Probabilistic models - Bayesian Methods

Mengye Ren

NYU

Oct 31, 2023

Parametric Family of Densities

- A **parametric family of densities** is a set

$$\{p(y \mid \theta) : \theta \in \Theta\},$$

  - where $p(y \mid \theta)$ is a density on a **sample space** $\mathcal{Y}$, and
  - θ is a **parameter** in a [finite dimensional] **parameter space** Θ.

Parametric Family of Densities

- A **parametric family of densities** is a set

$$\{p(y \mid \theta) : \theta \in \Theta\},$$

  - where $p(y \mid \theta)$ is a density on a **sample space** $\mathcal{Y}$, and
  - θ is a **parameter** in a [finite dimensional] **parameter space** Θ**.**
- This is the common starting point for a treatment of classical or Bayesian statistics.
- In this lecture, whenever we say "density", we could replace it with "mass function." (and replace integrals with sums).

Frequentist or "Classical" Statistics

- We're still working with a parametric family of densities:

$$\{p(y \mid \theta) \mid \theta \in \Theta\}.$$

Frequentist or "Classical" Statistics

- We're still working with a parametric family of densities:

$$\{p(y \mid \theta) \mid \theta \in \Theta\}.$$

- Assume that $p(y \mid \theta)$ governs the world we are observing, for some $\theta \in \Theta$.

Frequentist or "Classical" Statistics

- We're still working with a parametric family of densities:

$$\{p(y \mid \theta) \mid \theta \in \Theta\}.$$

- Assume that $p(y \mid \theta)$ governs the world we are observing, for some $\theta \in \Theta$.
- If we knew the right $\theta \in \Theta$, there would be no need for statistics.

Frequentist or "Classical" Statistics

- We're still working with a parametric family of densities:

$$\{p(y \mid \theta) \mid \theta \in \Theta\}.$$

- Assume that $p(y \mid \theta)$ governs the world we are observing, for some $\theta \in \Theta$.
- If we knew the right $\theta \in \Theta$, there would be no need for statistics.
- But instead of θ, we have data $\mathcal{D}$: $y_1, \ldots, y_n$ sampled i.i.d. from $p(y \mid \theta)$.

Frequentist or "Classical" Statistics

- We're still working with a parametric family of densities:

$$\{p(y \mid \theta) \mid \theta \in \Theta\}.$$

- Assume that $p(y \mid \theta)$ governs the world we are observing, for some $\theta \in \Theta$.
- If we knew the right $\theta \in \Theta$, there would be no need for statistics.
- But instead of θ, we have data $\mathcal{D}$: $y_1, \ldots, y_n$ sampled i.i.d. from $p(y \mid \theta)$.
- Statistics is about how to get by with $\mathcal{D}$ in place of θ.

Point Estimation

- One type of statistical problem is **point estimation**.

Point Estimation

- One type of statistical problem is **point estimation**.
- A **statistic** $s = s(\mathcal{D})$ is any function of the data.

Point Estimation

- One type of statistical problem is **point estimation**.
- A **statistic** $s = s(\mathcal{D})$ is any function of the data.
- A statistic $\hat{\theta} = \hat{\theta}(\mathcal{D})$ taking values in Θ is a **point estimator of** θ.

Point Estimation

- One type of statistical problem is **point estimation**.
- A **statistic** $s = s(\mathcal{D})$ is any function of the data.
- A statistic $\hat{\theta} = \hat{\theta}(\mathcal{D})$ taking values in Θ is a **point estimator of** θ.
- A good point estimator will have $\hat{\theta} \approx \theta$.
- **Desirable statistical properties of point estimators**:

Point Estimation

- One type of statistical problem is **point estimation**.
- A **statistic** $s = s(\mathcal{D})$ is any function of the data.
- A statistic $\hat{\theta} = \hat{\theta}(\mathcal{D})$ taking values in Θ is a **point estimator of** θ.
- A good point estimator will have $\hat{\theta} \approx \theta$.
- **Desirable statistical properties of point estimators**:
  - **Consistency:** As data size $n \to \infty$, we get $\hat{\theta}_n \to \theta$.

Point Estimation

- One type of statistical problem is **point estimation**.
- A **statistic** $s = s(\mathcal{D})$ is any function of the data.
- A statistic $\hat{\theta} = \hat{\theta}(\mathcal{D})$ taking values in Θ is a **point estimator of** θ.
- A good point estimator will have $\hat{\theta} \approx \theta$.
- **Desirable statistical properties of point estimators**:
  - **Consistency:** As data size $n \to \infty$, we get $\hat{\theta}_n \to \theta$.
  - **Efficiency:** (Roughly speaking) $\hat{\theta}_n$ is as accurate as we can get from a sample of size n.

Point Estimation

- One type of statistical problem is **point estimation**.
- A **statistic** $s = s(\mathcal{D})$ is any function of the data.
- A statistic $\hat{\theta} = \hat{\theta}(\mathcal{D})$ taking values in Θ is a **point estimator of** θ.
- A good point estimator will have $\hat{\theta} \approx \theta$.
- **Desirable statistical properties of point estimators**:
  - **Consistency:** As data size $n \to \infty$, we get $\hat{\theta}_n \to \theta$.
  - **Efficiency:** (Roughly speaking) $\hat{\theta}_n$ is as accurate as we can get from a sample of size n.
- **Maximum likelihood estimators** are consistent and efficient under reasonable conditions.

Example of Point Estimation: Coin Flipping

- Parametric family of mass functions:

$$p(\text{Heads} \mid \theta) = \theta,$$

  for $\theta \in \Theta = (0, 1)$.

Coin Flipping: MLE

- Data $\mathcal{D} = (H, H, T, T, T, T, T, H, \ldots, T)$, assumed i.i.d. flips.
  - n_h: number of heads
  - n_t: number of tails

Coin Flipping: MLE

- Data $\mathcal{D} = (H, H, T, T, T, T, T, H, \ldots, T)$, assumed i.i.d. flips.
  - n_h: number of heads
  - n_t: number of tails
- **Likelihood function** for data $\mathcal{D}$:

$$L_{\mathcal{D}}(\theta) =$$

Coin Flipping: MLE

- Data $\mathcal{D} = (H, H, T, T, T, T, T, H, \ldots, T)$, assumed i.i.d. flips.
  - n_h: number of heads
  - n_t: number of tails
- **Likelihood function** for data $\mathcal{D}$:

$$L_{\mathcal{D}}(\theta) = p(\mathcal{D} \mid \theta) = \theta^{n_h}(1-\theta)^{n_t}$$

- As usual, it is easier to maximize the log-likelihood function:

$$\begin{aligned} \hat{\theta}_{\text{MLE}} &= \underset{\theta \in \Theta}{\arg\max} \log L_{\mathcal{D}}(\theta) \\ &= \underset{\theta \in \Theta}{\arg\max} \left[n_h \log \theta + n_t \log(1-\theta) \right] \end{aligned}$$

Coin Flipping: MLE

- Data $\mathcal{D} = (H, H, T, T, T, T, T, H, \ldots, T)$, assumed i.i.d. flips.
  - n_h: number of heads
  - n_t: number of tails
- **Likelihood function** for data $\mathcal{D}$:

$$L_{\mathcal{D}}(\theta) = p(\mathcal{D} \mid \theta) = \theta^{n_h}(1-\theta)^{n_t}$$

- As usual, it is easier to maximize the log-likelihood function:

$$\begin{aligned}\hat{\theta}_{\text{MLE}} &= \underset{\theta \in \Theta}{\arg\max} \log L_{\mathcal{D}}(\theta) \\ &= \underset{\theta \in \Theta}{\arg\max} \left[n_h \log\theta + n_t \log(1-\theta)\right]\end{aligned}$$

- First order condition (equating the derivative to zero):

$$\frac{n_h}{\theta} - \frac{n_t}{1-\theta} = 0 \iff \theta = \frac{n_h}{n_h + n_t}$$

Coin Flipping: MLE

- Data $\mathcal{D} = (H, H, T, T, T, T, T, H, \ldots, T)$, assumed i.i.d. flips.
  - n_h: number of heads
  - n_t: number of tails
- **Likelihood function** for data $\mathcal{D}$:

$$L_{\mathcal{D}}(\theta) = p(\mathcal{D} \mid \theta) = \theta^{n_h}(1-\theta)^{n_t}$$

- As usual, it is easier to maximize the log-likelihood function:

$$\begin{aligned}\hat{\theta}_{\text{MLE}} &= \underset{\theta \in \Theta}{\arg\max} \log L_{\mathcal{D}}(\theta) \\ &= \underset{\theta \in \Theta}{\arg\max} \left[n_h \log\theta + n_t \log(1-\theta)\right]\end{aligned}$$

- First order condition (equating the derivative to zero):

$$\frac{n_h}{\theta} - \frac{n_t}{1-\theta} = 0 \iff \theta = \frac{n_h}{n_h + n_t}$$

$\hat{\theta}_{\text{MLE}}$ is the empirical fraction of heads.

- Baysian statistics introduces a crucial new ingredient: the **prior distribution.**

Bayesian Statistics

- Baysian statistics introduces a crucial new ingredient: the **prior distribution.**
- A **prior distribution** $p(\theta)$ is a distribution on the parameter space Θ.

Bayesian Statistics

- Baysian statistics introduces a crucial new ingredient: the **prior distribution.**
- A **prior distribution** $p(\theta)$ is a distribution on the parameter space Θ.
- The prior reflects our belief about θ, **before seeing any data**.

A Bayesian Model

- A [parametric] Bayesian model consists of two pieces:
  1. A parametric family of densities

$$\{p(\mathcal{D} \mid \theta) \mid \theta \in \Theta\}.$$

A Bayesian Model

- A [parametric] Bayesian model consists of two pieces:
  1. A parametric family of densities
  $$\{p(\mathcal{D} \mid \theta) \mid \theta \in \Theta\}.$$
  2. A **prior distribution** $p(\theta)$ on parameter space Θ.

A Bayesian Model

- A [parametric] Bayesian model consists of two pieces:
  1. A parametric family of densities
  $$\{p(\mathcal{D} \mid \theta) \mid \theta \in \Theta\}.$$
  2. A **prior distribution** $p(\theta)$ on parameter space Θ.
- Putting the pieces together, we get a joint density on θ and $\mathcal{D}$:
$$p(\mathcal{D}, \theta) = p(\mathcal{D} \mid \theta)p(\theta).$$

The Posterior Distribution

- The **posterior distribution** for θ is $p(\theta \mid \mathcal{D})$.

The Posterior Distribution

- The **posterior distribution** for θ is $p(\theta \mid \mathcal{D})$.
- Whereas the prior represents belief about θ before observing data $\mathcal{D}$,

The Posterior Distribution

- The **posterior distribution** for θ is $p(\theta \mid \mathcal{D})$.
- Whereas the prior represents belief about θ before observing data $\mathcal{D}$,
- The posterior represents the **rationally updated belief** about θ, after seeing $\mathcal{D}$.

Expressing the Posterior Distribution

- By Bayes rule, can write the posterior distribution as

$$p(\theta \mid \mathcal{D})$$

Expressing the Posterior Distribution

- By Bayes rule, can write the posterior distribution as

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{p(\mathcal{D})}.$$

Expressing the Posterior Distribution

- By Bayes rule, can write the posterior distribution as

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{p(\mathcal{D})}.$$

- Let's consider both sides as functions of θ, for fixed $\mathcal{D}$.

Expressing the Posterior Distribution

- By Bayes rule, can write the posterior distribution as

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{p(\mathcal{D})}.$$

- Let's consider both sides as functions of θ, for fixed $\mathcal{D}$.
- Then both sides are densities on Θ and we can write

$$\underbrace{p(\theta \mid \mathcal{D})}_{\text{posterior}} \propto \underbrace{p(\mathcal{D} \mid \theta)}_{\text{likelihood}} \underbrace{p(\theta)}_{\text{prior}}.$$

Expressing the Posterior Distribution

- By Bayes rule, can write the posterior distribution as

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{p(\mathcal{D})}.$$

- Let's consider both sides as functions of θ, for fixed $\mathcal{D}$.
- Then both sides are densities on Θ and we can write

$$\underbrace{p(\theta \mid \mathcal{D})}_{\text{posterior}} \propto \underbrace{p(\mathcal{D} \mid \theta)}_{\text{likelihood}} \underbrace{p(\theta)}_{\text{prior}}.$$

- Where $\propto$ means we've dropped factors that are independent of θ.

Coin Flipping: Bayesian Model

- Recall that we have a parametric family of mass functions:

$$p(\text{Heads} \mid \theta) = \theta,$$

for $\theta \in \Theta = (0,1)$.

Coin Flipping: Bayesian Model

- Recall that we have a parametric family of mass functions:

$$p(\text{Heads} \mid \theta) = \theta,$$

  for $\theta \in \Theta = (0,1)$.
- We need a prior distribution $p(\theta)$ on $\Theta = (0,1)$.

- Recall that we have a parametric family of mass functions:

$$p(\text{Heads} \mid \theta) = \theta,$$

  for $\theta \in \Theta = (0,1)$.
- We need a prior distribution $p(\theta)$ on $\Theta = (0,1)$.
- One convenient choice would be a distribution from the Beta family

Coin Flipping: Beta Prior

- **Prior:**

$$
\begin{aligned}
\theta &\sim \text{Beta}(\alpha, \beta) \\
p(\theta) &\propto \theta^{\alpha-1}(1-\theta)^{\beta-1}
\end{aligned}
$$

Figure by Horas based on the work of Krishnavedala (Own work) [Public domain], via Wikimedia Commons
http://commons.wikimedia.org/wiki/File:Beta_distribution_pdf.svg.

Coin Flipping: Beta Prior

- **Prior:**

$$\theta \sim \text{Beta}(\alpha, \beta)$$
$$p(\theta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

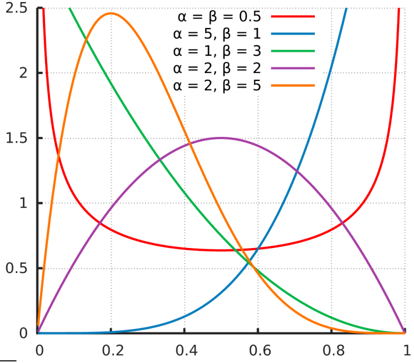


Figure by Horas based on the work of Krishnavedala (Own work) [Public domain], via Wikimedia Commons http://commons.wikimedia.org/wiki/File:Beta_distribution_pdf.svg.

Coin Flipping: Beta Prior

- **Prior:**

$$\begin{aligned}\theta &\sim \text{Beta}(h, t)\\ p(\theta) &\propto \theta^{h-1}(1-\theta)^{t-1}\end{aligned}$$

Coin Flipping: Beta Prior

- **Prior:**

$$
\begin{aligned}
\theta &\sim \text{Beta}(h, t) \\
p(\theta) &\propto \theta^{h-1}(1-\theta)^{t-1}
\end{aligned}
$$

- **Mean of Beta distribution:**

$$
\mathbb{E}\theta = \frac{h}{h+t}
$$

Coin Flipping: Beta Prior

- **Prior:**

$$\begin{aligned} \theta &\sim \text{Beta}(h, t) \\ p(\theta) &\propto \theta^{h-1}(1-\theta)^{t-1} \end{aligned}$$

- **Mean of Beta distribution:**

$$\mathbb{E}\theta = \frac{h}{h+t}$$

- **Mode of Beta distribution:**

$$\arg\max_{\theta} p(\theta) = \frac{h-1}{h+t-2}$$

for $h, t > 1$.

Coin Flipping: Posterior

- **Prior:**

$$\begin{aligned} \theta &\sim \text{Beta}(h, t) \\ p(\theta) &\propto \theta^{h-1}(1-\theta)^{t-1} \end{aligned}$$

Coin Flipping: Posterior

- **Prior:**

$$\begin{aligned}\theta &\sim \text{Beta}(h,t)\\ p(\theta) &\propto \theta^{h-1}(1-\theta)^{t-1}\end{aligned}$$

- **Likelihood function**

$$L(\theta) = p(\mathcal{D} \mid \theta) = \theta^{n_h}(1-\theta)^{n_t}$$

Coin Flipping: Posterior

- **Prior:**

$$\begin{aligned} \theta &\sim \text{Beta}(h,t) \\ p(\theta) &\propto \theta^{h-1}(1-\theta)^{t-1} \end{aligned}$$

- **Likelihood function**

$$L(\theta) = p(\mathcal{D} \mid \theta) = \theta^{n_h}(1-\theta)^{n_t}$$

- **Posterior density:**

$$p(\theta \mid \mathcal{D}) \quad \propto \quad p(\theta)p(\mathcal{D} \mid \theta)$$

Coin Flipping: Posterior

- **Prior:**

$$\begin{aligned} \theta &\sim \text{Beta}(h,t) \\ p(\theta) &\propto \theta^{h-1}(1-\theta)^{t-1} \end{aligned}$$

- **Likelihood function**

$$L(\theta) = p(\mathcal{D} \mid \theta) = \theta^{n_h}(1-\theta)^{n_t}$$

- **Posterior density:**

$$\begin{aligned} p(\theta \mid \mathcal{D}) &\propto p(\theta)p(\mathcal{D} \mid \theta) \\ &\propto \theta^{h-1}(1-\theta)^{t-1} \times \theta^{n_h}(1-\theta)^{n_t} \end{aligned}$$

Coin Flipping: Posterior

- **Prior:**

$$
\begin{aligned}
\theta &\sim \text{Beta}(h,t) \\
p(\theta) &\propto \theta^{h-1}(1-\theta)^{t-1}
\end{aligned}
$$

- **Likelihood function**

$$
L(\theta) = p(\mathcal{D} \mid \theta) = \theta^{n_h}(1-\theta)^{n_t}
$$

- **Posterior density:**

$$
\begin{aligned}
p(\theta \mid \mathcal{D}) &\propto p(\theta)p(\mathcal{D} \mid \theta) \\
&\propto \theta^{h-1}(1-\theta)^{t-1} \times \theta^{n_h}(1-\theta)^{n_t} \\
&= \theta^{h-1+n_h}(1-\theta)^{t-1+n_t}
\end{aligned}
$$

The Posterior is in the Beta Family!

