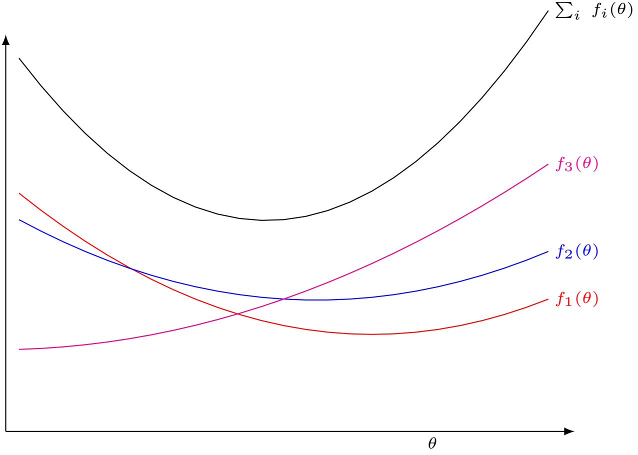
$$\nabla f(\theta) \approx \frac{1}{|\mathcal{I}_j|} \sum_{i \in \mathcal{I}_j} \nabla f_i(\theta) \qquad\qquad j \sim \mathrm{Unif}(\{1 \ldots M\})$$

This allows us to estimate $\nabla f(\theta)$ quickly by averaging $\nabla f_i(\theta)$ in a single mini-batch, without considering all N components.

- Consistent estimation (take expectation over j). This assumes the form (1). Not consistent for general f!

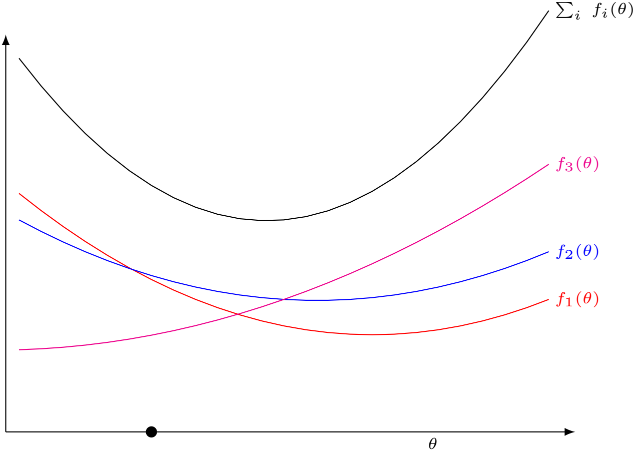
Intuition

(Acknowledgement: This slide is adapted from Greg Shakhnarovich's lecture.)



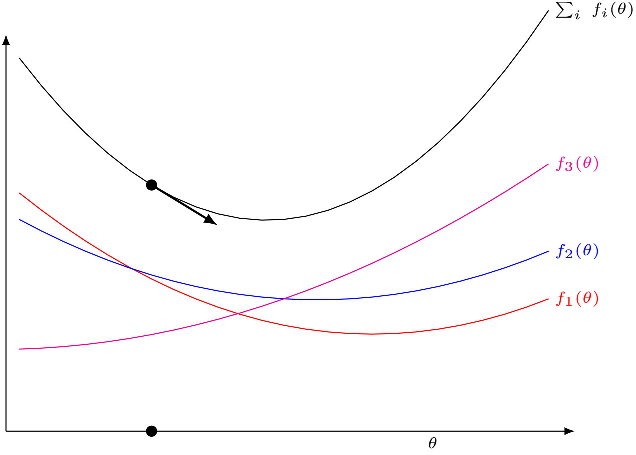
Intuition

(Acknowledgement: This slide is adapted from Greg Shakhnarovich's lecture.)



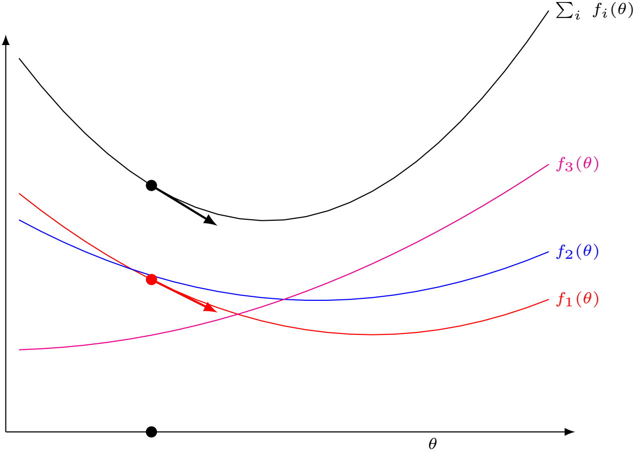
Intuition

(Acknowledgement: This slide is adapted from Greg Shakhnarovich's lecture.)



