Linear models for regression and classification

Dr. Alejandro Veloz

Supervised learning

Problem formulation

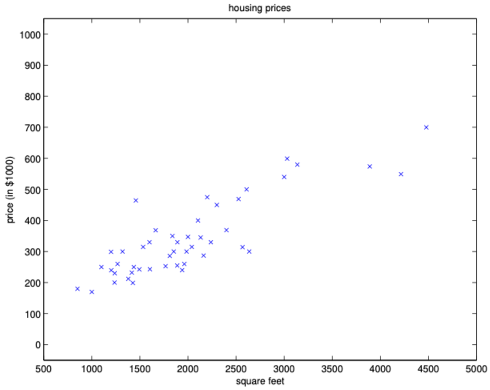
A *hypothesis* h is employed as a model for solving the problem, in that it maps inputs x to outputs y,

$$x \rightarrow \boxed{h} \rightarrow y$$

Supervised learning

Suppose we have a dataset giving the living areas and prices of houses:

Living area (feet ^{2})	Price $(1000\$$ s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮



Supervised learning - notation

- $x^{(i)}$ to denote the "input" variables (living area in this example), also called **input features.**

Supervised learning - notation

- $x^{(i)}$ to denote the “input” variables (living area in this example), also called **input features.**
- $y^{(i)}$ to denote the “output” or **target variable** that we are trying to predict (price).

Supervised learning - notation

- $x^{(i)}$ to denote the “input” variables (living area in this example), also called **input features.**
- $y^{(i)}$ to denote the “output” or **target variable** that we are trying to predict (price).
- A pair $\left(x^{(i)}, y^{(i)}\right)$ is called a **training example.**

Supervised learning - notation

- $x^{(i)}$ to denote the “input” variables (living area in this example), also called **input features.**
- $y^{(i)}$ to denote the “output” or **target variable** that we are trying to predict (price).
- A pair $\left(x^{(i)}, y^{(i)}\right)$ is called a **training example.**
- The **training set** is the list of n training examples

$$\left\{\left(x^{(i)}, y^{(i)}\right); i = 1, \ldots, n\right\}.$$

Supervised learning - notation

- $x^{(i)}$ to denote the "input" variables (living area in this example), also called **input features.**
- $y^{(i)}$ to denote the "output" or **target variable** that we are trying to predict (price).
- A pair $\left(x^{(i)}, y^{(i)}\right)$ is called a **training example.**
- The **training set** is the list of n training examples

$$\left\{\left(x^{(i)}, y^{(i)}\right); i = 1, \ldots, n\right\}.$$

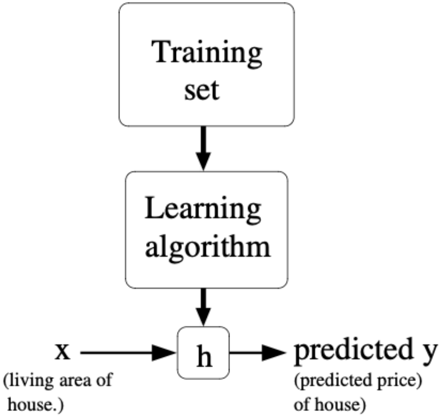
- We will also use $\mathcal{X}$ to denote the space of input values, and $\mathcal{Y}$ the space of output values. In this example, $\mathcal{X} = \mathcal{Y} = \mathbb{R}$.

Supervised learning

To describe the supervised learning problem slightly more formally, our goal is:

Given a training set, to learn a function $h : \mathcal{X} \mapsto \mathcal{Y}$ so that $h(x)$ is a "good" predictor for the corresponding value of y.

- This function h is called **hypothesis**.
- **Regression problem.** The target variable is continuous.
- **Classification problem.** The target variable can take on only a small number of discrete values.



Linear Models for Regression

Regression Basics

Regression Goal

Predict the value of *one or more* continuous target variables y given a D-dimensional vector x of input variables.

Linear Regression

Living area (feet^2)	#bedrooms	Price $(1000\$\text{s})$
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
$\vdots$	$\vdots$	$\vdots$

- We have features $\mathbf{x}^{(i)}$ in $\mathbb{R}^2$.
- We define $x_1^{(i)}$ as the living area of the i-th house in the training set, and $x_2^{(i)}$ is its number of bedrooms.

Linear Regression

- We must decide how to represent functions/hypotheses h.
- We decide to approximate y as a linear function of x:

$$h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

- θ_i 's are the parameters (also called weights) parameterizing the space of linear functions mapping from $\mathcal{X}$ to $\mathcal{Y}$.
- To simplify our notation, we also introduce the convention $x_0 = 1$ (this is the intercept term), such that:

$$h(\mathbf{x}) = \sum_{i=0}^{d} \theta_i x_i = \boldsymbol{\theta}^\top \mathbf{x}$$

Cost function

Given a training set, how do we learn the parameters θ?

Cost function

Given a training set, how do we learn the parameters θ?

- One reasonable method seems to be to make $h(\mathbf{x})$ close to y, at least for the training examples we have.

Cost function

Given a training set, how do we learn the parameters θ?