- **Prior:**

$$\begin{aligned} \theta &\sim \text{Beta}(h, t) \\ p(\theta) &\propto \theta^{h-1}(1-\theta)^{t-1} \end{aligned}$$

- **Posterior density:**

$$p(\theta \mid \mathcal{D}) \propto \theta^{h-1+n_h}(1-\theta)^{t-1+n_t}$$

The Posterior is in the Beta Family!

- **Prior:**

$$\begin{aligned} \theta &\sim \text{Beta}(h, t) \\ p(\theta) &\propto \theta^{h-1}(1-\theta)^{t-1} \end{aligned}$$

- **Posterior density:**

$$p(\theta \mid \mathcal{D}) \propto \theta^{h-1+n_h}(1-\theta)^{t-1+n_t}$$

- **Posterior is in the beta family**:

$$\theta \mid \mathcal{D} \sim \text{Beta}(h+n_h, t+n_t)$$

The Posterior is in the Beta Family!

- **Prior:**

$$\begin{aligned} \theta &\sim \text{Beta}(h,t) \\ p(\theta) &\propto \theta^{h-1}(1-\theta)^{t-1} \end{aligned}$$

- **Posterior density:**

$$p(\theta \mid \mathcal{D}) \propto \theta^{h-1+n_h}(1-\theta)^{t-1+n_t}$$

- **Posterior is in the beta family**:

$$\theta \mid \mathcal{D} \sim \text{Beta}(h+n_h, t+n_t)$$

- **Interpretation**:
  - Prior initializes our counts with h heads and t tails.
  - Posterior increments counts by observed n_h and n_t.

Sidebar: Conjugate Priors

- In this case, the posterior is in the same distribution family as the prior.
- Let π be a family of prior distributions on Θ.

Sidebar: Conjugate Priors

- In this case, the posterior is in the same distribution family as the prior.
- Let π be a family of prior distributions on Θ.
- Let P parametric family of distributions with parameter space Θ.

Sidebar: Conjugate Priors

- In this case, the posterior is in the same distribution family as the prior.
- Let π be a family of prior distributions on Θ.
- Let P parametric family of distributions with parameter space Θ.

Definition

A family of distributions π **is conjugate to** parametric model P if for any prior in π, the posterior is always in π.

Sidebar: Conjugate Priors

- In this case, the posterior is in the same distribution family as the prior.
- Let π be a family of prior distributions on Θ.
- Let P parametric family of distributions with parameter space Θ.

Definition

A family of distributions π **is conjugate to** parametric model P if for any prior in π, the posterior is always in π.

- The beta family is conjugate to the coin-flipping (i.e. Bernoulli) model.

Coin Flipping: Concrete Example

- Suppose we have a coin, possibly biased (**parametric probability model**):

$$p(\text{Heads} \mid \theta) = \theta.$$

Coin Flipping: Concrete Example

- Suppose we have a coin, possibly biased (**parametric probability model**):

$$p(\text{Heads} \mid \theta) = \theta.$$

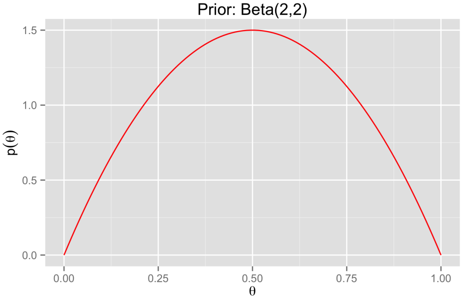
- **Parameter space** $\theta \in \Theta = [0,1]$.
- **Prior distribution:** $\theta \sim \text{Beta}(2,2)$.

Coin Flipping: Concrete Example

- Suppose we have a coin, possibly biased (**parametric probability model**):

$$p(\text{Heads} \mid \theta) = \theta.$$

- **Parameter space** $\theta \in \Theta = [0,1]$.
- **Prior distribution:** $\theta \sim \text{Beta}(2,2)$.



Example: Coin Flipping

- Next, we gather some data $\mathcal{D} = \{H, H, T, T, T, T, T, H, \ldots, T\}$:

Example: Coin Flipping

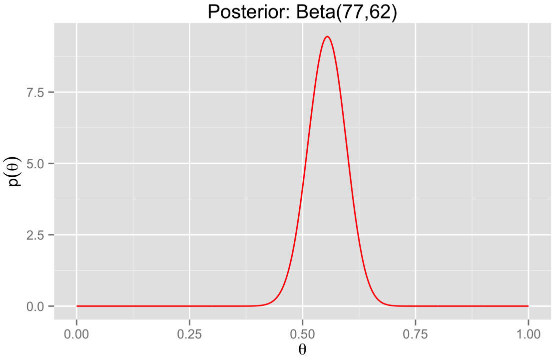
- Next, we gather some data $\mathcal{D} = \{H, H, T, T, T, T, T, H, \ldots, T\}$:
- Heads: 75 Tails: 60

Example: Coin Flipping

- Next, we gather some data $\mathcal{D} = \{H, H, T, T, T, T, T, H, \ldots, T\}$:
- Heads: 75 Tails: 60
  - $\hat{\theta}_{\text{MLE}} = \frac{75}{75+60} \approx 0.556$

Example: Coin Flipping

- Next, we gather some data $\mathcal{D} = \{H, H, T, T, T, T, T, H, \ldots, T\}$:
- Heads: 75 Tails: 60
  - $\hat{\theta}_{\text{MLE}} = \frac{75}{75+60} \approx 0.556$
- **Posterior distribution:** $\theta \mid \mathcal{D} \sim \textbf{Beta}(77, 62)$:



Bayesian Point Estimates

- We have the posterior distribution $\theta \mid \mathcal{D}$.
- What if someone asks us for a point estimate $\hat{\theta}$ for θ?

Bayesian Point Estimates

- We have the posterior distribution $\theta \mid \mathcal{D}$.
- What if someone asks us for a point estimate $\hat{\theta}$ for θ?
- Common options:

Bayesian Point Estimates

- We have the posterior distribution $\theta \mid \mathcal{D}$.
- What if someone asks us for a point estimate $\hat{\theta}$ for θ?
- Common options:
  - **posterior mean** $\hat{\theta} = \mathbb{E}[\theta \mid \mathcal{D}]$

Bayesian Point Estimates

- We have the posterior distribution $\theta \mid \mathcal{D}$.
- What if someone asks us for a point estimate $\hat{\theta}$ for θ?
- Common options:
  - **posterior mean** $\hat{\theta} = \mathbb{E}[\theta \mid \mathcal{D}]$
  - **maximum a posteriori (MAP) estimate** $\hat{\theta} = \arg\max_\theta p(\theta \mid \mathcal{D})$
    - Note: this is the **mode** of the posterior distribution

What else can we do with a posterior?

- Look at it: display uncertainty estimates to our client

What else can we do with a posterior?

- Look at it: display uncertainty estimates to our client
- Extract a **credible set** for θ (a Bayesian confidence interval).
  - e.g. Interval $[a, b]$ is a 95% **credible set** if

$$\mathbb{P}(\theta \in [a, b] \mid \mathcal{D}) \geqslant 0.95$$

What else can we do with a posterior?

- Look at it: display uncertainty estimates to our client
- Extract a **credible set** for θ (a Bayesian confidence interval).
  - e.g. Interval $[a, b]$ is a 95% **credible set** if

$$\mathbb{P}(\theta \in [a, b] \mid \mathcal{D}) \geqslant 0.95$$

- Select a point estimate using **Bayesian decision theory**:
  - Choose a loss function.
  - Find action **minimizing expected risk w.r.t. posterior**

Bayesian Decision Theory

- Ingredients:
  - **Parameter space** Θ.
  - **Prior**: Distribution $p(\theta)$ on Θ.
  - **Action space** $\mathcal{A}$.
  - **Loss function**: $\ell : \mathcal{A} \times \Theta \to R$.

Bayesian Decision Theory

- Ingredients:
  - **Parameter space** Θ.
  - **Prior**: Distribution $p(\theta)$ on Θ.
  - **Action space** $\mathcal{A}$.
  - **Loss function**: $\ell : \mathcal{A} \times \Theta \to \mathrm{R}$.
- The **posterior risk** of an action $a \in \mathcal{A}$ is

$$r(a) \quad := \quad \mathbb{E}\left[\ell(\theta, a) \mid \mathcal{D}\right]$$

Bayesian Decision Theory

- Ingredients:
  - **Parameter space** Θ.
  - **Prior**: Distribution $p(\theta)$ on Θ.
  - **Action space** $\mathcal{A}$.
  - **Loss function**: $\ell : \mathcal{A} \times \Theta \to \mathrm{R}$.
- The **posterior risk** of an action $a \in \mathcal{A}$ is

$$
\begin{aligned}
r(a) &:= \mathbb{E}\left[\ell(\theta, a) \mid \mathcal{D}\right] \\
&= \int \ell(\theta, a) p(\theta \mid \mathcal{D})\, d\theta.
\end{aligned}
$$

Bayesian Decision Theory

- Ingredients:
  - **Parameter space** Θ.
  - **Prior**: Distribution $p(\theta)$ on Θ.
  - **Action space** $\mathcal{A}$.
  - **Loss function**: $\ell : \mathcal{A} \times \Theta \to \mathbf{R}$.
- The **posterior risk** of an action $a \in \mathcal{A}$ is

$$
\begin{aligned}
r(a) &:= \mathbb{E}\left[\ell(\theta, a) \mid \mathcal{D}\right] \\
&= \int \ell(\theta, a) p(\theta \mid \mathcal{D})\, d\theta.
\end{aligned}
$$

  - It's the **expected loss under the posterior.**

Bayesian Decision Theory

- Ingredients:
    - **Parameter space** Θ.
    - **Prior**: Distribution $p(\theta)$ on Θ.
    - **Action space** $\mathcal{A}$.
    - **Loss function**: $\ell : \mathcal{A} \times \Theta \to \mathrm{R}$.
- The **posterior risk** of an action $a \in \mathcal{A}$ is

$$
\begin{aligned}
r(a) &:= \mathbb{E}\left[\ell(\theta, a) \mid \mathcal{D}\right] \\
&= \int \ell(\theta, a) p(\theta \mid \mathcal{D})\, d\theta .
\end{aligned}
$$

    - It's the **expected loss under the posterior.**
- A **Bayes action** a^* is an action that minimizes posterior risk:

$$
r(a^*) = \min_{a \in \mathcal{A}} r(a)
$$

Bayesian Point Estimation

- General Setup:
  - Data $\mathcal{D}$ generated by $p(y \mid \theta)$, for unknown $\theta \in \Theta$.

Bayesian Point Estimation

- General Setup:
  - Data $\mathcal{D}$ generated by $p(y \mid \theta)$, for unknown $\theta \in \Theta$.
  - We want to produce a **point estimate** for θ.

Bayesian Point Estimation

- General Setup:
  - Data $\mathcal{D}$ generated by $p(y \mid \theta)$, for unknown $\theta \in \Theta$.
  - We want to produce a **point estimate** for θ.
- Choose:

Bayesian Point Estimation

- General Setup:
  - Data $\mathcal{D}$ generated by $p(y \mid \theta)$, for unknown $\theta \in \Theta$.
  - We want to produce a **point estimate** for θ.
- Choose:
  - **Prior** $p(\theta)$ on $\Theta = \mathbf{R}$.

Bayesian Point Estimation

- General Setup:
  - Data $\mathcal{D}$ generated by $p(y \mid \theta)$, for unknown $\theta \in \Theta$.
  - We want to produce a **point estimate** for θ.
- Choose:
  - **Prior** $p(\theta)$ on $\Theta = \mathbf{R}$.
  - **Loss** $\ell(\hat{\theta}, \theta)$

Bayesian Point Estimation

- General Setup:
  - Data $\mathcal{D}$ generated by $p(y \mid \theta)$, for unknown $\theta \in \Theta$.
  - We want to produce a **point estimate** for θ.
- Choose:
  - **Prior** $p(\theta)$ on $\Theta = R$.
  - **Loss** $\ell(\hat{\theta}, \theta)$
- Find **action** $\hat{\theta} \in \Theta$ that minimizes the **posterior risk:**

$$r(\hat{\theta}) = \mathbb{E}\left[\ell(\hat{\theta}, \theta) \mid \mathcal{D}\right]$$

Bayesian Point Estimation

- General Setup:
  - Data $\mathcal{D}$ generated by $p(y \mid \theta)$, for unknown $\theta \in \Theta$.
  - We want to produce a **point estimate** for θ.
- Choose:
  - **Prior** $p(\theta)$ on $\Theta = \mathrm{R}$.
  - **Loss** $\ell(\hat{\theta}, \theta)$
- Find **action** $\hat{\theta} \in \Theta$ that minimizes the **posterior risk:**

$$
\begin{aligned}
r(\hat{\theta}) &= \mathbb{E}\left[\ell(\hat{\theta}, \theta) \mid \mathcal{D}\right] \\
&= \int \ell(\hat{\theta}, \theta) p(\theta \mid \mathcal{D})\, d\theta
\end{aligned}
$$

Important Cases

- Squared Loss : $\ell(\hat{\theta}, \theta) = \left(\theta - \hat{\theta}\right)^2 \quad \Rightarrow$ posterior mean
- Zero-one Loss: $\ell(\theta, \hat{\theta}) = \mathbb{1}[\theta \neq \hat{\theta}] \quad \Rightarrow$ posterior mode
- Absolute Loss : $\ell(\hat{\theta}, \theta) = \left|\theta - \hat{\theta}\right| \quad \Rightarrow$ posterior median

Important Cases

- Squared Loss : $\ell(\hat{\theta}, \theta) = \left(\theta - \hat{\theta}\right)^2 \quad \Rightarrow$ posterior mean
- Zero-one Loss: $\ell(\theta, \hat{\theta}) = \mathbb{1}[\theta \neq \hat{\theta}] \quad \Rightarrow$ posterior mode
- Absolute Loss : $\ell(\hat{\theta}, \theta) = \left|\theta - \hat{\theta}\right| \quad \Rightarrow$ posterior median
- Optimal decision depends on the loss function and the posterior distribution.
- We will derive the square loss case next.