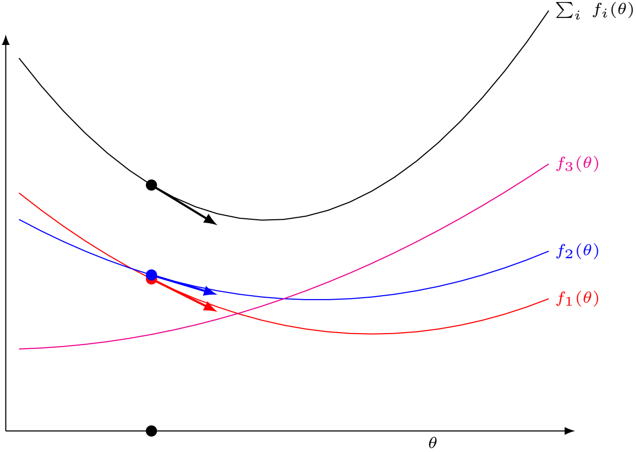
Intuition

(Acknowledgement: This slide is adapted from Greg Shakhnarovich's lecture.)



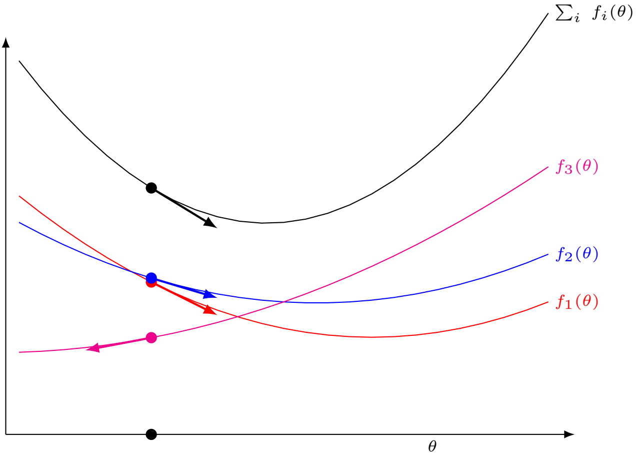
Intuition

(Acknowledgement: This slide is adapted from Greg Shakhnarovich's lecture.)

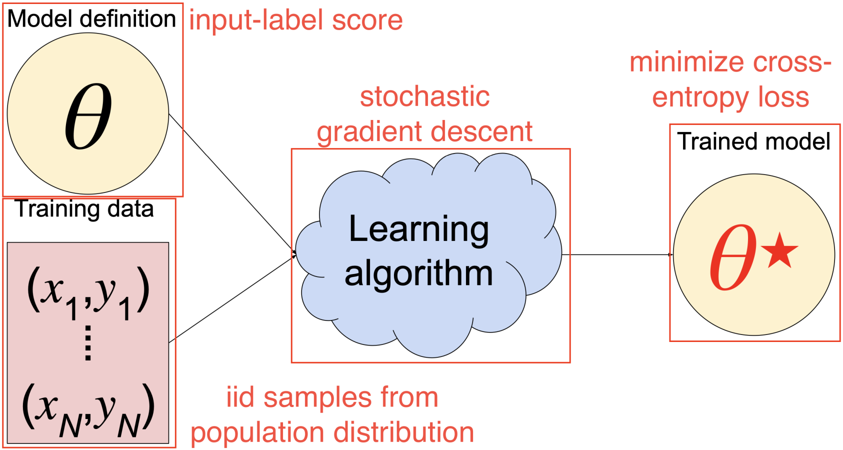


Intuition

(Acknowledgement: This slide is adapted from Greg Shakhnarovich's lecture.)



Summary



No matter what the task is, the principles will (largely) stay the same.