Lecture 2: VC Dimension

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Advanced Machine Learning
September–December 2021

Representational capacity

PAC learning guarantee

Let $\mathcal{H}$ be a hypothesis class, $\delta, \epsilon > 0$ and S a training set of size $n \geq \frac{1}{\epsilon}(\ln |\mathcal{H}| + \ln(1/\delta))$ drawn using D. With probability $\geq 1 - \delta$, every $h \in \mathcal{H}$ with true error $\text{err}_D > \epsilon$ has training error $\text{err}_S > 0$.

hypothesis $\mathcal{H}$

Sample error 0

$\Rightarrow$ True error $< \epsilon$

train error $\neq 0$ $\Rightarrow$ true error $\not< \epsilon$

$= 0$ $\geq \epsilon$

Representational capacity

PAC learning guarantee

Let $\mathcal{H}$ be a hypothesis class, $\delta, \epsilon > 0$ and S a training set of size $n \geq \frac{1}{\epsilon}(\ln|\mathcal{H}| + \ln(1/\delta))$ drawn using D. With probability $\geq 1 - \delta$, every $h \in \mathcal{H}$ with true error $\text{err}_D > \epsilon$ has training error $\text{err}_S > 0$.

Uniform convergence

Let $\mathcal{H}$ be a hypothesis class, $\delta, \epsilon > 0$. If a training set S of size $n \geq \frac{1}{2\epsilon^2}(\ln|\mathcal{H}| + \ln(2/\delta))$ is drawn using D, then with probability $\geq 1 - \delta$, every $h \in \mathcal{H}$ satisfies $|\text{err}_S(h) - \text{err}_D(h)| \leq \epsilon$.

Representational capacity

PAC learning guarantee

Let $\mathcal{H}$ be a hypothesis class, $\delta, \epsilon > 0$ and S a training set of size $n \geq \frac{1}{\epsilon}(\ln|\mathcal{H}| + \ln(1/\delta))$ drawn using D. With probability $\geq 1 - \delta$, every $h \in \mathcal{H}$ with true error $\text{err}_D > \epsilon$ has training error $\text{err}_S > 0$.

Uniform convergence

Let $\mathcal{H}$ be a hypothesis class, $\delta, \epsilon > 0$. If a training set S of size $n \geq \frac{1}{2\epsilon^2}(\ln|\mathcal{H}| + \ln(2/\delta))$ is drawn using D, then with probability $\geq 1 - \delta$, every $h \in \mathcal{H}$ satisfies $|\text{err}_S(h) - \text{err}_D(h)| \leq \epsilon$.

- $|\mathcal{H}|$ is representational capacity, when $\mathcal{H}$ is finite

Representational capacity

PAC learning guarantee

Let $\mathcal{H}$ be a hypothesis class, $\delta, \epsilon > 0$ and S a training set of size $n \geq \frac{1}{\epsilon}(\ln|\mathcal{H}| + \ln(1/\delta))$ drawn using D. With probability $\geq 1 - \delta$, every $h \in \mathcal{H}$ with true error $\text{err}_D > \epsilon$ has training error $\text{err}_S > 0$.

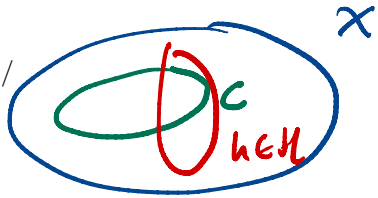
Uniform convergence

Let $\mathcal{H}$ be a hypothesis class, $\delta, \epsilon > 0$. If a training set S of size $n \geq \frac{1}{2\epsilon^2}(\ln|\mathcal{H}| + \ln(2/\delta))$ is drawn using D, then with probability $\geq 1 - \delta$, every $h \in \mathcal{H}$ satisfies $|\text{err}_S(h) - \text{err}_D(h)| \leq \epsilon$.

- $|\mathcal{H}|$ is representational capacity, when $\mathcal{H}$ is finite
- How do we adapt and apply these bounds when $\mathcal{H}$ is infinite?

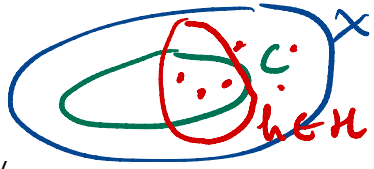
Shattering

- Set system: $(X, \mathcal{H})$
  - X is a set — instance space
  - $\mathcal{H}$, set of subsets of X — set of possible classifiers / hypotheses

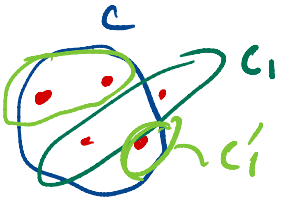


Shattering

- Set system: $(X, \mathcal{H})$
  - X is a set — instance space
  - $\mathcal{H}$, set of subsets of X — set of possible classifiers / hypotheses
- $A \subseteq X$ is shattered by $\mathcal{H}$ if every subset of A is given by $A \cap h$ for some $h \in \mathcal{H}$