Bayesian Point Estimation: Square Loss

- Find **action** $\hat{\theta} \in \Theta$ that minimizes **posterior risk**

$$r(\hat{\theta}) = \int \left(\theta - \hat{\theta}\right)^2 p(\theta \mid \mathcal{D})\, d\theta.$$

Bayesian Point Estimation: Square Loss

- Find **action** $\hat{\theta} \in \Theta$ that minimizes **posterior risk**

$$r(\hat{\theta}) = \int \left(\theta - \hat{\theta}\right)^2 p(\theta \mid \mathcal{D})\, d\theta.$$

- Differentiate:

$$\frac{dr(\hat{\theta})}{d\hat{\theta}} = -\int 2\left(\theta - \hat{\theta}\right) p(\theta \mid \mathcal{D})\, d\theta$$

Bayesian Point Estimation: Square Loss

- Find **action** $\hat{\theta} \in \Theta$ that minimizes **posterior risk**

$$r(\hat{\theta}) = \int \left(\theta - \hat{\theta}\right)^2 p(\theta \mid \mathcal{D})\, d\theta.$$

- Differentiate:

$$\begin{aligned} \frac{dr(\hat{\theta})}{d\hat{\theta}} &= -\int 2\left(\theta - \hat{\theta}\right) p(\theta \mid \mathcal{D})\, d\theta \\ &= -2\int \theta p(\theta \mid \mathcal{D})\, d\theta + 2\hat{\theta} \underbrace{\int p(\theta \mid \mathcal{D})\, d\theta}_{=1} \end{aligned}$$

Bayesian Point Estimation: Square Loss

- Find **action** $\hat{\theta} \in \Theta$ that minimizes **posterior risk**

$$r(\hat{\theta}) = \int \left(\theta - \hat{\theta}\right)^2 p(\theta \mid \mathcal{D})\, d\theta.$$

- Differentiate:

$$\begin{aligned}
\frac{dr(\hat{\theta})}{d\hat{\theta}} &= -\int 2\left(\theta - \hat{\theta}\right) p(\theta \mid \mathcal{D})\, d\theta \\
&= -2\int \theta p(\theta \mid \mathcal{D})\, d\theta + 2\hat{\theta} \underbrace{\int p(\theta \mid \mathcal{D})\, d\theta}_{=1} \\
&= -2\int \theta p(\theta \mid \mathcal{D})\, d\theta + 2\hat{\theta}
\end{aligned}$$

Bayesian Point Estimation: Square Loss

- Derivative of posterior risk is

$$\frac{dr(\hat{\theta})}{d\hat{\theta}} = -2\int \theta p(\theta \mid \mathcal{D})\, d\theta + 2\hat{\theta}.$$

Bayesian Point Estimation: Square Loss

- Derivative of posterior risk is

$$\frac{dr(\hat{\theta})}{d\hat{\theta}} = -2\int \theta p(\theta \mid \mathcal{D})\, d\theta + 2\hat{\theta}.$$

- First order condition $\frac{dr(\hat{\theta})}{d\hat{\theta}} = 0$ gives

$$\hat{\theta} = \int \theta p(\theta \mid \mathcal{D})\, d\theta$$

Bayesian Point Estimation: Square Loss

- Derivative of posterior risk is

$$\frac{dr(\hat{\theta})}{d\hat{\theta}} = -2\int \theta p(\theta \mid \mathcal{D})\,d\theta + 2\hat{\theta}.$$

- First order condition $\frac{dr(\hat{\theta})}{d\hat{\theta}} = 0$ gives

$$\begin{aligned}\hat{\theta} &= \int \theta p(\theta \mid \mathcal{D})\,d\theta \\ &= \mathbb{E}[\theta \mid \mathcal{D}]\end{aligned}$$

Bayesian Point Estimation: Square Loss

- Derivative of posterior risk is

$$\frac{dr(\hat{\theta})}{d\hat{\theta}} = -2\int \theta p(\theta \mid \mathcal{D})\,d\theta + 2\hat{\theta}.$$

- First order condition $\frac{dr(\hat{\theta})}{d\hat{\theta}} = 0$ gives

$$\begin{aligned}\hat{\theta} &= \int \theta p(\theta \mid \mathcal{D})\,d\theta \\ &= \mathbb{E}[\theta \mid \mathcal{D}]\end{aligned}$$

- The **Bayes action** for **square loss** is the posterior mean.

Recap and Interpretation

- The prior represents belief about θ before observing data $\mathcal{D}$.
- The posterior represents **rationally updated beliefs** after seeing $\mathcal{D}$.

Recap and Interpretation

- The prior represents belief about θ before observing data $\mathcal{D}$.
- The posterior represents **rationally updated beliefs** after seeing $\mathcal{D}$.
- All inferences and action-taking are based on the posterior distribution.

Recap and Interpretation

- The prior represents belief about θ before observing data $\mathcal{D}$.
- The posterior represents **rationally updated beliefs** after seeing $\mathcal{D}$.
- All inferences and action-taking are based on the posterior distribution.
- In the Bayesian approach,
  - No issue of justifying an estimator.

Recap and Interpretation

- The prior represents belief about θ before observing data $\mathcal{D}$.
- The posterior represents **rationally updated beliefs** after seeing $\mathcal{D}$.
- All inferences and action-taking are based on the posterior distribution.
- In the Bayesian approach,
  - No issue of justifying an estimator.
  - Only choices are
    - **family of distributions**, indexed by Θ, and
    - **prior distribution** on Θ

Recap and Interpretation

- The prior represents belief about θ before observing data $\mathcal{D}$.
- The posterior represents **rationally updated beliefs** after seeing $\mathcal{D}$.
- All inferences and action-taking are based on the posterior distribution.
- In the Bayesian approach,
  - No issue of justifying an estimator.
  - Only choices are
    - **family of distributions**, indexed by Θ, and
    - **prior distribution** on Θ
  - For decision making, we need a **loss function**.

Conditional Probability Modeling

- **Input space** $\mathcal{X}$
- **Outcome space** $\mathcal{Y}$
- **Action space** $\mathcal{A} = \{p(y) \mid p \text{ is a probability distribution on } \mathcal{Y}\}$.

Conditional Probability Modeling

- **Input space** $\mathcal{X}$
- **Outcome space** $\mathcal{Y}$
- **Action space** $\mathcal{A} = \{p(y) \mid p \text{ is a probability distribution on } \mathcal{Y}\}$.
- **Hypothesis space** $\mathcal{F}$ contains prediction functions $f : \mathcal{X} \to \mathcal{A}$.
- **Prediction function** $f \in \mathcal{F}$ takes input $x \in \mathcal{X}$ and produces a **distribution** on $\mathcal{Y}$

Conditional Probability Modeling

- **Input space** $\mathcal{X}$
- **Outcome space** $\mathcal{Y}$
- **Action space** $\mathcal{A} = \{p(y) \mid p \text{ is a probability distribution on } \mathcal{Y}\}$.
- **Hypothesis space** $\mathcal{F}$ contains prediction functions $f : \mathcal{X} \to \mathcal{A}$.
- **Prediction function** $f \in \mathcal{F}$ takes input $x \in \mathcal{X}$ and produces a **distribution** on $\mathcal{Y}$
- A **parametric family of conditional densities** is a set

$$\{p(y \mid x, \theta) : \theta \in \Theta\},$$

  - where $p(y \mid x, \theta)$ is a density on **outcome space** $\mathcal{Y}$ for each x in **input space** $\mathcal{X}$, and
  - θ is a **parameter** in a [finite dimensional] **parameter space** Θ.

Conditional Probability Modeling

- **Input space** $\mathcal{X}$
- **Outcome space** $\mathcal{Y}$
- **Action space** $\mathcal{A} = \{p(y) \mid p$ is a probability distribution on $\mathcal{Y}\}$.
- **Hypothesis space** $\mathcal{F}$ contains prediction functions $f : \mathcal{X} \to \mathcal{A}$.
- **Prediction function** $f \in \mathcal{F}$ takes input $x \in \mathcal{X}$ and produces a **distribution** on $\mathcal{Y}$
- A **parametric family of conditional densities** is a set

$$\{p(y \mid x, \theta) : \theta \in \Theta\},$$

  - where $p(y \mid x, \theta)$ is a density on **outcome space** $\mathcal{Y}$ for each x in **input space** $\mathcal{X}$, and
  - θ is a **parameter** in a [finite dimensional] **parameter space** Θ**.**

- This is the common starting point for either classical or Bayesian regression.

Classical treatment: Likelihood Function

- **Data:** $\mathcal{D} = (y_1, \ldots, y_n)$
- The probability density for our data $\mathcal{D}$ is

$$p(\mathcal{D} \mid x_1, \ldots, x_n, \theta) = \prod_{i=1}^{n} p(y_i \mid x_i, \theta).$$

Classical treatment: Likelihood Function

- **Data:** $\mathcal{D} = (y_1, \ldots, , y_n)$
- The probability density for our data $\mathcal{D}$ is

$$p(\mathcal{D} \mid x_1, \ldots, x_n, \theta) \quad = \quad \prod_{i=1}^{n} p(y_i \mid x_i, \theta).$$

- For fixed $\mathcal{D}$, the function $\theta \mapsto p(\mathcal{D} \mid x, \theta)$ is the **likelihood function**:

$$L_{\mathcal{D}}(\theta) = p(\mathcal{D} \mid x, \theta),$$

where $x = (x_1, \ldots, x_n)$.

Maximum Likelihood Estimator

- The **maximum likelihood estimator (MLE)** for θ in the family $\{p(y \mid x, \theta) \mid \theta \in \Theta\}$ is

$$\hat{\theta}_{\text{MLE}} = \underset{\theta \in \Theta}{\arg\max}\, L_{\mathcal{D}}(\theta).$$

- MLE corresponds to ERM, if we set the loss to be the negative log-likelihood.

Maximum Likelihood Estimator

- The **maximum likelihood estimator (MLE)** for θ in the family $\{p(y \mid x, \theta) \mid \theta \in \Theta\}$ is

$$\hat{\theta}_{\text{MLE}} = \underset{\theta \in \Theta}{\arg\max}\, L_{\mathcal{D}}(\theta).$$

- MLE corresponds to ERM, if we set the loss to be the negative log-likelihood.
- The corresponding prediction function is

$$\hat{f}(x) = p(y \mid x, \hat{\theta}_{\text{MLE}}).$$

Bayesian Conditional Models

- Input space $\mathcal{X} = \mathbf{R}^d$ Outcome space $\mathcal{Y} = \mathbf{R}$

Bayesian Conditional Models

- Input space $\mathcal{X} = \mathbf{R}^d$ Outcome space $\mathcal{Y} = \mathbf{R}$
- The Bayesian conditional model has two components:
  - A **parametric family of conditional densities**:

$$\{p(y \mid x, \theta) : \theta \in \Theta\}$$

Bayesian Conditional Models

- Input space $\mathcal{X} = \mathbf{R}^d$ Outcome space $\mathcal{Y} = \mathbf{R}$
- The Bayesian conditional model has two components:
  - A **parametric family of conditional densities**:

$$\{p(y \mid x, \theta) : \theta \in \Theta\}$$

  - A **prior distribution** $p(\theta)$ on $\theta \in \Theta$.

The Posterior Distribution

- The **prior distribution** $p(\theta)$ represents our beliefs about θ before seeing $\mathcal{D}$.

The Posterior Distribution

- The **prior distribution** $p(\theta)$ represents our beliefs about θ before seeing $\mathcal{D}$.
- The **posterior distribution** for θ is

$$p(\theta \mid \mathcal{D}, x)$$

The Posterior Distribution

- The **prior distribution** $p(\theta)$ represents our beliefs about θ before seeing $\mathcal{D}$.
- The **posterior distribution** for θ is

$$p(\theta \mid \mathcal{D}, x) \quad \propto \quad p(\mathcal{D} \mid \theta, x) p(\theta)$$

The Posterior Distribution

- The **prior distribution** $p(\theta)$ represents our beliefs about θ before seeing $\mathcal{D}$.
- The **posterior distribution** for θ is

$$
\begin{aligned}
p(\theta \mid \mathcal{D}, x) &\propto p(\mathcal{D} \mid \theta, x) p(\theta) \\
&= \underbrace{L_{\mathcal{D}}(\theta)}_{\text{likelihood}} \underbrace{p(\theta)}_{\text{prior}}
\end{aligned}
$$

The Posterior Distribution

- The **prior distribution** $p(\theta)$ represents our beliefs about θ before seeing $\mathcal{D}$.
- The **posterior distribution** for θ is

$$\begin{aligned} p(\theta \mid \mathcal{D}, x) &\propto p(\mathcal{D} \mid \theta, x) p(\theta) \\ &= \underbrace{L_{\mathcal{D}}(\theta)}_{\text{likelihood}} \underbrace{p(\theta)}_{\text{prior}} \end{aligned}$$

- Posterior represents the **rationally updated beliefs** after seeing $\mathcal{D}$.

The Posterior Distribution

- The **prior distribution** $p(\theta)$ represents our beliefs about θ before seeing $\mathcal{D}$.
- The **posterior distribution** for θ is

$$\begin{aligned} p(\theta \mid \mathcal{D}, x) &\propto p(\mathcal{D} \mid \theta, x) p(\theta) \\ &= \underbrace{L_{\mathcal{D}}(\theta)}_{\text{likelihood}} \underbrace{p(\theta)}_{\text{prior}} \end{aligned}$$

- Posterior represents the **rationally updated beliefs** after seeing $\mathcal{D}$.
- Each θ corresponds to a prediction function,
  - i.e. the conditional distribution function $p(y \mid x, \theta)$.

Point Estimates of Parameter

- What if we want point estimates of θ?

Point Estimates of Parameter

- What if we want point estimates of θ?
- We can use **Bayesian decision theory** to derive point estimates.

Point Estimates of Parameter

- What if we want point estimates of θ?
- We can use **Bayesian decision theory** to derive point estimates.
- We may want to use
  - $\hat{\theta} = \mathbb{E}[\theta \mid \mathcal{D}, x]$ (the posterior mean estimate)
  - $\hat{\theta} = \text{median}[\theta \mid \mathcal{D}, x]$
  - $\hat{\theta} = \arg\max_{\theta \in \Theta} p(\theta \mid \mathcal{D}, x)$ (the MAP estimate)
- depending on our loss function.

Back to the basic question - Bayesian Prediction Function

- Find a function takes input $x \in \mathcal{X}$ and produces a **distribution** on $\mathcal{Y}$

Back to the basic question - Bayesian Prediction Function

- Find a function takes input $x \in \mathcal{X}$ and produces a **distribution** on $\mathcal{Y}$
- In the frequentist approach:
  - Choose family of conditional probability densities (hypothesis space).
  - Select one conditional probability from family, e.g. using MLE.

Back to the basic question - Bayesian Prediction Function

- Find a function takes input $x \in \mathcal{X}$ and produces a **distribution** on $\mathcal{Y}$
- In the frequentist approach:
  - Choose family of conditional probability densities (hypothesis space).
  - Select one conditional probability from family, e.g. using MLE.
- In the Bayesian setting:

Back to the basic question - Bayesian Prediction Function

- Find a function takes input $x \in \mathcal{X}$ and produces a **distribution** on $\mathcal{Y}$
- In the frequentist approach:
  - Choose family of conditional probability densities (hypothesis space).
  - Select one conditional probability from family, e.g. using MLE.
- In the Bayesian setting:
  - We choose a parametric family of conditional densities

$$\{p(y \mid x, \theta) : \theta \in \Theta\},$$

  - and a prior distribution $p(\theta)$ on this set.

Back to the basic question - Bayesian Prediction Function

- Find a function takes input $x \in \mathcal{X}$ and produces a **distribution** on $\mathcal{Y}$
- In the frequentist approach:
  - Choose family of conditional probability densities (hypothesis space).
  - Select one conditional probability from family, e.g. using MLE.
- In the Bayesian setting:
  - We choose a parametric family of conditional densities

$$\{p(y \mid x, \theta) : \theta \in \Theta\},$$

  - and a prior distribution $p(\theta)$ on this set.
- Having set our Bayesian model, how do we predict a distribution on y for input x?
- We don't need to make a discrete selection from the hypothesis space: we **maintain uncertainty**.

The Prior Predictive Distribution

- Suppose we have not yet observed any data.

The Prior Predictive Distribution

- Suppose we have not yet observed any data.
- In the Bayesian setting, we can still produce a prediction function.

The Prior Predictive Distribution

- Suppose we have not yet observed any data.
- In the Bayesian setting, we can still produce a prediction function.
- The **prior predictive distribution** is given by

$$x \mapsto p(y \mid x)$$

The Prior Predictive Distribution

- Suppose we have not yet observed any data.
- In the Bayesian setting, we can still produce a prediction function.
- The **prior predictive distribution** is given by

$$x \mapsto p(y \mid x) = \int p(y \mid x; \theta) p(\theta) \, d\theta.$$

The Prior Predictive Distribution

- Suppose we have not yet observed any data.
- In the Bayesian setting, we can still produce a prediction function.
- The **prior predictive distribution** is given by

$$x \mapsto p(y \mid x) = \int p(y \mid x; \theta) p(\theta) \, d\theta.$$

- This is an average of all conditional densities in our family, weighted by the prior.

The Posterior Predictive Distribution

- Suppose we've already seen data $\mathcal{D}$.

The Posterior Predictive Distribution

- Suppose we've already seen data $\mathcal{D}$.
- The **posterior predictive distribution** is given by

$$x \mapsto p(y \mid x, \mathcal{D})$$

The Posterior Predictive Distribution

- Suppose we've already seen data $\mathcal{D}$.
- The **posterior predictive distribution** is given by

$$x \mapsto p(y \mid x, \mathcal{D}) = \int p(y \mid x; \theta) p(\theta \mid \mathcal{D}) \, d\theta.$$

The Posterior Predictive Distribution

- Suppose we've already seen data $\mathcal{D}$.
- The **posterior predictive distribution** is given by

$$x \mapsto p(y \mid x, \mathcal{D}) = \int p(y \mid x; \theta) p(\theta \mid \mathcal{D})\, d\theta.$$

- This is an average of all conditional densities in our family, weighted by the posterior.

Comparison to Frequentist Approach

- In Bayesian statistics we have two distributions on Θ:
  - the prior distribution $p(\theta)$
  - the posterior distribution $p(\theta \mid \mathcal{D})$.

Comparison to Frequentist Approach

- In Bayesian statistics we have two distributions on Θ:
  - the prior distribution $p(\theta)$
  - the posterior distribution $p(\theta \mid \mathcal{D})$.
- These distributions over parameters correspond to distributions on the hypothesis space:

$$\{p(y \mid x, \theta) : \theta \in \Theta\}.$$

Comparison to Frequentist Approach

- In Bayesian statistics we have two distributions on Θ:
  - the prior distribution $p(\theta)$
  - the posterior distribution $p(\theta \mid \mathcal{D})$.
- These distributions over parameters correspond to distributions on the hypothesis space:

$$\{p(y \mid x, \theta) : \theta \in \Theta\}.$$

- In the frequentist approach, we choose $\hat{\theta} \in \Theta$, and predict

$$p(y \mid x, \hat{\theta}(\mathcal{D})).$$

Comparison to Frequentist Approach

- In Bayesian statistics we have two distributions on Θ:
  - the prior distribution $p(\theta)$
  - the posterior distribution $p(\theta \mid \mathcal{D})$.
- These distributions over parameters correspond to distributions on the hypothesis space:

$$\{p(y \mid x, \theta) : \theta \in \Theta\}.$$

- In the frequentist approach, we choose $\hat{\theta} \in \Theta$, and predict

$$p(y \mid x, \hat{\theta}(\mathcal{D})).$$

- In the Bayesian approach, we integrate out over Θ w.r.t. $p(\theta \mid \mathcal{D})$ and predict with

$$p(y \mid x, \mathcal{D}) = \int p(y \mid x; \theta) p(\theta \mid \mathcal{D}) \, d\theta$$

What if we don't want a full distribution on y?

- Once we have a predictive distribution $p(y \mid x, \mathcal{D})$,
  - we can easily generate single point predictions.

What if we don't want a full distribution on y?

- Once we have a predictive distribution $p(y \mid x, \mathcal{D})$,
  - we can easily generate single point predictions.
- $x \mapsto \mathbb{E}\left[y \mid x, \mathcal{D}\right]$, to minimize expected square error.

What if we don't want a full distribution on y?

- Once we have a predictive distribution $p(y \mid x, \mathcal{D})$,
  - we can easily generate single point predictions.
- $x \mapsto \mathbb{E}[y \mid x, \mathcal{D}]$, to minimize expected square error.
- $x \mapsto \text{median}[y \mid x, \mathcal{D}]$, to minimize expected absolute error

What if we don't want a full distribution on y?

- Once we have a predictive distribution $p(y \mid x, \mathcal{D})$,
  - we can easily generate single point predictions.
- $x \mapsto \mathbb{E}[y \mid x, \mathcal{D}]$, to minimize expected square error.
- $x \mapsto \text{median}[y \mid x, \mathcal{D}]$, to minimize expected absolute error
- $x \mapsto \arg\max_{y \in \mathcal{Y}} p(y \mid x, \mathcal{D})$, to minimize expected 0/1 loss

What if we don't want a full distribution on y?

- Once we have a predictive distribution $p(y \mid x, \mathcal{D})$,
  - we can easily generate single point predictions.
- $x \mapsto \mathbb{E}[y \mid x, \mathcal{D}]$, to minimize expected square error.
- $x \mapsto \text{median}[y \mid x, \mathcal{D}]$, to minimize expected absolute error
- $x \mapsto \arg\max_{y \in \mathcal{Y}} p(y \mid x, \mathcal{D})$, to minimize expected 0/1 loss
- Each of these can be derived from $p(y \mid x, \mathcal{D})$.

Example in 1-Dimension: Setup

- Input space $\mathcal{X} = [-1, 1]$ Output space $\mathcal{Y} = \mathbf{R}$
- Given x, the world generates y as

$$y = w_0 + w_1 x + \varepsilon,$$

where $\varepsilon \sim \mathcal{N}(0, 0.2^2)$.

Example in 1-Dimension: Setup

- Input space $\mathcal{X} = [-1, 1]$ Output space $\mathcal{Y} = \mathrm{R}$
- Given x, the world generates y as

$$y = w_0 + w_1 x + \varepsilon,$$

where $\varepsilon \sim \mathcal{N}(0, 0.2^2)$.

- Written another way, the **conditional probability model** is

$$y \mid x, w_0, w_1 \sim \mathcal{N}\left(w_0 + w_1 x, 0.2^2\right).$$

- What's the parameter space?

Example in 1-Dimension: Setup

- Input space $\mathcal{X} = [-1, 1]$ Output space $\mathcal{Y} = \mathbf{R}$
- Given x, the world generates y as

$$y = w_0 + w_1 x + \varepsilon,$$

where $\varepsilon \sim \mathcal{N}(0, 0.2^2)$.

- Written another way, the **conditional probability model** is

$$y \mid x, w_0, w_1 \sim \mathcal{N}\left(w_0 + w_1 x\,,\, 0.2^2\right).$$

- What's the parameter space? $\mathbf{R}^2$.

Example in 1-Dimension: Setup

- Input space $\mathcal{X} = [-1, 1]$ Output space $\mathcal{Y} = \mathbf{R}$
- Given x, the world generates y as

$$y = w_0 + w_1 x + \varepsilon,$$

where $\varepsilon \sim \mathcal{N}(0, 0.2^2)$.

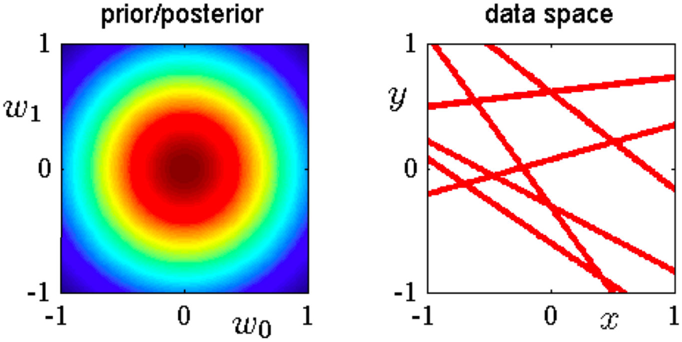
- Written another way, the **conditional probability model** is

$$y \mid x, w_0, w_1 \sim \mathcal{N}\left(w_0 + w_1 x \,,\, 0.2^2\right).$$

- What's the parameter space? $\mathbf{R}^2$.
- **Prior distribution:** $w = (w_0, w_1) \sim \mathcal{N}\left(0, \frac{1}{2} I\right)$

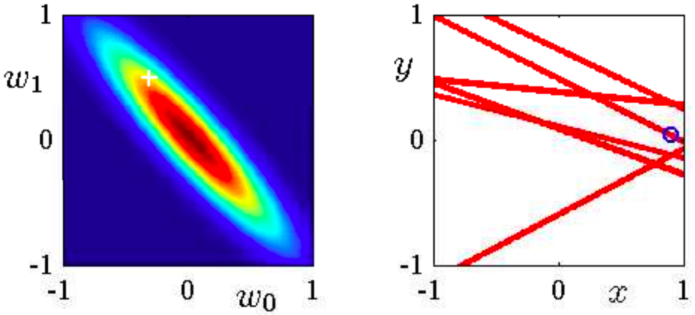
Example in 1-Dimension: Prior Situation

- **Prior distribution:** $w = (w_0, w_1) \sim \mathcal{N}\left(0, \frac{1}{2}I\right)$ (Illustrated on left)



- On right, $y(x) = \mathbb{E}[y \mid x, w] = w_0 + w_1 x$, for randomly chosen $w \sim p(w) = \mathcal{N}\left(0, \frac{1}{2}I\right)$.

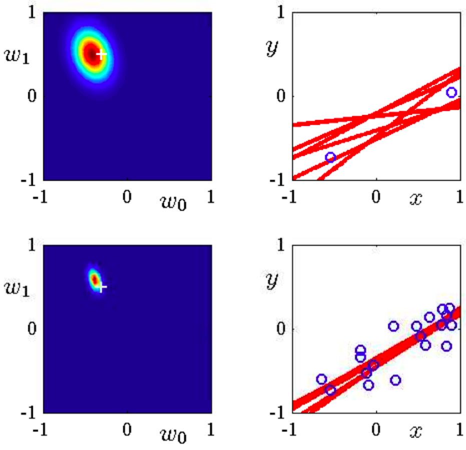
Bishop's PRML Fig 3.7



- On left: posterior distribution; white cross indicates true parameters
- On right:
  - blue circle indicates the training observation
  - red lines, $y(x) = \mathbb{E}[y \mid x, w] = w_0 + w_1 x$, for randomly chosen $w \sim p(w|\mathcal{D})$ (posterior)

Bishop's PRML Fig 3.7

Example in 1-Dimension: 2 and 20 Observations



Bishop's PRML Fig 3.7

Closed Form for Posterior

- Model:

$$w \sim \mathcal{N}(0, \Sigma_0)$$

Closed Form for Posterior

- Model:

$$
\begin{array}{rcl}
w & \sim & \mathcal{N}(0, \Sigma_0) \\
y_i \mid x, w & \text{i.i.d.} & \mathcal{N}(w^T x_i, \sigma^2)
\end{array}
$$

Closed Form for Posterior

- Model:

$$
\begin{aligned}
w &\sim \mathcal{N}(0, \Sigma_0) \\
y_i \mid x, w &\ \text{i.i.d.}\ \mathcal{N}(w^T x_i, \sigma^2)
\end{aligned}
$$

- Design matrix X Response column vector y

Closed Form for Posterior

- Model:

$$
\begin{aligned}
w &\sim \mathcal{N}(0, \Sigma_0) \\
y_i \mid x, w &\quad \text{i.i.d.} \quad \mathcal{N}(w^T x_i, \sigma^2)
\end{aligned}
$$

- Design matrix X Response column vector y
- **Posterior distribution is a Gaussian distribution:**

$$
w \mid \mathcal{D} \sim
$$

Closed Form for Posterior

- Model:

$$\begin{aligned} w &\sim \mathcal{N}(0, \Sigma_0) \\ y_i \mid x, w &\text{ i.i.d. } \mathcal{N}(w^T x_i, \sigma^2) \end{aligned}$$

- Design matrix X Response column vector y
- **Posterior distribution is a Gaussian distribution:**

$$\begin{aligned} w \mid \mathcal{D} &\sim \mathcal{N}(\mu_P, \Sigma_P) \\ \mu_{\mathbf{P}} &= \left(X^T X + \sigma^2 \Sigma_0^{-1}\right)^{-1} X^T y \\ \Sigma_{\mathbf{P}} &= \left(\sigma^{-2} X^T X + \Sigma_0^{-1}\right)^{-1} \end{aligned}$$

Closed Form for Posterior

- Model:

$$\begin{aligned} w &\sim \mathcal{N}(0, \Sigma_0) \\ y_i \mid x, w &\;\text{i.i.d.}\; \mathcal{N}(w^T x_i, \sigma^2) \end{aligned}$$

- Design matrix X Response column vector y
- **Posterior distribution is a Gaussian distribution:**

$$\begin{aligned} w \mid \mathcal{D} &\sim \mathcal{N}(\mu_P, \Sigma_P) \\ \mu_{\mathbf{P}} &= \left(X^T X + \sigma^2 \Sigma_0^{-1}\right)^{-1} X^T y \\ \Sigma_{\mathbf{P}} &= \left(\sigma^{-2} X^T X + \Sigma_0^{-1}\right)^{-1} \end{aligned}$$

- **Posterior Variance** Σ_P gives us a natural **uncertainty measure.**

Closed Form for Posterior

- **Posterior distribution is a Gaussian distribution:**

$$w \mid \mathcal{D} \quad \sim$$

Closed Form for Posterior

- **Posterior distribution is a Gaussian distribution:**

$$
\begin{aligned}
w \mid \mathcal{D} &\sim \mathcal{N}(\mu_P, \Sigma_P) \\
\mu_{\mathbf{P}} &= \left(X^T X + \sigma^2 \Sigma_0^{-1}\right)^{-1} X^T y \\
\Sigma_{\mathbf{P}} &= \left(\sigma^{-2} X^T X + \Sigma_0^{-1}\right)^{-1}
\end{aligned}
$$

Closed Form for Posterior

- **Posterior distribution is a Gaussian distribution:**

$$
\begin{aligned}
w \mid \mathcal{D} &\sim \mathcal{N}(\mu_P, \Sigma_P) \\
\mu_{\mathbf{P}} &= \left(X^T X + \sigma^2 \Sigma_0^{-1}\right)^{-1} X^T y \\
\Sigma_{\mathbf{P}} &= \left(\sigma^{-2} X^T X + \Sigma_0^{-1}\right)^{-1}
\end{aligned}
$$

- If we want point estimates of w, **MAP estimator** and the **posterior mean** are given by

Closed Form for Posterior

- **Posterior distribution is a Gaussian distribution:**

$$\begin{aligned} w \mid \mathcal{D} &\sim \mathcal{N}(\mu_P, \Sigma_P) \\ \mu_{\mathbf{P}} &= \left(X^T X + \sigma^2 \Sigma_0^{-1}\right)^{-1} X^T y \\ \Sigma_{\mathbf{P}} &= \left(\sigma^{-2} X^T X + \Sigma_0^{-1}\right)^{-1} \end{aligned}$$

- If we want point estimates of w, **MAP estimator** and the **posterior mean** are given by

$$\hat{w} = \mu_P = \left(X^T X + \sigma^2 \Sigma_0^{-1}\right)^{-1} X^T y$$

Closed Form for Posterior

- **Posterior distribution is a Gaussian distribution:**

$$
\begin{aligned}
w \mid \mathcal{D} &\sim \mathcal{N}(\mu_P, \Sigma_P) \\
\mu_{\mathbf{P}} &= \left(X^T X + \sigma^2 \Sigma_0^{-1}\right)^{-1} X^T y \\
\Sigma_{\mathbf{P}} &= \left(\sigma^{-2} X^T X + \Sigma_0^{-1}\right)^{-1}
\end{aligned}
$$

- If we want point estimates of w, **MAP estimator** and the **posterior mean** are given by

$$
\hat{w} = \mu_P = \left(X^T X + \sigma^2 \Sigma_0^{-1}\right)^{-1} X^T y
$$

- For the prior variance $\Sigma_0 = \frac{\sigma^2}{\lambda} I$, we get

$$
\hat{w} = \mu_P = \left(X^T X + \lambda I\right)^{-1} X^T y,
$$

Closed Form for Posterior

- **Posterior distribution is a Gaussian distribution:**

$$
\begin{aligned}
w \mid \mathcal{D} &\sim \mathcal{N}(\mu_P, \Sigma_P) \\
\mu_{\mathbf{P}} &= \left(X^T X + \sigma^2 \Sigma_0^{-1}\right)^{-1} X^T y \\
\Sigma_{\mathbf{P}} &= \left(\sigma^{-2} X^T X + \Sigma_0^{-1}\right)^{-1}
\end{aligned}
$$

- If we want point estimates of w, **MAP estimator** and the **posterior mean** are given by

$$
\hat{w} = \mu_P = \left(X^T X + \sigma^2 \Sigma_0^{-1}\right)^{-1} X^T y
$$

- For the prior variance $\Sigma_0 = \frac{\sigma^2}{\lambda} I$, we get

$$
\hat{w} = \mu_P = \left(X^T X + \lambda I\right)^{-1} X^T y,
$$

which is of course the ridge regression solution.

Connection the MAP to Ridge Regression

- The **Posterior density** on w for $\Sigma_0 = \frac{\sigma^2}{\lambda} I$:

$$p(w \mid \mathcal{D}) \quad \propto \quad \underbrace{\exp\left(-\frac{\lambda}{2\sigma^2}\|w\|^2\right)}_{\textbf{prior}} \underbrace{\prod_{i=1}^{n} \exp\left(-\frac{(y_i - w^T x_i)^2}{2\sigma^2}\right)}_{\textbf{likelihood}}$$

Connection the MAP to Ridge Regression

- The **Posterior density** on w for $\Sigma_0 = \frac{\sigma^2}{\lambda} I$:

$$p(w \mid \mathcal{D}) \quad \propto \quad \underbrace{\exp\left(-\frac{\lambda}{2\sigma^2}\|w\|^2\right)}_{\textbf{prior}} \underbrace{\prod_{i=1}^{n} \exp\left(-\frac{(y_i - w^T x_i)^2}{2\sigma^2}\right)}_{\textbf{likelihood}}$$

- To find the **MAP**, we minimize the negative log posterior:

$$\hat{w}_{\text{MAP}} \quad = \quad \underset{w \in \mathbb{R}^d}{\arg\min} \left[-\log p(w \mid \mathcal{D})\right]$$

Connection the MAP to Ridge Regression

- The **Posterior density** on w for $\Sigma_0 = \frac{\sigma^2}{\lambda} I$:

$$p(w \mid \mathcal{D}) \quad \propto \quad \underbrace{\exp\left(-\frac{\lambda}{2\sigma^2}\|w\|^2\right)}_{\textbf{prior}} \underbrace{\prod_{i=1}^{n} \exp\left(-\frac{(y_i - w^T x_i)^2}{2\sigma^2}\right)}_{\textbf{likelihood}}$$

- To find the **MAP**, we minimize the negative log posterior:

$$\begin{aligned}
\hat{w}_{\text{MAP}} &= \underset{w \in \mathbb{R}^d}{\arg\min} \left[-\log p(w \mid \mathcal{D})\right] \\
&= \underset{w \in \mathbb{R}^d}{\arg\min} \underbrace{\sum_{i=1}^{n} (y_i - w^T x_i)^2}_{\text{log-likelihood}} + \underbrace{\lambda \|w\|^2}_{\text{log-prior}}
\end{aligned}$$

Connection the MAP to Ridge Regression

- The **Posterior density** on w for $\Sigma_0 = \frac{\sigma^2}{\lambda} I$:

$$p(w \mid \mathcal{D}) \quad \propto \quad \underbrace{\exp\left(-\frac{\lambda}{2\sigma^2}\|w\|^2\right)}_{\textbf{prior}} \underbrace{\prod_{i=1}^{n} \exp\left(-\frac{(y_i - w^T x_i)^2}{2\sigma^2}\right)}_{\textbf{likelihood}}$$

- To find the **MAP**, we minimize the negative log posterior:

$$\begin{aligned} \hat{w}_{\mathrm{MAP}} &= \underset{w \in \mathbb{R}^d}{\arg\min} \left[-\log p(w \mid \mathcal{D})\right] \\ &= \underset{w \in \mathbb{R}^d}{\arg\min} \underbrace{\sum_{i=1}^{n} (y_i - w^T x_i)^2}_{\text{log-likelihood}} + \underbrace{\lambda \|w\|^2}_{\text{log-prior}} \end{aligned}$$

- Which is the ridge regression objective.