$2^{|A|}$ $A =$



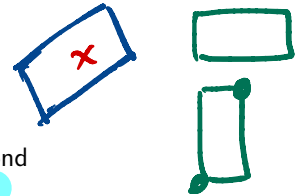
$C_1 = \bar{C}_1'$

Shattering

- Set system: $(X, \mathcal{H})$
  - X is a set — instance space
  - $\mathcal{H}$, set of subsets of X — set of possible classifiers / hypotheses
- $A \subseteq X$ is shattered by $\mathcal{H}$ if every subset of A is given by $A \cap h$ for some $h \in \mathcal{H}$
  - Every way of splitting A is captured by a hypothesis in $\mathcal{H}$
  - $2^{|A|}$ different subsets of A

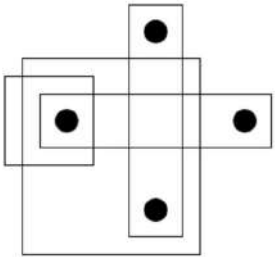
Shattering

- Set system: $(X, \mathcal{H})$
  - X is a set — instance space
  - $\mathcal{H}$, set of subsets of X — set of possible classifiers / hypotheses
- $A \subseteq X$ is shattered by $\mathcal{H}$ if every subset of A is given by $A \cap h$ for some $h \in \mathcal{H}$
  - Every way of splitting A is captured by a hypothesis in $\mathcal{H}$
  - $2^{|A|}$ different subsets of A
- Example:
  - $X = \mathbb{R} \times \mathbb{R}$
  - $\mathcal{H}$: Axis-parallel rectangles
  - A : Four points forming a diamond



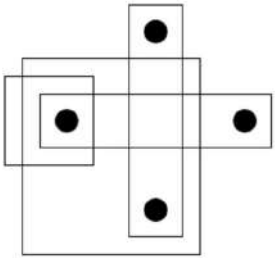
Shattering

- Set system: $(X, \mathcal{H})$
  - X is a set — instance space
  - $\mathcal{H}$, set of subsets of X — set of possible classifiers / hypotheses
- $A \subseteq X$ is shattered by $\mathcal{H}$ if every subset of A is given by $A \cap h$ for some $h \in \mathcal{H}$
  - Every way of splitting A is captured by a hypothesis in $\mathcal{H}$
  - $2^{|A|}$ different subsets of A
- Example:
  - $X = \mathbb{R} \times \mathbb{R}$
  - $\mathcal{H}$: Axis-parallel rectangles
  - A : Four points forming a diamond
  - $\mathcal{H}$ shatters A



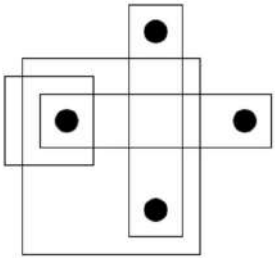
VC-Dimension [Vapnik-Chervonenkis]

- VC-Dimension of $\mathcal{H}$ — size of the largest subset of X shattered by $\mathcal{H}$
  - For axis-parallel rectangles, VC-dimension is at least 4

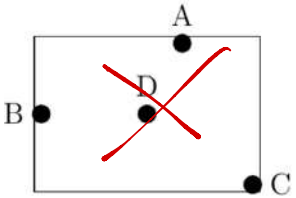


VC-Dimension [Vapnik-Chervonenkis]

- VC-Dimension of $\mathcal{H}$ — size of the largest subset of X shattered by $\mathcal{H}$
  - For axis-parallel rectangles, VC-dimension is at least 4
  - Not a universal requirement — some sets of size 4 may not be shattered

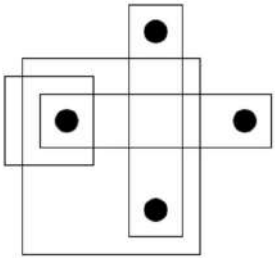


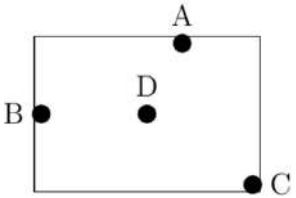
$\{A, B, C\}$ cannot be captured



VC-Dimension [Vapnik-Chervonenkis]

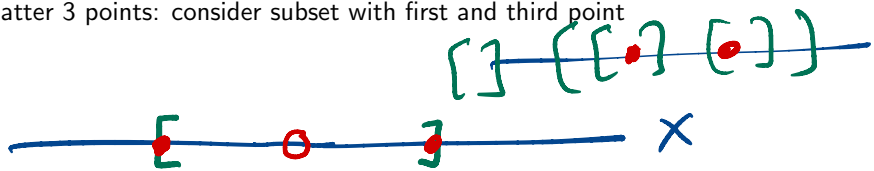
- VC-Dimension of $\mathcal{H}$ — size of the largest subset of X shattered by $\mathcal{H}$
  - For axis-parallel rectangles, VC-dimension is at least 4
  - Not a universal requirement — some sets of size 4 may not be shattered
- No set of size 5 can be shattered by axis-parallel rectangles
  - Draw a bounding box rectangle — each edge touches a boundary point
  - At least one point lies inside the bounding box
  - Any set that includes the boundary points also includes the interior point