Predictive Posterior Distribution

- Given a new input point x_{new}, how do we predict y_{new} ?

Predictive Posterior Distribution

- Given a new input point x_{new}, how do we predict y_{new} ?
- **Predictive distribution**

$$p(y_{\textbf{new}} \mid x_{\textbf{new}}, \mathcal{D}) \quad =$$

Predictive Posterior Distribution

- Given a new input point x_{new}, how do we predict y_{new} ?
- **Predictive distribution**

$$p(y_{\textbf{new}} \mid x_{\textbf{new}}, \mathcal{D}) \quad = \quad \int p(y_{\textbf{new}} \mid x_{\textbf{new}}, w, \mathcal{D}) p(w \mid \mathcal{D})\, dw$$

Predictive Posterior Distribution

- Given a new input point x_{new}, how do we predict y_{new} ?
- **Predictive distribution**

$$
\begin{aligned}
p(y_{\textbf{new}} \mid x_{\textbf{new}}, \mathcal{D}) &= \int p(y_{\textbf{new}} \mid x_{\textbf{new}}, w, \mathcal{D}) p(w \mid \mathcal{D})\, dw \\
&= \int p(y_{\textbf{new}} \mid x_{\textbf{new}}, w) p(w \mid \mathcal{D})\, dw
\end{aligned}
$$

Predictive Posterior Distribution

- Given a new input point x_{new}, how do we predict y_{new} ?
- **Predictive distribution**

$$
\begin{aligned}
p(y_{\textbf{new}} \mid x_{\textbf{new}}, \mathcal{D}) &= \int p(y_{\textbf{new}} \mid x_{\textbf{new}}, w, \mathcal{D}) p(w \mid \mathcal{D})\, dw \\
&= \int p(y_{\textbf{new}} \mid x_{\textbf{new}}, w) p(w \mid \mathcal{D})\, dw
\end{aligned}
$$

- For Gaussian regression, predictive distribution has closed form.

Closed Form for Predictive Distribution

- **Model**:

$$w \sim \mathcal{N}(0, \Sigma_0)$$

- **Model**:

$$
\begin{aligned}
w &\sim \mathcal{N}(0, \Sigma_0) \\
y_i \mid x, w &\ \text{i.i.d.}\ \mathcal{N}(w^T x_i, \sigma^2)
\end{aligned}
$$

Closed Form for Predictive Distribution

- **Model**:

$$
\begin{aligned}
w &\sim \mathcal{N}(0, \Sigma_0) \\
y_i \mid x, w &\text{ i.i.d. } \mathcal{N}(w^T x_i, \sigma^2)
\end{aligned}
$$

- **Predictive Distribution**

$$
p(y_{\text{new}} \mid x_{\text{new}}, \mathcal{D}) = \int p(y_{\text{new}} \mid x_{\text{new}}, w) p(w \mid \mathcal{D}) \, dw.
$$

  - Averages over prediction for each w, weighted by posterior distribution.

Closed Form for Predictive Distribution

- **Model**:

$$\begin{array}{rcl} w & \sim & \mathcal{N}(0, \Sigma_0) \\ y_i \mid x, w & \text{i.i.d.} & \mathcal{N}(w^T x_i, \sigma^2) \end{array}$$

- **Predictive Distribution**

$$p(y_{\text{new}} \mid x_{\text{new}}, \mathcal{D}) = \int p(y_{\text{new}} \mid x_{\text{new}}, w) p(w \mid \mathcal{D})\, dw.$$

  - Averages over prediction for each w, weighted by posterior distribution.

- **Closed form:**

$$y_{\text{new}} \mid x_{\text{new}}, \mathcal{D} \sim \mathcal{N}\left(\eta_{\text{new}}, \sigma^2_{\text{new}}\right)$$

Closed Form for Predictive Distribution

- **Model**:

$$
\begin{aligned}
w &\sim \mathcal{N}(0, \Sigma_0) \\
y_i \mid x, w &\quad \text{i.i.d.} \quad \mathcal{N}(w^T x_i, \sigma^2)
\end{aligned}
$$

- **Predictive Distribution**

$$
p(y_{\text{new}} \mid x_{\text{new}}, \mathcal{D}) = \int p(y_{\text{new}} \mid x_{\text{new}}, w) p(w \mid \mathcal{D})\, dw.
$$

  - Averages over prediction for each w, weighted by posterior distribution.

- **Closed form:**

$$
\begin{aligned}
y_{\text{new}} \mid x_{\text{new}}, \mathcal{D} &\sim \mathcal{N}\left(\eta_{\text{new}}, \sigma^2_{\text{new}}\right) \\
\eta_{\text{new}} &= \mu_P^T x_{\text{new}}
\end{aligned}
$$

Closed Form for Predictive Distribution

- **Model**:

$$\begin{array}{rcl} w & \sim & \mathcal{N}(0, \Sigma_0) \\ y_i \mid x, w & \text{i.i.d.} & \mathcal{N}(w^T x_i, \sigma^2) \end{array}$$

- **Predictive Distribution**

$$p(y_{\text{new}} \mid x_{\text{new}}, \mathcal{D}) = \int p(y_{\text{new}} \mid x_{\text{new}}, w) p(w \mid \mathcal{D})\, dw.$$

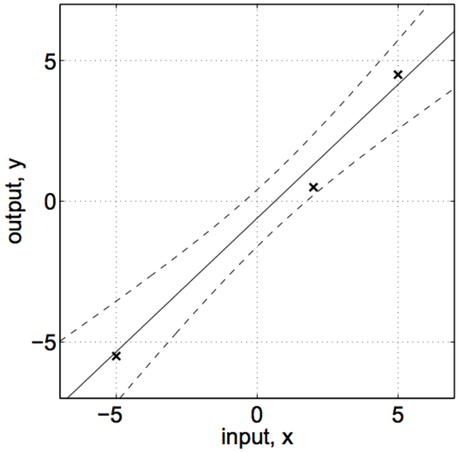
  - Averages over prediction for each w, weighted by posterior distribution.

- **Closed form:**

$$\begin{array}{rcl} y_{\text{new}} \mid x_{\text{new}}, \mathcal{D} & \sim & \mathcal{N}\left(\eta_{\text{new}}, \sigma^2_{\text{new}}\right) \\ \eta_{\text{new}} & = & \mu_P^T x_{\text{new}} \\ \sigma^2_{\text{new}} & = & \underbrace{x_{\text{new}}^T \Sigma_P x_{\text{new}}}_{\text{from variance in } w} + \underbrace{\sigma^2}_{\text{inherent variance in } y} \end{array}$$

Bayesian Regression Provides Uncertainty Estimates

- With predictive distributions, we can give mean prediction with error bands:



Rasmussen and Williams' *Gaussian Processes for Machine Learning*, Fig.2.1(b)

Motivation

- So far, most algorithms we've learned are designed for binary classification.
    - Sentiment analysis (positive vs. negative)
    - Spam filter (spam vs. non-spam)

Motivation

- So far, most algorithms we've learned are designed for binary classification.
  - Sentiment analysis (positive vs. negative)
  - Spam filter (spam vs. non-spam)
- Many real-world problems have more than two classes.
  - Document classification (over 10 classes)
  - Object recognition (over 20k classes)
  - Face recognition (millions of classes)

Motivation

- So far, most algorithms we've learned are designed for binary classification.
  - Sentiment analysis (positive vs. negative)
  - Spam filter (spam vs. non-spam)
- Many real-world problems have more than two classes.
  - Document classification (over 10 classes)
  - Object recognition (over 20k classes)
  - Face recognition (millions of classes)
- What are some potential issues when we have a large number of classes?
  - Computation cost
  - Class imbalance
  - Different cost of errors

Today's lecture

- How to *reduce* multiclass classification to binary classification?
  - We can think of binary classifier or linear regression as a black box. Naive ways:
  - E.g. multiple binary classifiers produce a binary code for each class (000, 001, 010)
  - E.g. a linear regression produces a numerical value for each class (1.0, 2.0, 3.0)

Today's lecture

- How to *reduce* multiclass classification to binary classification?
  - We can think of binary classifier or linear regression as a black box. Naive ways:
  - E.g. multiple binary classifiers produce a binary code for each class (000, 001, 010)
  - E.g. a linear regression produces a numerical value for each class (1.0, 2.0, 3.0)
- How do we *generalize* binary classification algorithm to the multiclass setting?
  - We also need to think about the loss function.

Today's lecture

- How to *reduce* multiclass classification to binary classification?
  - We can think of binary classifier or linear regression as a black box. Naive ways:
  - E.g. multiple binary classifiers produce a binary code for each class (000, 001, 010)
  - E.g. a linear regression produces a numerical value for each class (1.0, 2.0, 3.0)
- How do we *generalize* binary classification algorithm to the multiclass setting?
  - We also need to think about the loss function.
- Example of very large output space: structured prediction.
  - Multi-class: Mutually exclusive class structure.
  - Text: Temporal relational structure.

One-vs-All / One-vs-Rest

Setting

- Input space: $\mathcal{X}$
- Output space: $\mathcal{Y} = \{1, \ldots, k\}$

One-vs-All / One-vs-Rest

Setting

- Input space: $\mathcal{X}$
- Output space: $\mathcal{Y} = \{1, \ldots, k\}$

Training

- Train k binary classifiers, one for each class: $h_1, \ldots, h_k : \mathcal{X} \to \mathrm{R}$.
- Classifier h_i distinguishes class i (+1) from the rest (-1).

One-vs-All / One-vs-Rest

Setting

- Input space: $\mathcal{X}$
- Output space: $\mathcal{Y} = \{1, \ldots, k\}$

Training

- Train k binary classifiers, one for each class: $h_1, \ldots, h_k : \mathcal{X} \to \mathrm{R}$.
- Classifier h_i distinguishes class i (+1) from the rest (-1).

Prediction

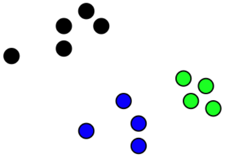
- Majority vote:

$$h(x) = \underset{i \in \{1, \ldots, k\}}{\arg\max}\, h_i(x)$$

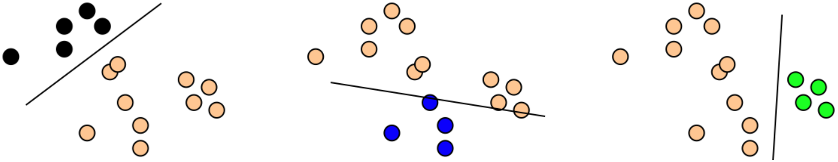
- Ties can be broken arbitrarily.

OvA: 3-class example (linear classifier)

<1-> Consider a dataset with three classes:

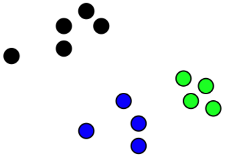


<2-> Train OvA classifiers:

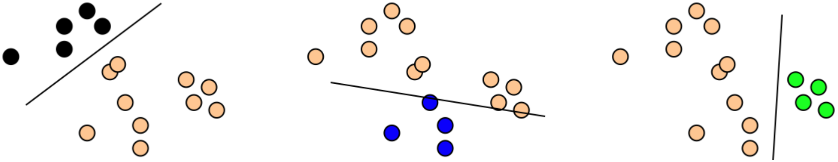


OvA: 3-class example (linear classifier)

<1-> Consider a dataset with three classes:

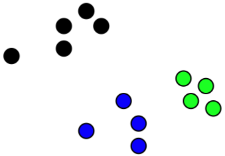


<2-> Train OvA classifiers:



OvA: 3-class example (linear classifier)

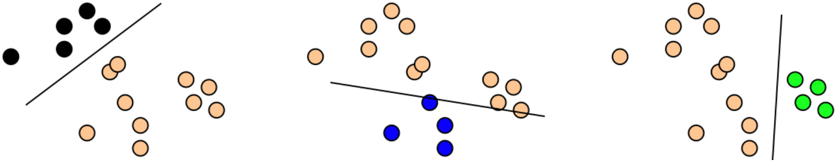
<1-> Consider a dataset with three classes:



Assumption: each class is linearly separable from the rest.

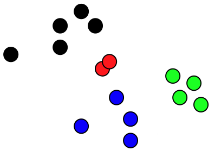
Ideal case: only target class has positive score.

<2-> Train OvA classifiers:

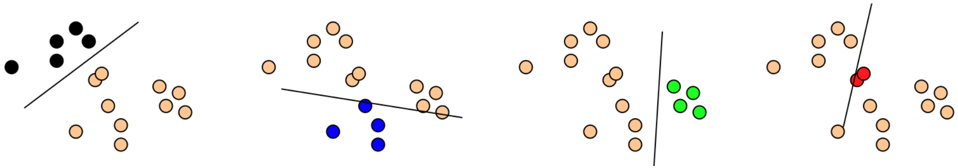


OvA: 4-class non linearly separable example

Consider a dataset with four classes:

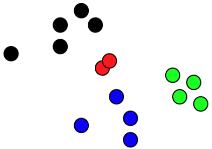


Train OvA classifiers:



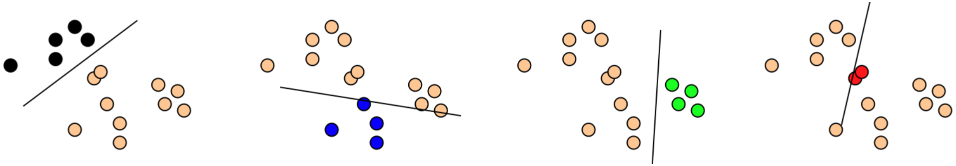
OvA: 4-class non linearly separable example

Consider a dataset with four classes:



Cannot separate red points from the rest.
Which classes might have low accuracy?

Train OvA classifiers:



All vs All / One vs One / All pairs

Setting

- Input space: $\mathcal{X}$
- Output space: $\mathcal{Y} = \{1, \ldots, k\}$

All vs All / One vs One / All pairs

Setting

- Input space: $\mathcal{X}$
- Output space: $\mathcal{Y} = \{1, \ldots, k\}$

Training

- Train $\binom{k}{2}$ binary classifiers, one for each pair: $h_{ij} : \mathcal{X} \to \mathrm{R}$ for $i \in [1, k]$ and $j \in [i+1, k]$.
- Classifier h_{ij} distinguishes class i (+1) from class j (-1).

All vs All / One vs One / All pairs

Setting

- Input space: $\mathcal{X}$
- Output space: $\mathcal{Y} = \{1, \ldots, k\}$

Training

- Train $\binom{k}{2}$ binary classifiers, one for each pair: $h_{ij} : \mathcal{X} \to \mathrm{R}$ for $i \in [1, k]$ and $j \in [i+1, k]$.
- Classifier h_{ij} distinguishes class i (+1) from class j (-1).

Prediction

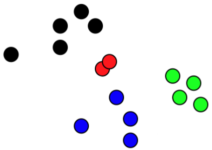
- Majority vote (each class gets $k-1$ votes)

$$h(x) = \underset{i \in \{1,\ldots,k\}}{\arg\max} \sum_{j \neq i} \underbrace{h_{ij}(x) i < j}_{\text{class } i \text{ is } +1} - \underbrace{h_{ji}(x) j < i}_{\text{class } i \text{ is -1}}$$

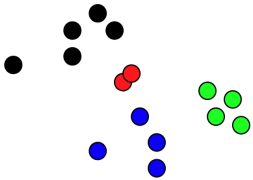
- Tournament
- Ties can be broken arbitrarily.

AvA: four-class example

Consider a dataset with four classes:

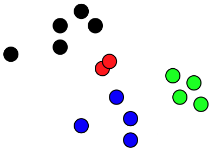


What's the decision region for the red class?



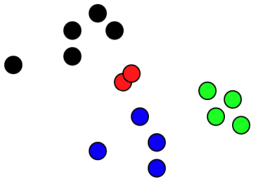
AvA: four-class example

Consider a dataset with four classes:



Assumption: each pair of classes are linearly separable.
More expressive than OvA.

What's the decision region for the red class?