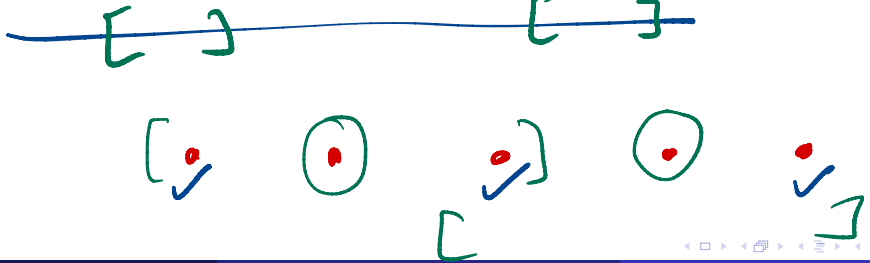
VC-Dimension, Examples

- Intervals of reals have VC-dimension 2
  - $X = \mathbb{R}$, $\mathcal{H} = \{[a, b] \mid a \leq b \in \mathbb{R}\}$
  - Cannot shatter 3 points: consider subset with first and third point



VC-Dimension, Examples

- Intervals of reals have VC-dimension 2
  - $X = \mathbb{R}$, $\mathcal{H} = \{[a, b] \mid a \leq b \in \mathbb{R}\}$
  - Cannot shatter 3 points: consider subset with first and third point
- Pairs of intervals of reals have VC-dimension 4
  - $X = \mathbb{R}$, $\mathcal{H} = \{[a, b] \cup [c, d] \mid a \leq b, c \leq d \in \mathbb{R}\}$
  - Cannot shatter 5 points: consider subset with first, third and fifth point



VC-Dimension, Examples

- Intervals of reals have VC-dimension 2
  - $X = \mathbb{R}$, $\mathcal{H} = \{[a, b] \mid a \leq b \in \mathbb{R}\}$
  - Cannot shatter 3 points: consider subset with first and third point
- Pairs of intervals of reals have VC-dimension 4
  - $X = \mathbb{R}$, $\mathcal{H} = \{[a, b] \cup [c, d] \mid a \leq b, c \leq d \in \mathbb{R}\}$
  - Cannot shatter 5 points: consider subset with first, third and fifth point
- Finite sets of real numbers
  - $X = \mathbb{R}$, $\mathcal{H} = \{Z \mid Z \subseteq \mathbb{R}, |Z| < \infty\}$
  - Can shatter any finite set of reals — VC-dimension is infinite

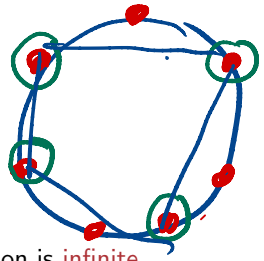
$A \subseteq_{fin} \mathbb{R}$

$X \subseteq A$, $X \subseteq_{fin} \mathbb{R}$

VC-Dimension, Examples

- Intervals of reals have VC-dimension 2
  - $X = \mathbb{R}$, $\mathcal{H} = \{[a, b] \mid a \leq b \in \mathbb{R}\}$
  - Cannot shatter 3 points: consider subset with first and third point
- Pairs of intervals of reals have VC-dimension 4
  - $X = \mathbb{R}$, $\mathcal{H} = \{[a, b] \cup [c, d] \mid a \leq b, c \leq d \in \mathbb{R}\}$
  - Cannot shatter 5 points: consider subset with first, third and fifth point
- Finite sets of real numbers
  - $X = \mathbb{R}$, $\mathcal{H} = \{Z \mid Z \subseteq \mathbb{R}, |Z| < \infty\}$
  - Can shatter any finite set of reals — VC-dimension is infinite
- Convex polygons, $X = \mathbb{R} \times \mathbb{R}$
  - For any n, place n points on unit circle
  - Each subset of these points is a convex polygon — VC-dimension is infinite

$|A| = 7$



VC-dimension and machine learning

PAC learning guarantee

Let $\mathcal{H}$ be a hypothesis class, $\delta, \epsilon > 0$ and S a training set of size $n \geq \frac{1}{\epsilon}(\ln |\mathcal{H}| + \ln(1/\delta))$ drawn using D. With probability $\geq 1 - \delta$, every $h \in \mathcal{H}$ with true error $\text{err}_D > \epsilon$ has training error $\text{err}_S > 0$.

VC-dimension and machine learning

PAC learning guarantee

Let $\mathcal{H}$ be a hypothesis class, $\delta, \epsilon > 0$ and S a training set of size $n \geq \frac{1}{\epsilon}(\ln|\mathcal{H}| + \ln(1/\delta))$ drawn using D. With probability $\geq 1 - \delta$, every $h \in \mathcal{H}$ with true error $\text{err}_D > \epsilon$ has training error $\text{err}_S > 0$.

- We can rewrite this using VC-dimension.