OvA vs AvA

		OvA	AvA
computation	train	$O(k\qquad\qquad)$	$O(k^2\qquad\qquad)$
	test	$O(k\qquad)$	$O(k^2\qquad)$

OvA vs AvA

		OvA	AvA
computation	train	$O(kB_{\text{train}}(n))$	$O(k^2 B_{\text{train}}(n/k))$
	test	$O(kB_{\text{test}})$	$O(k^2 B_{\text{test}})$

challenges

OvA vs AvA

		OvA	AvA
computation	train	$O(kB_{\text{train}}(n))$	$O(k^2 B_{\text{train}}(n/k))$
	test	$O(kB_{\text{test}})$	$O(k^2 B_{\text{test}})$
challenges	train	class imbalance	small training set
	test	calibration / scale tie breaking	

Lack theoretical justification but simple to implement and works well in practice (when # classes is small).

Code word for labels

Using the reduction approach, can you train fewer than k binary classifiers?

Code word for labels

Using the reduction approach, can you train fewer than k binary classifiers?
Key idea: Encode labels as binary codes and predict the code bits directly.

Code word for labels

Using the reduction approach, can you train fewer than k binary classifiers?

Key idea: Encode labels as binary codes and predict the code bits directly.

OvA encoding:

class	h_1	h_2	h_3	h_4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Code word for labels

Using the reduction approach, can you train fewer than k binary classifiers?
Key idea: Encode labels as binary codes and predict the code bits directly.
OvA encoding:

class	h_1	h_2	h_3	h_4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

OvA uses k bits to encode each label, what's the minimal number of bits you can use?

Error correcting output codes (ECOC)

Example: 8 classes, 6-bit code

class	h_1	h_2	h_3	h_4	h_5	h_6
1	0	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	1	0	1	0
4	1	1	0	0	0	0
5	1	1	0	0	1	0
6	0	0	1	1	0	1
7	0	0	1	0	0	0
8	0	1	0	1	0	0

Training Binary classifier h_i:

- +1: classes whose i-th bit is 1
- -1: classes whose i-th bit is 0

Error correcting output codes (ECOC)

Example: 8 classes, 6-bit code

class	h_1	h_2	h_3	h_4	h_5	h_6
1	0	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	1	0	1	0
4	1	1	0	0	0	0
5	1	1	0	0	1	0
6	0	0	1	1	0	1
7	0	0	1	0	0	0
8	0	1	0	1	0	0

Training Binary classifier h_i:

- +1: classes whose i-th bit is 1
- -1: classes whose i-th bit is 0

Prediction Closest label in terms of Hamming distance.

h_1	h_2	h_3	h_4	h_5	h_6
0	1	1	0	1	1

Error correcting output codes (ECOC)

Example: 8 classes, 6-bit code

class	h_1	h_2	h_3	h_4	h_5	h_6
1	0	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	1	0	1	0
4	1	1	0	0	0	0
5	1	1	0	0	1	0
6	0	0	1	1	0	1
7	0	0	1	0	0	0
8	0	1	0	1	0	0

Training Binary classifier h_i:

- +1: classes whose i-th bit is 1
- -1: classes whose i-th bit is 0

Prediction Closest label in terms of Hamming distance.

h_1	h_2	h_3	h_4	h_5	h_6
0	1	1	0	1	1

Code design Want good binary classifiers.

Error correcting output codes: summary

- Computationally more efficient than OvA (a special case of ECOC). Better for large k.
- Why not use the minimal number of bits ($\log_2 k$)?

Error correcting output codes: summary

- Computationally more efficient than OvA (a special case of ECOC). Better for large k.
- Why not use the minimal number of bits ($\log_2 k$)?
  - If the minimum Hamming distance between any pair of code word is d, then it can correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors.
  - In plain words, if rows are far from each other, ECOC is robust to errors.

Error correcting output codes: summary

- Computationally more efficient than OvA (a special case of ECOC). Better for large k.
- Why not use the minimal number of bits ($\log_2 k$)?
  - If the minimum Hamming distance between any pair of code word is d, then it can correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors.
  - In plain words, if rows are far from each other, ECOC is robust to errors.
- Trade-off between code distance and binary classification performance.
- Nice theoretical results [Allwein et al., 2000] (also incoporates AvA).

Review

Reduction-based approaches:

- Reducing multiclass classification to binary classification: OvA, AvA
- Key is to design "natural" binary classification problems without large computation cost.

Review

Reduction-based approaches:

- Reducing multiclass classification to binary classification: OvA, AvA
- Key is to design "natural" binary classification problems without large computation cost.

But,

- Unclear how to generalize to extremely large # of classes.
- ImageNet: >20k labels; Wikipedia: >1M categories.

Next, generalize previous algorithms to multiclass settings.

Binary Logistic Regression

- Given an input x, we would like to output a classification between (0,1).

$$f(x) = sigmoid(z) = \frac{1}{1+\exp(-z)} = \frac{1}{1+\exp(-w^\top x - b)}. \tag{1}$$

Binary Logistic Regression

- Given an input x, we would like to output a classification between (0,1).

$$f(x) = sigmoid(z) = \frac{1}{1+\exp(-z)} = \frac{1}{1+\exp(-w^\top x - b)}. \tag{1}$$

- The other class is represented in $1 - f(x)$:

$$1 - f(x) = \frac{\exp(-w^\top x - b)}{1+\exp(-w^\top x - b)} = \frac{1}{1+\exp(w^\top x + b)} = sigmoid(-z). \tag{2}$$

Binary Logistic Regression

- Given an input x, we would like to output a classification between (0,1).

$$f(x) = sigmoid(z) = \frac{1}{1+\exp(-z)} = \frac{1}{1+\exp(-w^\top x - b)}. \tag{1}$$

- The other class is represented in $1 - f(x)$:

$$1 - f(x) = \frac{\exp(-w^\top x - b)}{1+\exp(-w^\top x - b)} = \frac{1}{1+\exp(w^\top x + b)} = sigmoid(-z). \tag{2}$$

- Another way to view: one class has $(+w, +b)$ and the other class has $(-w, -b)$.

Multi-class Logistic Regression

- Now what if we have one w_c for each class c?

Multi-class Logistic Regression

- Now what if we have one w_c for each class c?

$$f_c(x) = \frac{\exp(w_c^\top x) + b_c}{\sum_c \exp(w_c^\top x + b_c)} \tag{3}$$

- Also called "softmax" in neural networks.

Multi-class Logistic Regression

- Now what if we have one w_c for each class c?

$$f_c(x) = \frac{\exp(w_c^\top x) + b_c}{\sum_c \exp(w_c^\top x + b_c)} \tag{3}$$

- Also called "softmax" in neural networks.
- Loss function: $L = \sum_i -y_c^{(i)} \log f_c(x^{(i)})$

Multi-class Logistic Regression

- Now what if we have one w_c for each class c?

$$f_c(x) = \frac{\exp(w_c^\top x) + b_c}{\sum_c \exp(w_c^\top x + b_c)} \tag{3}$$

- Also called "softmax" in neural networks.
- Loss function: $L = \sum_i -y_c^{(i)} \log f_c(x^{(i)})$
- Gradient: $\frac{\partial L}{\partial z} = f - y$. Recall: MSE loss.

Comparison to OvA

- **Base Hypothesis Space**: $\mathcal{H} = \{h : \mathcal{X} \to \mathrm{R}\}$ (score functions).
- **Multiclass Hypothesis Space** (for k classes):

$$\mathcal{F} = \left\{ x \mapsto \arg\max_i h_i(x) \mid h_1, \ldots, h_k \in \mathcal{H} \right\}$$

Comparison to OvA

- **Base Hypothesis Space**: $\mathcal{H} = \{h : \mathcal{X} \to \mathrm{R}\}$ (score functions).
- **Multiclass Hypothesis Space** (for k classes):

$$\mathcal{F} = \left\{ x \mapsto \arg\max_i h_i(x) \mid h_1, \ldots, h_k \in \mathcal{H} \right\}$$

- Intuitively, $h_i(x)$ scores how likely x is to be from class i.
- OvA objective: $h_i(x) > 0$ for x with label i and $h_i(x) < 0$ for x with all other labels.

Comparison to OvA

- **Base Hypothesis Space**: $\mathcal{H} = \{h : \mathcal{X} \to \mathrm{R}\}$ (score functions).
- **Multiclass Hypothesis Space** (for k classes):

$$\mathcal{F} = \left\{ x \mapsto \arg\max_i h_i(x) \mid h_1, \ldots, h_k \in \mathcal{H} \right\}$$

- Intuitively, $h_i(x)$ scores how likely x is to be from class i.
- OvA objective: $h_i(x) > 0$ for x with label i and $h_i(x) < 0$ for x with all other labels.
- At test time, to predict (x, i) correctly we only need

$$h_i(x) > h_j(x) \qquad \forall j \neq i. \tag{4}$$

Multiclass Perceptron

- Base linear predictors: $h_i(x) = w_i^T x \ (w \in^d)$.

Multiclass Perceptron

- Base linear predictors: $h_i(x) = w_i^T x$ $(w \in^d)$.
- Multiclass perceptron:

Given a multiclass dataset $= (x, y)$;
Initialize $w \leftarrow 0$;
for $iter = 1, 2, \ldots, T$ **do**
  for $(x, y) \in$ **do**
    $\hat{y} = \arg\max_{y' \in} w_{y'}^T x$;
    if $\hat{y} \neq y$ **then** `// We've made a mistake`
      $w_y \leftarrow w_y + x$; `// Move the target-class scorer towards` x
      $w_{\hat{y}} \leftarrow w_{\hat{y}} - x$; `// Move the wrong-class scorer away from` x
    end
  end
end

Rewrite the scoring function

- Remember that we want to scale to very large # of classes and reuse algorithms and analysis for binary classification
  - $\Longrightarrow$ a single weight vector is desired
- How to rewrite the equation such that we have one w instead of k?

Rewrite the scoring function

- Remember that we want to scale to very large # of classes and reuse algorithms and analysis for binary classification
  - $\implies$ a single weight vector is desired
- How to rewrite the equation such that we have one w instead of k?

$$w_i^T x = w^T \psi(x, i) \tag{5}$$
$$h_i(x) = h(x, i) \tag{6}$$

  - Encode labels in the feature space.
  - Score for each label $\to$ score for the "*compatibility*" of a label and an input.

The Multivector Construction

How to construct the feature map ψ?

How to construct the feature map ψ?

- What if we stack w_i's together $(x \in^2, = 1, 2, 3)$

$$w = \left(\underbrace{-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}}_{w_1}, \underbrace{0, 1}_{w_2}, \underbrace{\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}}_{w_3} \right)$$

The Multivector Construction

How to construct the feature map ψ?

- What if we stack w_i's together ($x \in^2, = 1, 2, 3$)

$$w = \left(\underbrace{-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}}_{w_1}, \underbrace{0, 1}_{w_2}, \underbrace{\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}}_{w_3} \right)$$

- And then do the following: $\Psi : \mathrm{R}^2 \times \{1, 2, 3\} \to \mathrm{R}^6$ defined by

$$\begin{aligned}
\Psi(x, 1) &:= (x_1, x_2, 0, 0, 0, 0) \\
\Psi(x, 2) &:= (0, 0, x_1, x_2, 0, 0) \\
\Psi(x, 3) &:= (0, 0, 0, 0, x_1, x_2)
\end{aligned}$$

The Multivector Construction

How to construct the feature map ψ?

- What if we stack w_i's together ($x \in^2, = 1, 2, 3$)

$$w = \left(\underbrace{-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}}_{w_1}, \underbrace{0, 1}_{w_2}, \underbrace{\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}}_{w_3} \right)$$

- And then do the following: $\Psi : \mathrm{R}^2 \times \{1, 2, 3\} \to \mathrm{R}^6$ defined by

$$\begin{aligned} \Psi(x, 1) &:= (x_1, x_2, 0, 0, 0, 0) \\ \Psi(x, 2) &:= (0, 0, x_1, x_2, 0, 0) \\ \Psi(x, 3) &:= (0, 0, 0, 0, x_1, x_2) \end{aligned}$$

- Then $\langle w, \Psi(x, y) \rangle = \langle w_y, x \rangle$, which is what we want.

Rewrite multiclass perceptron

Multiclass perceptron using the multivector construction.

Given a multiclass dataset $= (x, y)$;
Initialize $w \leftarrow 0$;
for $iter = 1, 2, \ldots, T$ **do**
  for $(x, y) \in$ **do**
    $\hat{y} = \arg\max_{y' \in} w^T \psi(x, y')$; `// Equivalent to` $\arg\max_{y' \in} w_{y'}^T x$
    if $\hat{y} \neq y$ **then** `// We've made a mistake`
      $w \leftarrow w + \psi(x, y)$; `// Move the scorer towards` $\psi(x, y)$
      $w \leftarrow w - \psi(x, \hat{y})$; `// Move the scorer away from` $\psi(x, \hat{y})$
    end
  end
end

Rewrite multiclass perceptron

Multiclass perceptron using the multivector construction.

Given a multiclass dataset $= (x, y)$;
Initialize $w \leftarrow 0$;
for $iter = 1, 2, \ldots, T$ **do**
  for $(x, y) \in$ **do**
    $\hat{y} = \arg\max_{y' \in} w^T \psi(x, y')$; `// Equivalent to` $\arg\max_{y' \in} w_{y'}^T x$
    if $\hat{y} \neq y$ **then** `// We've made a mistake`
      $w \leftarrow w + \psi(x, y)$; `// Move the scorer towards` $\psi(x, y)$
      $w \leftarrow w - \psi(x, \hat{y})$; `// Move the scorer away from` $\psi(x, \hat{y})$
    end
  end
end

Exercise: What is the base binary classification problem in multiclass perceptron?

Features

Toy multiclass example: Part-of-speech classification

- $\mathcal{X} = \{\text{All possible words}\}$
- $\mathcal{Y} = \{\text{NOUN,VERB,ADJECTIVE,} \ldots\}$.

Features

Toy multiclass example: Part-of-speech classification

- $\mathcal{X} = \{\text{All possible words}\}$
- $\mathcal{Y} = \{\text{NOUN,VERB,ADJECTIVE,}\ldots\}$.
- Features of $x \in \mathcal{X}$: [The word itself], ENDS_IN_ly, ENDS_IN_ness, ...

How to construct the feature vector?

- Multivector construction: $w \in^{d \times k}$—doesn't scale.

Features

Toy multiclass example: Part-of-speech classification

- $\mathcal{X} = \{\text{All possible words}\}$
- $\mathcal{Y} = \{\text{NOUN,VERB,ADJECTIVE,}\ldots\}$.
- Features of $x \in \mathcal{X}$: [The word itself], ENDS_IN_ly, ENDS_IN_ness, ...

How to construct the feature vector?

- Multivector construction: $w \in^{d \times k}$—doesn't scale.
- Directly design features for each class.

$$\Psi(x,y) = (\psi_1(x,y), \psi_2(x,y), \psi_3(x,y), \ldots, \psi_d(x,y)) \tag{7}$$

- Size can be bounded by d.

Features

Sample training data:

`The boy grabbed the apple and ran away quickly .`

Features

Sample training data:

`The boy grabbed the apple and ran away quickly .`

Feature:

$$
\begin{aligned}
\psi_1(x,y) &= \mathbb{1}[x = \text{apple AND } y = \text{NOUN}] \\
\psi_2(x,y) &= \mathbb{1}[x = \text{run AND } y = \text{NOUN}] \\
\psi_3(x,y) &= \mathbb{1}[x = \text{run AND } y = \text{VERB}] \\
\psi_4(x,y) &= \mathbb{1}[x \text{ ENDS\_IN\_ly AND } y = \text{ADVERB}] \\
&\ldots
\end{aligned}
$$

Features

Sample training data:

```
The boy grabbed the apple and ran away quickly .
```

Feature:

$$
\begin{aligned}
\psi_1(x,y) &= \mathbb{1}[x = \text{apple AND } y = \text{NOUN}] \\
\psi_2(x,y) &= \mathbb{1}[x = \text{run AND } y = \text{NOUN}] \\
\psi_3(x,y) &= \mathbb{1}[x = \text{run AND } y = \text{VERB}] \\
\psi_4(x,y) &= \mathbb{1}[x \text{ ENDS\_IN\_ly AND } y = \text{ADVERB}] \\
&\ldots
\end{aligned}
$$

- E.g., $\Psi(x = \text{run}, y = \text{NOUN}) = (0, 1, 0, 0, \ldots)$

Features

Sample training data:

```
The boy grabbed the apple and ran away quickly .
```

Feature:

$$
\begin{aligned}
\psi_1(x,y) &= \mathbb{1}[x = \text{apple AND } y = \text{NOUN}] \\
\psi_2(x,y) &= \mathbb{1}[x = \text{run AND } y = \text{NOUN}] \\
\psi_3(x,y) &= \mathbb{1}[x = \text{run AND } y = \text{VERB}] \\
\psi_4(x,y) &= \mathbb{1}[x \text{ ENDS\_IN\_ly AND } y = \text{ADVERB}] \\
&\ldots
\end{aligned}
$$

- E.g., $\Psi(x = \text{run}, y = \text{NOUN}) = (0, 1, 0, 0, \ldots)$
- After training, what's w_1, w_2, w_3, w_4?

Features

Sample training data:

```
The boy grabbed the apple and ran away quickly .
```

Feature:

$$
\begin{aligned}
\psi_1(x,y) &= \mathbb{1}[x = \text{apple AND } y = \text{NOUN}] \\
\psi_2(x,y) &= \mathbb{1}[x = \text{run AND } y = \text{NOUN}] \\
\psi_3(x,y) &= \mathbb{1}[x = \text{run AND } y = \text{VERB}] \\
\psi_4(x,y) &= \mathbb{1}[x \text{ ENDS\_IN\_ly AND } y = \text{ADVERB}] \\
&\cdots
\end{aligned}
$$

- E.g., $\Psi(x = \text{run}, y = \text{NOUN}) = (0, 1, 0, 0, \ldots)$
- After training, what's w_1, w_2, w_3, w_4?
- No need to include features unseen in training data.

Feature templates: implementation

- Flexible, neighboring words, suffix/prefix.
- "Read off" features from the training data.
- Often sparse—efficient in practice, NLP problems.
- Can use a hash function: template $\rightarrow 1, 2, \ldots, d$.

Review

Ingredients in multiclass classification:

- Scoring functions for each class (similar to ranking).
- Represent labels in the input space $\Longrightarrow$ single weight vector.

Review

Ingredients in multiclass classification:

- Scoring functions for each class (similar to ranking).
- Represent labels in the input space $\Longrightarrow$ single weight vector.

We've seen

- How to generalize the perceptron algorithm to multiclass setting.
- Very simple idea. Was popular in NLP for structured prediction (tagging, parsing).

Review

Ingredients in multiclass classification:

- Scoring functions for each class (similar to ranking).
- Represent labels in the input space $\implies$ single weight vector.

We've seen

- How to generalize the perceptron algorithm to multiclass setting.
- Very simple idea. Was popular in NLP for structured prediction (tagging, parsing).

Next,

- How to generalize SVM to the multiclass setting.
- Concept check: Why might one prefer SVM / perceptron?

Margin for Multiclass

Binary

- Margin for (,):

$$w^T \tag{8}$$

- Want margin to be large and positive (w^T has same sign as)

Margin for Multiclass

Binary

- Margin for $(,)$:

$$w^T \tag{8}$$

- Want margin to be large and positive (w^T has same sign as)

Multiclass

- Class-specific margin for $(,)$:

$$h(,) - h(,y). \tag{9}$$

- Difference between scores of the correct class and each other class
- Want margin to be large and positive for all $y \neq$.

Multiclass SVM: separable case

Binary

$$\min_{w} \quad \frac{1}{2}\|w\|^2 \tag{10}$$

$$\text{s.t.} \quad \underbrace{w^T}_{\text{margin}} \geqslant 1 \quad \forall (,) \in \tag{11}$$

Multiclass SVM: separable case

Binary

$$\min_{w} \quad \frac{1}{2}\|w\|^2 \tag{10}$$

$$\text{s.t.} \quad \underbrace{w^{T}}_{\text{margin}} \geqslant 1 \quad \forall (,) \in \tag{11}$$

Multiclass As in the binary case, take 1 as our target margin.

$$m_{n,y}(w) \quad \underbrace{\langle w, \Psi(,)\rangle}_{\text{score of correct class}} - \underbrace{\langle w, \Psi(,y)\rangle}_{\text{score of other class}} \tag{12}$$

$$\min_{w} \quad \frac{1}{2}\|w\|^2 \tag{13}$$

$$\text{s.t.} \quad m_{n,y}(w) \geqslant 1 \quad \forall (,) \in, y \neq \tag{14}$$

Multiclass SVM: separable case

Binary

$$
\begin{aligned}
\min_{w} \quad & \frac{1}{2}\|w\|^2 && (10) \\
\text{s.t.} \quad & \underbrace{w^T}_{\text{margin}} \geqslant 1 \quad \forall(,) \in && (11)
\end{aligned}
$$

Multiclass As in the binary case, take 1 as our target margin.

$$
m_{n,y}(w) \quad \underbrace{\langle w, \Psi(,) \rangle}_{\text{score of correct class}} - \underbrace{\langle w, \Psi(,y) \rangle}_{\text{score of other class}} \qquad (12)
$$

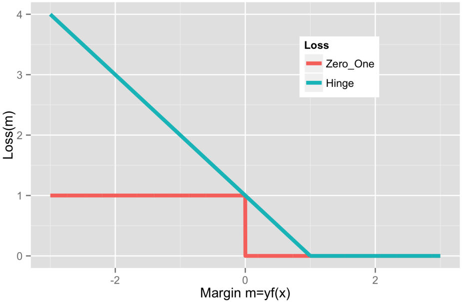
$$
\begin{aligned}
\min_{w} \quad & \frac{1}{2}\|w\|^2 && (13) \\
\text{s.t.} \quad & m_{n,y}(w) \geqslant 1 \quad \forall(,) \in, y \neq && (14)
\end{aligned}
$$

Exercise: write the objective for the non-separable case

Recap: hingle loss for binary classification

- Hinge loss: a convex upperbound on the 0-1 loss

$$\ell_{\text{hinge}}(y, \hat{y}) = \max(0, 1 - yh(x)) \tag{15}$$



Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$ \tag{16} $$

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = y \neq y' \tag{16}$$

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = y \neq y' \tag{16}$$

- In general, can also have different cost for each class.

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = y \neq y' \tag{16}$$

- In general, can also have different cost for each class.
- Upper bound on $\Delta(y, y')$.

$$\hat{y} \operatorname*{arg\,max}_{y' \in} \left\langle w, \Psi(x, y') \right\rangle \tag{17}$$

$$\tag{19}$$

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = y \neq y' \tag{16}$$

- In general, can also have different cost for each class.
- Upper bound on $\Delta(y, y')$.

$$\hat{y} \arg\max_{y' \in} \left\langle w, \Psi(x, y') \right\rangle \tag{17}$$

$$\implies \langle w, \Psi(x, y) \rangle \leqslant \langle w, \Psi(x, \hat{y}) \rangle \tag{18}$$

$$\tag{19}$$

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = y \neq y' \tag{16}$$

- In general, can also have different cost for each class.
- Upper bound on $\Delta(y, y')$.

$$\hat{y} \arg\max_{y' \in} \langle w, \Psi(x, y') \rangle \tag{17}$$

$$\implies \langle w, \Psi(x, y) \rangle \leqslant \langle w, \Psi(x, \hat{y}) \rangle \tag{18}$$

$$\implies \Delta(y, \hat{y}) \leqslant \Delta(y, \hat{y}) - \langle w, \Psi(x, y) - \Psi(x, \hat{y}) \rangle \qquad \text{When are they equal?} \tag{19}$$

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = y \neq y' \tag{16}$$

- In general, can also have different cost for each class.
- Upper bound on $\Delta(y, y')$.

$$\begin{aligned}
& \hat{y} \operatorname*{arg\,max}_{y' \in} \left\langle w, \Psi(x, y') \right\rangle && (17)\\
\implies & \langle w, \Psi(x, y)\rangle \leqslant \langle w, \Psi(x, \hat{y})\rangle && (18)\\
\implies & \Delta(y, \hat{y}) \leqslant \Delta(y, \hat{y}) - \langle w, \Psi(x, y) - \Psi(x, \hat{y})\rangle \qquad \text{When are they equal?} && (19)
\end{aligned}$$

- Generalized hinge loss:

$$\ell_{\text{hinge}}(y, x, w) \max_{y' \in} \Delta(y, y') - \left\langle w, \Psi(x, y) - \Psi(x, y') \right\rangle \tag{20}$$

Multiclass SVM with Hinge Loss

- Recall the hinge loss formulation for binary SVM (without the bias term):

$$\min_{w \in \mathrm{R}^d} \frac{1}{2}||w||^2 + C \sum_{n=1}^{N} \max\left(0, 1 - \underbrace{w^T}_{\text{margin}}\right).$$

Multiclass SVM with Hinge Loss

- Recall the hinge loss formulation for binary SVM (without the bias term):

$$\min_{w \in \mathrm{R}^d} \frac{1}{2}||w||^2 + C \sum_{n=1}^{N} \max\left(0, 1 - \underbrace{w^T}_{\text{margin}}\right).$$

- The multiclass objective:

$$\min_{w \in \mathrm{R}^d} \frac{1}{2}||w||^2 + C \sum_{n=1}^{N} \max_{y' \in} \Delta(y, y') - \underbrace{\left\langle w, \Psi(x, y) - \Psi(x, y') \right\rangle}_{\text{margin}}$$

  - $\Delta(y, y')$ as target margin for each class.
  - If margin $m_{n,y'}(w)$ meets or exceeds its target $\Delta(, y')$ $\forall y \in$, then no loss on example n.

Recap: What Have We Got?

- Problem: Multiclass classification $\mathcal{Y} = \{1, \ldots, k\}$

Recap: What Have We Got?

- Problem: Multiclass classification $\mathcal{Y} = \{1, \ldots, k\}$
- Solution 1: One-vs-All
  - Train k models: $h_1(x), \ldots, h_k(x) : \mathcal{X} \to \mathrm{R}$.
  - Predict with $\arg\max_{y \in \mathcal{Y}} h_y(x)$.
  - Gave simple example where this fails for linear classifiers

Recap: What Have We Got?

- Problem: Multiclass classification $\mathcal{Y} = \{1, \ldots, k\}$
- Solution 1: One-vs-All
  - Train k models: $h_1(x), \ldots, h_k(x) : \mathcal{X} \to \mathrm{R}$.
  - Predict with $\arg\max_{y \in \mathcal{Y}} h_y(x)$.
  - Gave simple example where this fails for linear classifiers
- Solution 2: Multiclass loss
  - Train one model: $h(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathrm{R}$.
  - Prediction involves solving $\arg\max_{y \in \mathcal{Y}} h(x, y)$.

Does it work better in practice?

- Paper by Rifkin & Klautau: "In Defense of One-Vs-All Classification" (2004)
  - Extensive experiments, carefully done
    - albeit on relatively small UCI datasets
  - Suggests one-vs-all works just as well in practice
    - (or at least, the advantages claimed by earlier papers for multiclass methods were not compelling)

Does it work better in practice?

- Paper by Rifkin & Klautau: "In Defense of One-Vs-All Classification" (2004)
  - Extensive experiments, carefully done
    - albeit on relatively small UCI datasets
  - Suggests one-vs-all works just as well in practice
    - (or at least, the advantages claimed by earlier papers for multiclass methods were not compelling)
- Compared
  - many multiclass frameworks (including the one we discuss)
  - one-vs-all for SVMs with RBF kernel
  - one-vs-all for square loss with RBF kernel (for classification!)
- All performed roughly the same

Why Are We Bothering with Multiclass?

- The framework we have developed for multiclass
  - compatibility features / scoring functions
  - multiclass margin
  - target margin / multiclass loss

Why Are We Bothering with Multiclass?

- The framework we have developed for multiclass
  - compatibility features / scoring functions
  - multiclass margin
  - target margin / multiclass loss
- Generalizes to situations where k is very large and one-vs-all is intractable.

Why Are We Bothering with Multiclass?

- The framework we have developed for multiclass
  - compatibility features / scoring functions
  - multiclass margin
  - target margin / multiclass loss
- Generalizes to situations where k is very large and one-vs-all is intractable.
- Key idea is that we can generalize across outputs y by using features of y.

Example: Part-of-speech (POS) Tagging

- Given a sentence, give a part of speech tag for each word:

x	$\underbrace{\text{[START]}}_{x_0}$	$\underbrace{\text{He}}_{x_1}$	$\underbrace{\text{eats}}_{x_2}$	$\underbrace{\text{apples}}_{x_3}$
y	$\underbrace{\text{[START]}}_{y_0}$	$\underbrace{\text{Pronoun}}_{y_1}$	$\underbrace{\text{Verb}}_{y_2}$	$\underbrace{\text{Noun}}_{y_3}$

Example: Part-of-speech (POS) Tagging

- Given a sentence, give a part of speech tag for each word:

x	$\underbrace{\text{[START]}}_{x_0}$	$\underbrace{\text{He}}_{x_1}$	$\underbrace{\text{eats}}_{x_2}$	$\underbrace{\text{apples}}_{x_3}$
y	$\underbrace{\text{[START]}}_{y_0}$	$\underbrace{\text{Pronoun}}_{y_1}$	$\underbrace{\text{Verb}}_{y_2}$	$\underbrace{\text{Noun}}_{y_3}$

- $\mathcal{V} = \{\text{all English words}\} \cup \{\text{[START]}, \text{"."}\}$
- $\mathcal{X} = \mathcal{V}^n$, $n = 1, 2, 3, \ldots$ [Word sequences of any length]

Example: Part-of-speech (POS) Tagging

- Given a sentence, give a part of speech tag for each word:

x	$\underbrace{\text{[START]}}_{x_0}$	$\underbrace{\text{He}}_{x_1}$	$\underbrace{\text{eats}}_{x_2}$	$\underbrace{\text{apples}}_{x_3}$
y	$\underbrace{\text{[START]}}_{y_0}$	$\underbrace{\text{Pronoun}}_{y_1}$	$\underbrace{\text{Verb}}_{y_2}$	$\underbrace{\text{Noun}}_{y_3}$

- $\mathcal{V} = \{\text{all English words}\} \cup \{\text{[START]}, ".\text{"}\}$
- $\mathcal{X} = \mathcal{V}^n$, $n = 1, 2, 3, \ldots$ [Word sequences of any length]
- $\mathcal{P} = \{\text{START}, \text{Pronoun}, \text{Verb}, \text{Noun}, \text{Adjective}\}$
- $\mathcal{Y} = \mathcal{P}^n$, $n = 1, 2, 3, \ldots$[Part of speech sequence of any length]

Multiclass Hypothesis Space

- Discrete output space: $\mathcal{Y}(x)$
  - Very large but has structure, e.g., linear chain (sequence labeling), tree (parsing)
  - Size depends on input x

Multiclass Hypothesis Space

- Discrete output space: $\mathcal{Y}(x)$
  - Very large but has structure, e.g., linear chain (sequence labeling), tree (parsing)
  - Size depends on input x
- Base Hypothesis Space: $\mathcal{H} = \{h : \mathcal{X} \times \mathcal{Y} \to \mathrm{R}\}$
  - $h(x, y)$ gives compatibility score between input x and output y

Multiclass Hypothesis Space

- Discrete output space: $\mathcal{Y}(x)$
  - Very large but has structure, e.g., linear chain (sequence labeling), tree (parsing)
  - Size depends on input x
- Base Hypothesis Space: $\mathcal{H} = \{h : \mathcal{X} \times \mathcal{Y} \to \mathrm{R}\}$
  - $h(x, y)$ gives compatibility score between input x and output y
- Multiclass hypothesis space

$$\mathcal{F} = \left\{ x \mapsto \underset{y \in \mathcal{Y}}{\arg\max}\, h(x, y) \mid h \in \mathcal{H} \right\}$$

- Final prediction function is an $f \in \mathcal{F}$.
- For each $f \in \mathcal{F}$ there is an underlying compatibility score function $h \in \mathcal{H}$.

Structured Prediction

- Part-of-speech tagging

x:	he	eats	apples
y:	pronoun	verb	noun

- Multiclass hypothesis space:

$$h(x,y) = w^T \Psi(x,y) \tag{21}$$

$$\mathcal{F} = \left\{ x \mapsto \arg\max_{y \in \mathcal{Y}} h(x,y) \mid h \in \mathcal{H} \right\} \tag{22}$$

- A special case of multiclass classification
- How to design the feature map Ψ? What are the considerations?

Unary features

- A **unary feature** only depends on
  - the label at a single position, y_i, and x
- Example:

$$
\begin{aligned}
\phi_1(x, y_i) &= \mathbb{1}[x_i = \text{runs}]\mathbb{1}[y_i = \text{Verb}] \\
\phi_2(x, y_i) &= \mathbb{1}[x_i = \text{runs}]\mathbb{1}[y_i = \text{Noun}] \\
\phi_3(x, y_i) &= \mathbb{1}[x_{i-1} = \text{He}]\mathbb{1}[x_i = \text{runs}]\mathbb{1}[y_i = \text{Verb}]
\end{aligned}
$$

Markov features

- A **markov feature** only depends on
  - two adjacent labels, y_{i-1} and y_i, and x
- Example:

$$\begin{aligned}\theta_1(x, y_{i-1}, y_i) &= \mathbb{1}[y_{i-1} = \text{Pronoun}]\mathbb{1}[y_i = \text{Verb}] \\ \theta_2(x, y_{i-1}, y_i) &= \mathbb{1}[y_{i-1} = \text{Pronoun}]\mathbb{1}[y_i = \text{Noun}]\end{aligned}$$

- Reminiscent of Markov models in the output space
- Possible to have higher-order features

Local Feature Vector and Compatibility Score

- At each position i in sequence, define the **local feature vector** (unary and markov):

$$\begin{aligned}\Psi_i(x, y_{i-1}, y_i) &= (\phi_1(x, y_i), \phi_2(x, y_i), \ldots, \\ &\quad \theta_1(x, y_{i-1}, y_i), \theta_2(x, y_{i-1}, y_i), \ldots)\end{aligned}$$

Local Feature Vector and Compatibility Score

- At each position i in sequence, define the **local feature vector** (unary and markov):

$$\begin{aligned}\Psi_i(x, y_{i-1}, y_i) &= (\phi_1(x, y_i), \phi_2(x, y_i), \ldots, \\ &\quad \theta_1(x, y_{i-1}, y_i), \theta_2(x, y_{i-1}, y_i), \ldots)\end{aligned}$$

- And **local compatibility score** at position i: $\langle w, \Psi_i(x, y_{i-1}, y_i)\rangle$.