Sample bound using VC-dimension

For any class $\mathcal{H}$ and distribution D, if a training sample S is drawn using D of size $O\left(\frac{1}{\epsilon}\left[\text{VC-dim}(\mathcal{H})\ln\frac{1}{\epsilon} + \ln\frac{1}{\delta}\right]\right)$, then with probability $\geq 1 - \delta$,

- every $h \in \mathcal{H}$ with true error $\text{err}_D(h) \geq \epsilon$ has training error $\text{err}_S(h) > 0$,
- i.e., every $h \in \mathcal{H}$ with training error $\text{err}_S(h) = 0$. has true error $\text{err}_D(h) < \epsilon$

VC-dimension and machine learning

PAC learning guarantee

Let $\mathcal{H}$ be a hypothesis class, $\delta, \epsilon > 0$ and S a training set of size $n \geq \frac{1}{\epsilon}(\ln |\mathcal{H}| + \ln(1/\delta))$ drawn using D. With probability $\geq 1 - \delta$, every $h \in \mathcal{H}$ with true error $\text{err}_D > \epsilon$ has training error $\text{err}_S > 0$.

- We can rewrite this using VC-dimension. Can similarly restate uniform convergence.

Sample bound using VC-dimension

For any class $\mathcal{H}$ and distribution D, if a training sample S is drawn using D of size $O\left(\frac{1}{\epsilon}\left[\text{VC-dim}(\mathcal{H}) \ln \frac{1}{\epsilon} + \ln \frac{1}{\delta}\right]\right)$, then with probability $\geq 1 - \delta$,

- every $h \in \mathcal{H}$ with true error $\text{err}_D(h) \geq \epsilon$ has training error $\text{err}_S(h) > 0$,
- i.e., every $h \in \mathcal{H}$ with training error $\text{err}_S(h) = 0$. has true error $\text{err}_D(h) < \epsilon$

Summary

- PAC learning and uniform convergence use size of finite hypothesis set as measure of representational capacity
- VC-dimension provides a way of measuring capacity for infinite hypothesis sets
- VC-dimension may be finite or infinite
- For finite VC-dimension, we have analogues of PAC learning guarantee and uniform convergence
- Note that these theoretical bounds are hard to use in practice
- Difficult, if not impossible, to compute VC-dimension for complex models

Lecture 3: Loss functions

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Advanced Machine Learning
September–December 2021

Gradient descent

- Supervised learning estimates parameters for a model based on training data

Gradient descent

- Supervised learning estimates parameters for a model based on training data
- Parameter estimate is through gradient descent
  - Define a loss function measuring the error with respect to training data
  - Compute gradients with respect to each parameter
  - Adjust parameters by a small step in direction opposite to gradients

Gradient descent

- Supervised learning estimates parameters for a model based on training data
- Parameter estimate is through gradient descent
  - Define a loss function measuring the error with respect to training data
  - Compute gradients with respect to each parameter
  - Adjust parameters by a small step in direction opposite to gradients
- Typical loss functions include mean squared error (MSE) and cross entropy

Gradient descent

- Supervised learning estimates parameters for a model based on training data
- Parameter estimate is through gradient descent
  - Define a loss function measuring the error with respect to training data
  - Compute gradients with respect to each parameter
  - Adjust parameters by a small step in direction opposite to gradients
- Typical loss functions include mean squared error (MSE) and cross entropy
- How do arrive at these loss functions?

Maximum likelihood estimators (MLE)

- Build a model M from training data $D = \{(x_1, y_1,), (x_2, y_2,), \ldots, (x_n, y_n)\}$

Maximum likelihood estimators (MLE)

- Build a model M from training data $D = \{(x_1, y_1,), (x_2, y_2,), \ldots, (x_n, y_n)\}$
- Learning — define M by computing parameters θ

Maximum likelihood estimators (MLE)

- Build a model M from training data $D = \{(x_1, y_1,), (x_2, y_2,), \ldots, (x_n, y_n)\}$
- Learning — define M by computing parameters θ
- Model predicts value $\hat{y}$ on input x_i with probability $P_{\text{model}}(\hat{y} \mid x_i, \theta)$

Maximum likelihood estimators (MLE)

- Build a model M from training data $D = \{(x_1, y_1,), (x_2, y_2,), \ldots, (x_n, y_n)\}$
- Learning — define M by computing parameters θ
- Model predicts value $\hat{y}$ on input x_i with probability $P_{\text{model}}(\hat{y} \mid x_i, \theta)$
- Probability of predicting correct value is $P_{\text{model}}(y_i \mid x_i, \theta)$

Maximum likelihood estimators (MLE)

- Build a model M from training data $D = \{(x_1, y_1,), (x_2, y_2,), \ldots, (x_n, y_n)\}$
- Learning — define M by computing parameters θ
- Model predicts value $\hat{y}$ on input x_i with probability $P_{\text{model}}(\hat{y} \mid x_i, \theta)$
- Probability of predicting correct value is $P_{\text{model}}(y_i \mid x_i, \theta)$
- Likelihood is $\prod_{i=1}^{n} P_{\text{model}}(y_i \mid x_i, \theta)$

Maximum likelihood estimators (MLE)

- Build a model M from training data $D = \{(x_1, y_1,), (x_2, y_2,), \ldots, (x_n, y_n)\}$
- Learning — define M by computing parameters θ
- Model predicts value $\hat{y}$ on input x_i with probability $P_{\text{model}}(\hat{y} \mid x_i, \theta)$
- Probability of predicting correct value is $P_{\text{model}}(y_i \mid x_i, \theta)$
- Likelihood is $\prod_{i=1}^{n} P_{\text{model}}(y_i \mid x_i, \theta)$
- Find M that maximizes the likelihood