Local Feature Vector and Compatibility Score

- At each position i in sequence, define the **local feature vector** (unary and markov):

$$\begin{aligned}\Psi_i(x, y_{i-1}, y_i) &= (\phi_1(x, y_i), \phi_2(x, y_i), \ldots, \\ &\quad \theta_1(x, y_{i-1}, y_i), \theta_2(x, y_{i-1}, y_i), \ldots)\end{aligned}$$

- And **local compatibility score** at position i: $\langle w, \Psi_i(x, y_{i-1}, y_i)\rangle$.
- The compatibility score for (x, y) is the sum of local compatibility scores:

$$\sum_i \langle w, \Psi_i(x, y_{i-1}, y_i)\rangle = \left\langle w, \sum_i \Psi_i(x, y_{i-1}, y_i)\right\rangle = \langle w, \Psi(x, y)\rangle, \tag{23}$$

where we define the **sequence feature vector** by

$$\Psi(x, y) = \sum_i \Psi_i(x, y_{i-1}, y_i). \qquad \text{decomposable}$$

Structured perceptron

Given a dataset $= (x, y)$;
Initialize $w \leftarrow 0$;
for $iter = 1, 2, \ldots, T$ **do**
  for $(x, y) \in$ **do**
    $\hat{y} = \arg\max_{y' \in (x)} w^T \psi(x, y')$;
    if $\hat{y} \neq y$ **then** `// We've made a mistake`
      $w \leftarrow w + \Psi(x, y)$; `// Move the scorer towards` $\psi(x, y)$
      $w \leftarrow w - \Psi(x, \hat{y})$; `// Move the scorer away from` $\psi(x, \hat{y})$
    end
  end
end

Structured perceptron

Given a dataset $= (x, y)$;
Initialize $w \leftarrow 0$;
for $iter = 1, 2, \ldots, T$ **do**
  for $(x, y) \in$ **do**
    $\hat{y} = \arg\max_{y' \in (x)} w^T \psi(x, y')$;
    if $\hat{y} \neq y$ **then** `// We've made a mistake`
      $w \leftarrow w + \Psi(x, y)$; `// Move the scorer towards` $\psi(x, y)$
      $w \leftarrow w - \Psi(x, \hat{y})$; `// Move the scorer away from` $\psi(x, \hat{y})$
    end
  end
end

Identical to the multiclass perceptron algorithm except the $\arg\max$ is now over the structured output space (x).

Structured hinge loss

- Recall the generalized hinge loss

$$\ell_{\text{hinge}}(y, \hat{y}) \max_{y' \in (x)} \Delta(y, y') + \left\langle w, \Psi(x, y') - \Psi(x, y) \right\rangle \tag{24}$$

- What is $\Delta(y, y')$ for two sequences?

Structured hinge loss

- Recall the generalized hinge loss

$$\ell_{\text{hinge}}(y, \hat{y}) \max_{y' \in (x)} \Delta(y, y') + \left\langle w, \Psi(x, y') - \Psi(x, y) \right\rangle \tag{24}$$

- What is $\Delta(y, y')$ for two sequences?
- **Hamming loss** is common:

$$\Delta(y, y') = \frac{1}{L} \sum_{i=1}^{L} \mathbb{1}[y_i \neq y_i']$$

where L is the sequence length.

Structured SVM

Exercise:

- Write down the objective of structured SVM using the structured hinge loss.
- Stochastic sub-gradient descent for structured SVM (similar to HW3 P3)
- Compare with the structured perceptron algorithm

The argmax problem for sequences

Problem To compute predictions, we need to find $\arg\max_{y \in \mathcal{Y}(x)} \langle w, \Psi(x, y) \rangle$, and $|\mathcal{Y}(x)|$ is exponentially large.

Figure by Daumé III. A course in machine learning. Figure 17.1.

The argmax problem for sequences

Problem To compute predictions, we need to find $\arg\max_{y \in \mathcal{Y}(x)} \langle w, \Psi(x, y) \rangle$, and $|\mathcal{Y}(x)|$ is exponentially large.

Observation $\Psi(x, y)$ decomposes to $\sum_i \Psi_i(x, y)$.

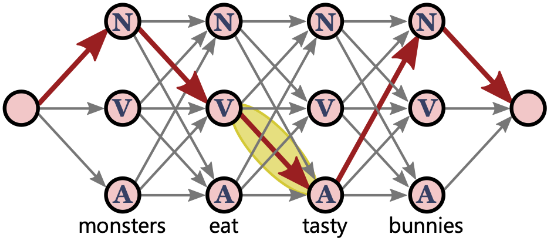
Figure by Daumé III. A course in machine learning. Figure 17.1.

The argmax problem for sequences

Problem To compute predictions, we need to find $\arg\max_{y \in \mathcal{Y}(x)} \langle w, \Psi(x,y) \rangle$, and $|\mathcal{Y}(x)|$ is exponentially large.

Observation $\Psi(x,y)$ decomposes to $\sum_i \Psi_i(x,y)$.

Solution Dynamic programming (similar to the Viterbi algorithm)



What's the running time?

Figure by Daumé III. A course in machine learning. Figure 17.1.

Conditional random field (CRF)

- Recall that we can write logistic regression in a general form:

$$p(y|x) = \frac{1}{Z(x)} \exp(w^\top \psi(x, y)).$$

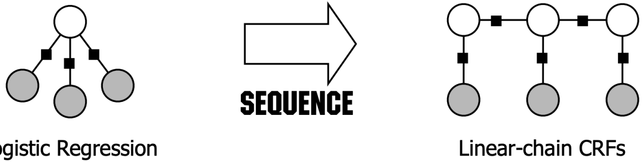
- Z is normalization constant: $Z(x) = \sum_{y \in Y} \exp(w^\top \psi(x, y))$.

Conditional random field (CRF)

- Recall that we can write logistic regression in a general form:

$$p(y|x) = \frac{1}{Z(x)} \exp(w^\top \psi(x, y)).$$

- Z is normalization constant: $Z(x) = \sum_{y \in Y} \exp(w^\top \psi(x, y))$.
- Example: linear chain $\{y_t\}$
- We can incorporate unary and Markov features: $p(y|x) = \frac{1}{Z(x)} \exp(\sum_t w^\top \psi(x, y_t, y_{t-1}))$



Conditional random field (CRF)

- Compared to Structured SVM, CRF has a probabilistic interpretation.
- We can draw samples in the output space.

Conditional random field (CRF)

- Compared to Structured SVM, CRF has a probabilistic interpretation.
- We can draw samples in the output space.
- How do we learn w? Maximum log likelihood, and regularization term: $\lambda \|w\|^2$
- Loss function:

$$
\begin{aligned}
l(w) &= -\frac{1}{N} \sum_{i=1}^{N} \log p(y^{(i)} | x^{(i)}) + \frac{1}{2} \lambda \|w\|^2 \\
&= -\frac{1}{N} \sum_{i} \sum_{t} \sum_{k} w_k \psi_k (y_t^{(i)}, y_{t-1}^{(i)}) + \frac{1}{N} \sum_{i} \log Z(x^{(i)}) + \frac{1}{2} \sum_{k} \lambda w_k^2
\end{aligned}
$$

Conditional random field (CRF)

- Loss function:

$$l(w) = -\frac{1}{N}\sum_i \sum_t \sum_k w_k \psi_k(x^{(i)}, y_t^{(i)}, y_{t-1}^{(i)}) + \frac{1}{N}\sum_i \log Z(x^{(i)}) + \frac{1}{2}\sum_k \lambda w_k^2$$

- Gradient:

$$\frac{\partial l(w)}{\partial w_k} = -\frac{1}{N}\sum_i \sum_t \sum_k \psi_k(x^{(i)}, y_t^{(i)}, y_{t-1}^{(i)}) \tag{25}$$

$$+\frac{1}{N}\sum_i \frac{\partial}{\partial w_k} \log \sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'}\psi_{k'}(x^{(i)}, y'_t, y'_{t-1})) + \sum_k \lambda w_k \tag{26}$$

Conditional random field (CRF)

- What is $\frac{1}{N}\sum_i \sum_t \sum_k \psi_k(x^{(i)}, y_t^{(i)}, y_{t-1}^{(i)})$?

Conditional random field (CRF)

- What is $\frac{1}{N}\sum_i \sum_t \sum_k \psi_k(x^{(i)}, y_t^{(i)}, y_{t-1}^{(i)})$?
- It is the expectation $\psi_k(x^{(i)}, y_t, y_{t-1})$ under the empirical distribution $\tilde{p}(x, y) = \frac{1}{N}\sum_i \mathbb{1}[x = x^{(i)}]\mathbb{1}[y = y^{(i)}]$.

Conditional random field (CRF)

- What is $\frac{1}{N}\sum_i \frac{\partial}{\partial w_k} \log \sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y'_t, y'_{t-1}))$?

Conditional random field (CRF)

- What is $\frac{1}{N}\sum_i \frac{\partial}{\partial w_k} \log \sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y'_t, y'_{t-1}))$?

$$\frac{1}{N}\sum_i \frac{\partial}{\partial w_k} \log \sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y'_t, y'_{t-1})) \tag{27}$$

$$= \frac{1}{N}\sum_i \left[\sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y'_t, y'_{t-1}))\right]^{-1} \tag{28}$$

$$\left[\sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y^{(i)}_t, y^{(i)}_{t-1})) \sum_t \psi_k(x^{(i)}, y'_t, y'_{t-1})\right] \tag{29}$$

$$= \frac{1}{N}\sum_i \sum_t \sum_{y' \in Y} p(y'_t, y'_{t-1}|x) \psi_k(x^{(i)}, y'_t, y'_{t-1}) \tag{30}$$

Conditional random field (CRF)

- What is $\frac{1}{N}\sum_i \frac{\partial}{\partial w_k} \log \sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'}\psi_{k'}(x^{(i)}, y'_t, y'_{t-1}))$?

$$\frac{1}{N}\sum_i \frac{\partial}{\partial w_k} \log \sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'}\psi_{k'}(x^{(i)}, y'_t, y'_{t-1})) \tag{27}$$

$$= \frac{1}{N}\sum_i \left[\sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'}\psi_{k'}(x^{(i)}, y'_t, y'_{t-1}))\right]^{-1} \tag{28}$$

$$\left[\sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'}\psi_{k'}(x^{(i)}, y^{(i)}_t, y^{(i)}_{t-1})) \sum_t \psi_k(x^{(i)}, y'_t, y'_{t-1})\right] \tag{29}$$

$$= \frac{1}{N}\sum_i \sum_t \sum_{y' \in Y} p(y'_t, y'_{t-1}|x)\psi_k(x^{(i)}, y'_t, y'_{t-1}) \tag{30}$$

- It is the expectation of $\psi_k(x^{(i)}, y'_t, y'_{t-1})$ under the model distribution $p(y'_t, y'_{t-1}|x)$.

Conditional random field (CRF)

- To compute the gradient, we need to infer expectation under the model distribution $p(y|x)$.

Conditional random field (CRF)

- To compute the gradient, we need to infer expectation under the model distribution $p(y|x)$.
- Compare the learning algorithms: in structured SVM we need to compute the argmax, whereas in CRF we need to compute the model expectation.

Conditional random field (CRF)

- To compute the gradient, we need to infer expectation under the model distribution $p(y|x)$.
- Compare the learning algorithms: in structured SVM we need to compute the argmax, whereas in CRF we need to compute the model expectation.
- Both problems are NP-hard for general graphs.

CRF Inference

- In the linear chain structure, we can use the forward-backward algorithm for inference, similar to Viterbi.

CRF Inference

- In the linear chain structure, we can use the forward-backward algorithm for inference, similar to Viterbi.
- Initiate $\alpha_j(1) = \exp(w^\top \psi(y_1 = j, x_1))$

CRF Inference

- In the linear chain structure, we can use the forward-backward algorithm for inference, similar to Viterbi.
- Initiate $\alpha_j(1) = \exp(w^\top \psi(y_1 = j, x_1))$
- Recursion: $\alpha_j(t) = \sum_i \alpha_i(t-1) \exp(w^\top \psi(y_t = j, y_{t-1} = i, x_t))$

- In the linear chain structure, we can use the forward-backward algorithm for inference, similar to Viterbi.
- Initiate $\alpha_j(1) = \exp(w^\top \psi(y_1 = j, x_1))$
- Recursion: $\alpha_j(t) = \sum_i \alpha_i(t-1) \exp(w^\top \psi(y_t = j, y_{t-1} = i, x_t))$
- Result: $Z(x) = \sum_j \alpha_j(T)$

CRF Inference

- In the linear chain structure, we can use the forward-backward algorithm for inference, similar to Viterbi.
- Initiate $\alpha_j(1) = \exp(w^\top \psi(y_1 = j, x_1))$
- Recursion: $\alpha_j(t) = \sum_i \alpha_i(t-1) \exp(w^\top \psi(y_t = j, y_{t-1} = i, x_t))$
- Result: $Z(x) = \sum_j \alpha_j(T)$
- Similar for the backward direction.

CRF Inference

- In the linear chain structure, we can use the forward-backward algorithm for inference, similar to Viterbi.
- Initiate $\alpha_j(1) = \exp(w^\top \psi(y_1 = j, x_1))$
- Recursion: $\alpha_j(t) = \sum_i \alpha_i(t-1) \exp(w^\top \psi(y_t = j, y_{t-1} = i, x_t))$
- Result: $Z(x) = \sum_j \alpha_j(T)$
- Similar for the backward direction.
- Test time, again use Viterbi algorithm to infer argmax.

CRF Inference

- In the linear chain structure, we can use the forward-backward algorithm for inference, similar to Viterbi.
- Initiate $\alpha_j(1) = \exp(w^\top \psi(y_1 = j, x_1))$
- Recursion: $\alpha_j(t) = \sum_i \alpha_i(t-1) \exp(w^\top \psi(y_t = j, y_{t-1} = i, x_t))$
- Result: $Z(x) = \sum_j \alpha_j(T)$
- Similar for the backward direction.
- Test time, again use Viterbi algorithm to infer argmax.
- The inference algorithm can be generalized to belief propagation (BP) in a tree structure (exact inference).

CRF Inference

- In the linear chain structure, we can use the forward-backward algorithm for inference, similar to Viterbi.
- Initiate $\alpha_j(1) = \exp(w^\top \psi(y_1 = j, x_1))$
- Recursion: $\alpha_j(t) = \sum_i \alpha_i(t-1) \exp(w^\top \psi(y_t = j, y_{t-1} = i, x_t))$
- Result: $Z(x) = \sum_j \alpha_j(T)$
- Similar for the backward direction.
- Test time, again use Viterbi algorithm to infer argmax.
- The inference algorithm can be generalized to belief propagation (BP) in a tree structure (exact inference).
- In general graphs, we rely on approximate inference (e.g. loopy belief propagation).

Examples

- POS tag Relationship between constituents, e.g. NP is likely to be followed by a VP.

Examples

- POS tag Relationship between constituents, e.g. NP is likely to be followed by a VP.
- Semantic segmentation
  Relationship between pixels, e.g. a grass pixel is likely to be next to another grass pixel, and a sky pixel is likely to be above a grass pixel.

Examples

- POS tag Relationship between constituents, e.g. NP is likely to be followed by a VP.
- Semantic segmentation
  Relationship between pixels, e.g. a grass pixel is likely to be next to another grass pixel, and a sky pixel is likely to be above a grass pixel.
- Multi-label learning
  An image may contain multiple class labels, e.g. a bus is likely to co-occur with a car.

Conclusion

Multiclass algorithms

- Reduce to binary classification, OvA, AvA
  - Good enough for simple multiclass problems
  - They don't scale and have simplified assumptions

Conclusion

Multiclass algorithms

- Reduce to binary classification, OvA, AvA
    - Good enough for simple multiclass problems
    - They don't scale and have simplified assumptions
- Generalize binary classification algorithms using multiclass loss
    - Multi-class perceptron, multi-class logistics regression, multi-class SVM

Conclusion

Multiclass algorithms

- Reduce to binary classification, OvA, AvA
  - Good enough for simple multiclass problems
  - They don't scale and have simplified assumptions
- Generalize binary classification algorithms using multiclass loss
  - Multi-class perceptron, multi-class logistics regression, multi-class SVM
- Structured prediction: Structured SVM, CRF. Data containing structure. Extremely large output space. Text and image applications.