Log likelihood

- Maximize the likelihood $\prod_{i=1}^{n} P_{\text{model}}(y_i \mid x_i, \theta)$

Log likelihood

- Maximize the likelihood $\prod_{i=1}^{n} P_{\text{model}}(y_i \mid x_i, \theta)$

- log is an increasing function, so we can equivalently maximize log likelihood

$$\log\left(\prod_{i=1}^{n} P_{\text{model}}(y_i \mid x_i, \theta)\right)$$

Log likelihood

- Maximize the likelihood $\prod_{i=1}^{n} P_{\text{model}}(y_i \mid x_i, \theta)$
- log is an increasing function, so we can equivalently maximize log likelihood

$$\log\left(\prod_{i=1}^{n} P_{\text{model}}(y_i \mid x_i, \theta)\right)$$

- Rewrite log likelihood as a sum

$$\log\left(\prod_{i=1}^{n} P_{\text{model}}(y_i \mid x_i, \theta)\right) = \sum_{i=1}^{n} \log(P_{\text{model}}(y_i \mid x_i, \theta))$$

Maximizing Log likelihood

- Define $P_{\text{data}}(y \mid x_i)$ as follows: $P_{\text{data}}(y \mid x_i) = \begin{cases} 1 & \text{if } y = y_i \\ 0 & \text{otherwise} \end{cases}$

Maximizing Log likelihood

- Define $P_{\text{data}}(y \mid x_i)$ as follows: $P_{\text{data}}(y \mid x_i) = \begin{cases} 1 & \text{if } y = y_i \\ 0 & \text{otherwise} \end{cases}$
- For each x_i, $P_{\text{data}}(y_i \mid x_i) = 1$, so rewrite log likelihood as

$$\sum_{i=1}^{n} \log(P_{\text{model}}(y_i \mid x_i, \theta)) = \sum_{i=1}^{n} P_{\text{data}}(y_i \mid x_i) \cdot \log(P_{\text{model}}(y_i \mid x_i, \theta))$$

Maximizing Log likelihood

- Define $P_{\text{data}}(y \mid x_i)$ as follows: $P_{\text{data}}(y \mid x_i) = \begin{cases} 1 & \text{if } y = y_i \\ 0 & \text{otherwise} \end{cases}$
- For each x_i, $P_{\text{data}}(y_i \mid x_i) = 1$, so rewrite log likelihood as

$$\sum_{i=1}^{n} \log(P_{\text{model}}(y_i \mid x_i, \theta)) = \sum_{i=1}^{n} P_{\text{data}}(y_i \mid x_i) \cdot \log(P_{\text{model}}(y_i \mid x_i, \theta))$$

- Log likelihood is a function of the learned parameters θ

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} P_{\text{data}}(y_i \mid x_i) \log(P_{\text{model}}(y_i \mid x_i, \theta))$$

Maximizing Log likelihood

- Define $P_{\text{data}}(y \mid x_i)$ as follows: $P_{\text{data}}(y \mid x_i) = \begin{cases} 1 & \text{if } y = y_i \\ 0 & \text{otherwise} \end{cases}$
- For each x_i, $P_{\text{data}}(y_i \mid x_i) = 1$, so rewrite log likelihood as

$$\sum_{i=1}^{n} \log(P_{\text{model}}(y_i \mid x_i, \theta)) = \sum_{i=1}^{n} P_{\text{data}}(y_i \mid x_i) \cdot \log(P_{\text{model}}(y_i \mid x_i, \theta))$$

- Log likelihood is a function of the learned parameters θ

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} P_{\text{data}}(y_i \mid x_i) \log(P_{\text{model}}(y_i \mid x_i, \theta))$$

- To maximize, find an optimum value of θ: $\dfrac{\partial \mathcal{L}(\theta)}{\partial \theta} = 0$

Cross entropy

- Let $X = \{x_1, x_2, \ldots, x_k\}$ with a probability distribution P

Cross entropy

- Let $X = \{x_1, x_2, \ldots, x_k\}$ with a probability distribution P
- Entropy is defined as $H(P) = -\sum_{i=1}^{k} P(x_i) \log P(x_i)$
  - Average number of bits to encode each element of X

Cross entropy

- Let $X = \{x_1, x_2, \ldots, x_k\}$ with a probability distribution P
- Entropy is defined as $H(P) = -\sum_{i=1}^{k} P(x_i) \log P(x_i)$
  - Average number of bits to encode each element of X
- Given two distributions P and Q over X, cross entropy is defined as

$$H(P, Q) = -\sum_{i=1}^{k} P(x_i) \log Q(x_i)$$

Cross entropy

- Let $X = \{x_1, x_2, \ldots, x_k\}$ with a probability distribution P
- Entropy is defined as $H(P) = -\sum_{i=1}^{k} P(x_i) \log P(x_i)$
  - Average number of bits to encode each element of X
- Given two distributions P and Q over X, cross entropy is defined as

  $H(P, Q) = -\sum_{i=1}^{k} P(x_i) \log Q(x_i)$
  - Imagine an encoding based on Q where true distribution is P
  - Again, average number of bits to encode each element of X

Cross entropy

- Let $X = \{x_1, x_2, \ldots, x_k\}$ with a probability distribution P
- Entropy is defined as $H(P) = -\sum_{i=1}^{k} P(x_i) \log P(x_i)$
  - Average number of bits to encode each element of X
- Given two distributions P and Q over X, cross entropy is defined as

  $H(P, Q) = -\sum_{i=1}^{k} P(x_i) \log Q(x_i)$
  - Imagine an encoding based on Q where true distribution is P
  - Again, average number of bits to encode each element of X
- Note that cross entropy is not symmetric: $H(P, Q) \neq H(Q, P)$

Cross entropy and MLE

- Maximum likelihood estimator (MLE) — maximize

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} P_{\text{data}}(y_i \mid x_i) \log(P_{\text{model}}(y_i \mid x_i, \theta))$$

P( ) log (a( ))

Cross entropy and MLE

- Maximum likelihood estimator (MLE) — maximize

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} P_{\text{data}}(y_i \mid x_i) \log(P_{\text{model}}(y_i \mid x_i, \theta))$$

- P_{model} is an estimate for the true distribution P_{data}

Cross entropy and MLE

- Maximum likelihood estimator (MLE) — maximize

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} P_{\text{data}}(y_i \mid x_i) \log(P_{\text{model}}(y_i \mid x_i, \theta))$$

- P_{model} is an estimate for the true distribution P_{data}
- $H(P_{\text{data}}, P_{\text{model}}) = -\sum_{i=1}^{k} P_{\text{data}}(y \mid x_i) \log(P_{\text{model}}(y \mid x_i, \theta))$

Cross entropy and MLE

- Maximum likelihood estimator (MLE) — maximize

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} P_{\text{data}}(y_i \mid x_i) \log(P_{\text{model}}(y_i \mid x_i, \theta))$$

- P_{model} is an estimate for the true distribution P_{data}
- $H(P_{\text{data}}, P_{\text{model}}) = -\sum_{i=1}^{k} P_{\text{data}}(y \mid x_i) \log(P_{\text{model}}(y \mid x_i, \theta))$
- $H(P_{\text{data}}, P_{\text{model}}) = -\mathcal{L}(\theta)$

Cross entropy and MLE

- Maximum likelihood estimator (MLE) — maximize

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} P_{\text{data}}(y_i \mid x_i) \log(P_{\text{model}}(y_i \mid x_i, \theta))$$

- P_{model} is an estimate for the true distribution P_{data}
- $H(P_{\text{data}}, P_{\text{model}}) = -\sum_{i=1}^{k} P_{\text{data}}(y \mid x_i) \log(P_{\text{model}}(y \mid x_i, \theta))$
- $H(P_{\text{data}}, P_{\text{model}}) = -\mathcal{L}(\theta)$
- Minimizing cross entropy is the same as maximizing likelihood

Cross entropy and MLE

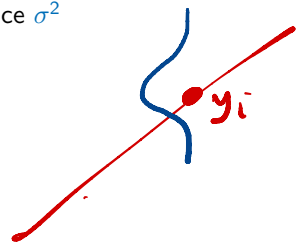
- Maximum likelihood estimator (MLE) — maximize

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} P_{\text{data}}(y_i \mid x_i) \log(P_{\text{model}}(y_i \mid x_i, \theta))$$

- P_{model} is an estimate for the true distribution P_{data}
- $H(P_{\text{data}}, P_{\text{model}}) = -\sum_{i=1}^{k} P_{\text{data}}(y \mid x_i) \log(P_{\text{model}}(y \mid x_i, \theta))$
- $H(P_{\text{data}}, P_{\text{model}}) = -\mathcal{L}(\theta)$
- Minimizing cross entropy is the same as maximizing likelihood
- The "cross entropy loss function" is a special form of this generic observation

Regression and MSE loss

- Training input is $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
  - Noisy outputs from a linear function
  - $y_i = w^T x_i + \epsilon$
  - $\epsilon \sim \mathcal{N}(0, \sigma^2)$: Gaussian noise, mean 0, fixed variance σ^2
  - $y_i \sim \mathcal{N}(\mu_i, \sigma^2)$, $\mu_i = w^T x_i$



Regression and MSE loss

- Training input is $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
  - Noisy outputs from a linear function
  - $y_i = w^T x_i + \epsilon$
  - $\epsilon \sim \mathcal{N}(0, \sigma^2)$: Gaussian noise, mean 0, fixed variance σ^2
  - $y_i \sim \mathcal{N}(\mu_i, \sigma^2)$, $\mu_i = w^T x_i$
- Model gives us an estimate for w, so regression learns μ_i for each x_i

Regression and MSE loss

- Training input is $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
  - Noisy outputs from a linear function
  - $y_i = w^T x_i + \epsilon$
  - $\epsilon \sim \mathcal{N}(0, \sigma^2)$: Gaussian noise, mean 0, fixed variance σ^2
  - $y_i \sim \mathcal{N}(\mu_i, \sigma^2)$, $\mu_i = w^T x_i$
- Model gives us an estimate for w, so regression learns μ_i for each x_i
- $P_{\text{model}}(y_i \mid x_i, \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu_i)^2}{2\sigma^2}}$

Regression and MSE loss

- Training input is $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
  - Noisy outputs from a linear function
  - $y_i = w^T x_i + \epsilon$
  - $\epsilon \sim \mathcal{N}(0, \sigma^2)$: Gaussian noise, mean 0, fixed variance σ^2
  - $y_i \sim \mathcal{N}(\mu_i, \sigma^2)$, $\mu_i = w^T x_i$
- Model gives us an estimate for w, so regression learns μ_i for each x_i
- $P_{\text{model}}(y_i \mid x_i, \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu_i)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-w^T x_i)^2}{2\sigma^2}}$

Regression and MSE loss

- Training input is $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
  - Noisy outputs from a linear function
  - $y_i = w^T x_i + \epsilon$
  - $\epsilon \sim \mathcal{N}(0, \sigma^2)$: Gaussian noise, mean 0, fixed variance σ^2
  - $y_i \sim \mathcal{N}(\mu_i, \sigma^2)$, $\mu_i = w^T x_i$
- Model gives us an estimate for w, so regression learns μ_i for each x_i
- $P_{\text{model}}(y_i \mid x_i, \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu_i)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-w^T x_i)^2}{2\sigma^2}}$
- Log likelihood

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-w^T x_i)^2}{2\sigma^2}} \right)$$

Regression and MSE loss

- Training input is $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
  - Noisy outputs from a linear function
  - $y_i = w^T x_i + \epsilon$
  - $\epsilon \sim \mathcal{N}(0, \sigma^2)$: Gaussian noise, mean 0, fixed variance σ^2
  - $y_i \sim \mathcal{N}(\mu_i, \sigma^2)$, $\mu_i = w^T x_i$
- Model gives us an estimate for w, so regression learns μ_i for each x_i
- $P_{\text{model}}(y_i \mid x_i, \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu_i)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-w^T x_i)^2}{2\sigma^2}}$
- Log likelihood (assuming natural logarithm)

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-w^T x_i)^2}{2\sigma^2}}\right) = n \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \sum_{i=1}^{n} \frac{(y - w^T x_i)^2}{2\sigma^2}$$

Regression and MSE loss

- Log likelihood: $\mathcal{L}(\theta) = n \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) - \sum_{i=1}^{n} \frac{(y - w^T x_i)^2}{2\sigma^2}$

Regression and MSE loss

- Log likelihood: $\mathcal{L}(\theta) = n \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) - \sum_{i=1}^{n} \frac{(y - w^T x_i)^2}{2\sigma^2}$
- $w^T x_i$ is predicted value $\hat{y}_i$

Regression and MSE loss

- Log likelihood: $\mathcal{L}(\theta) = n \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) - \sum_{i=1}^{n} \frac{(y - w^T x_i)^2}{2\sigma^2}$
- $w^T x_i$ is predicted value $\hat{y}_i$
- To maximize $\mathcal{L}(\theta)$ with respect to w, ignore all terms that do not depend on w

Regression and MSE loss

- Log likelihood: $\mathcal{L}(\theta) = n \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) - \sum_{i=1}^{n} \frac{(y - w^T x_i)^2}{2\sigma^2}$
- $w^T x_i$ is predicted value $\hat{y}_i$
- To maximize $\mathcal{L}(\theta)$ with respect to w, ignore all terms that do not depend on w
- Optimum value of w is given by

$$\hat{w}_{\text{MSE}} = \arg\max_{w} \left[-\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \right]$$

Regression and MSE loss

- Log likelihood: $\mathcal{L}(\theta) = n \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) - \sum_{i=1}^{n} \frac{(y - w^T x_i)^2}{2\sigma^2}$
- $w^T x_i$ is predicted value $\hat{y}_i$
- To maximize $\mathcal{L}(\theta)$ with respect to w, ignore all terms that do not depend on w
- Optimum value of w is given by

$$\hat{w}_{\text{MSE}} = \arg\max_{w} \left[-\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \right] = \arg\min_{w} \left[\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \right]$$

Regression and MSE loss

- Log likelihood: $\mathcal{L}(\theta) = n \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) - \sum_{i=1}^{n} \frac{(y - w^T x_i)^2}{2\sigma^2}$
- $w^T x_i$ is predicted value $\hat{y}_i$
- To maximize $\mathcal{L}(\theta)$ with respect to w, ignore all terms that do not depend on w
- Optimum value of w is given by

$$\hat{w}_{\text{MSE}} = \arg\max_w \left[-\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \right] = \arg\min_w \left[\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \right]$$

- Assuming data points are generated by linear function and then perturbed by Gaussian noise, MSE is the "correct" loss function to maximize likelihood (and minimize cross entropy)

Binary classification

- Compute linear output $z_i = w^T x_i$, then apply sigmoid $\sigma(z) = \dfrac{1}{1+e^{-z}}$

Binary classification

- Compute linear output $z_i = w^T x_i$, then apply sigmoid $\sigma(z) = \dfrac{1}{1+e^{-z}}$
- Let $a_i = \sigma(z_i)$.

Binary classification

- Compute linear output $z_i = w^T x_i$, then apply sigmoid $\sigma(z) = \dfrac{1}{1+e^{-z}}$
- Let $a_i = \sigma(z_i)$. So, $P_{\text{model}}(y_i = 1) = a_i$, $P_{\text{model}}(y_i = 0) = 1 - a_i$

Binary classification

- Compute linear output $z_i = w^T x_i$, then apply sigmoid $\sigma(z) = \dfrac{1}{1+e^{-z}}$
- Let $a_i = \sigma(z_i)$. So, $P_{\text{model}}(y_i = 1) = a_i$, $P_{\text{model}}(y_i = 0) = 1 - a_i$
- Cross entropy: $\sum_{i=1}^{n} \sum_{j \in \{0,1\}} P_{\text{data}}(y_i = j) \log(P_{\text{model}}(y_i = j \mid x_i, \theta))$

Binary classification

- Compute linear output $z_i = w^T x_i$, then apply sigmoid $\sigma(z) = \dfrac{1}{1+e^{-z}}$
- Let $a_i = \sigma(z_i)$. So, $P_{\text{model}}(y_i = 1) = a_i$, $P_{\text{model}}(y_i = 0) = 1 - a_i$
- Cross entropy: $\displaystyle\sum_{i=1}^{n} \sum_{j \in \{0,1\}} P_{\text{data}}(y_i = j) \log(P_{\text{model}}(y_i = j \mid x_i, \theta))$
- Expand:
  $\displaystyle\sum_{i=1}^{n} P_{\text{data}}(y_i = 0) \log P_{\text{model}}(y_i = 0 \mid x_i, \theta) + P_{\text{data}}(y_i = 1) \log P_{\text{model}}(y_i = 1 \mid x_i, \theta)$

Binary classification

- Compute linear output $z_i = w^T x_i$, then apply sigmoid $\sigma(z) = \dfrac{1}{1+e^{-z}}$
- Let $a_i = \sigma(z_i)$. So, $P_{\text{model}}(y_i = 1) = a_i$, $P_{\text{model}}(y_i = 0) = 1 - a_i$
- Cross entropy: $\displaystyle\sum_{i=1}^{n} \sum_{j \in \{0,1\}} P_{\text{data}}(y_i = j) \log(P_{\text{model}}(y_i = j \mid x_i, \theta))$
- Expand:
  $\displaystyle\sum_{i=1}^{n} P_{\text{data}}(y_i = 0) \log P_{\text{model}}(y_i = 0 \mid x_i, \theta) + P_{\text{data}}(y_i = 1) \log P_{\text{model}}(y_i = 1 \mid x_i, \theta)$
- Equivalently, $\displaystyle\sum_{i=1}^{n} (1 - y_i) \cdot \log(1 - a_i) + y_i \cdot \log a_i$

Binary classification

- Compute linear output $z_i = w^T x_i$, then apply sigmoid $\sigma(z) = \dfrac{1}{1+e^{-z}}$
- Let $a_i = \sigma(z_i)$. So, $P_{\text{model}}(y_i = 1) = a_i$, $P_{\text{model}}(y_i = 0) = 1 - a_i$
- Cross entropy: $\displaystyle\sum_{i=1}^{n} \sum_{j \in \{0,1\}} P_{\text{data}}(y_i = j) \log(P_{\text{model}}(y_i = j \mid x_i, \theta))$
- Expand:
  $\displaystyle\sum_{i=1}^{n} P_{\text{data}}(y_i = 0) \log P_{\text{model}}(y_i = 0 \mid x_i, \theta) + P_{\text{data}}(y_i = 1) \log P_{\text{model}}(y_i = 1 \mid x_i, \theta)$
- Equivalently, $\displaystyle\sum_{i=1}^{n} (1 - y_i) \cdot \log(1 - a_i) + y_i \cdot \log a_i$
- Recommended loss function, directly minimizes cross entropy

Summary

- Our goal is to find a maximum likelihood estimator

Summary

- Our goal is to find a maximum likelihood estimator
- Gradient descent uses a loss function to optimize parameters

Summary

- Our goal is to find a maximum likelihood estimator
- Gradient descent uses a loss function to optimize parameters
- Finding MLE is equivalent to minimizing cross entropy $H(P_{\text{data}}, P_{\text{model}})$

Summary

- Our goal is to find a maximum likelihood estimator
- Gradient descent uses a loss function to optimize parameters
- Finding MLE is equivalent to minimizing cross entropy $H(P_{\text{data}}, P_{\text{model}})$
- Applying this to a given situation, we arrive at concrete loss functions
  - Mean square error for regression
  - "Cross entropy" for binary classification

Summary

- Our goal is to find a maximum likelihood estimator
- Gradient descent uses a loss function to optimize parameters
- Finding MLE is equivalent to minimizing cross entropy $H(P_{\text{data}}, P_{\text{model}})$
- Applying this to a given situation, we arrive at concrete loss functions
  - Mean square error for regression
  - "Cross entropy" for binary classification