- One reasonable method seems to be to make $h(\mathbf{x})$ close to y, at least for the training examples we have.
- To formalize this, we will define a function that measures, for each value of the θ's, how close the $h\left(\mathbf{x}^{(i)}\right)$'s are to the corresponding $y^{(i)}$'s.

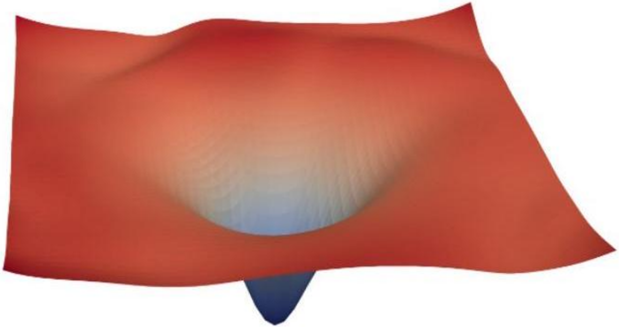
Cost function

Given a training set, how do we learn the parameters θ?

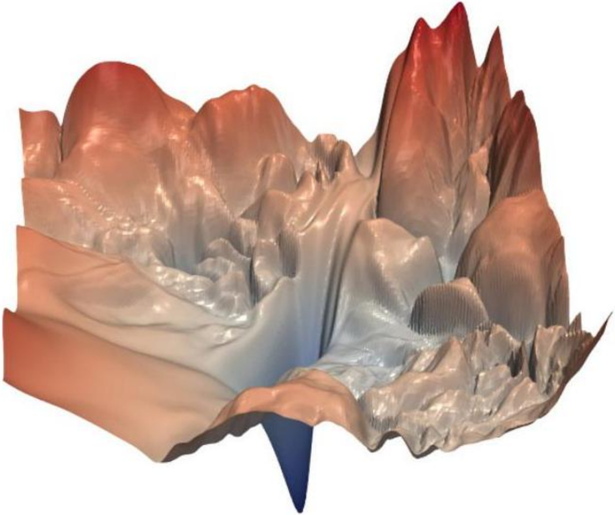
- One reasonable method seems to be to make $h(\mathbf{x})$ close to y, at least for the training examples we have.
- To formalize this, we will define a function that measures, for each value of the θ's, how close the $h\left(\mathbf{x}^{(i)}\right)$'s are to the corresponding $y^{(i)}$'s.
- We define the **cost function**:

$$J(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{n} \left(h_\theta\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2$$

Error surface



Error surface



Learning algorithm

Random-Regression

Require: Data $\mathcal{D}$, integer k

1: **for** $i = 1$ **to** k **do**

2: Randomly generate hypothesis $\boldsymbol{\theta}(i)$

3: **end for**

4: Let $i = \arg\min_j J(\boldsymbol{\theta}(j); \mathcal{D})$

5: **return** $\boldsymbol{\theta}(i)$

How do you think increasing the number of guesses k will change the training error of the resulting hypothesis?

Learning algorithm: sequential learning

- We want to choose $\boldsymbol{\theta}$ so as to minimize $J(\boldsymbol{\theta})$.

Learning algorithm: sequential learning

- We want to choose $\boldsymbol{\theta}$ so as to minimize $J(\boldsymbol{\theta})$.
- **Search algorithm (gradient descent):** start with some "initial guess" for $\boldsymbol{\theta}$, and repeatedly change $\boldsymbol{\theta}$ to make $J(\boldsymbol{\theta})$ smaller, until hopefully we converge to a value of $\boldsymbol{\theta}$ that minimizes $J(\boldsymbol{\theta})$.

Learning algorithm: sequential learning

- We want to choose $\boldsymbol{\theta}$ so as to minimize $J(\boldsymbol{\theta})$.
- **Search algorithm (gradient descent):** start with some "initial guess" for $\boldsymbol{\theta}$, and repeatedly change $\boldsymbol{\theta}$ to make $J(\boldsymbol{\theta})$ smaller, until hopefully we converge to a value of $\boldsymbol{\theta}$ that minimizes $J(\boldsymbol{\theta})$.
- The update step is:

$$\boldsymbol{\theta}^{(\tau+1)} = \boldsymbol{\theta}^{(\tau)} - \eta \nabla E_n$$

  where τ denotes the iteration number, and η is a learning rate parameter.
- The value of η needs to be chosen with care to ensure that the algorithm converges (Bishop and Nabney, 2008).

Learning algorithm: sequential learning

- For the model:

$$h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + ... + \theta_n x_n \ ,$$

  the update step is:

$$\theta_j := \theta_j - \eta \frac{\partial}{\partial \theta_j} J(\theta).$$

- This update is simultaneously performed for all values of $j = 0, \ldots, d.$
- η is called the learning rate.

Learning algorithm: sequential learning

- For the linear regression model, the update step can be:
  - **Batch gradient descent**

$$\theta_j := \theta_j + \eta \sum_{i=1}^{n} \left(y^{(i)} - h_\theta\left(\mathbf{x}^{(i)}\right)\right) x_j^{(i)}, (\text{ for every } j)$$

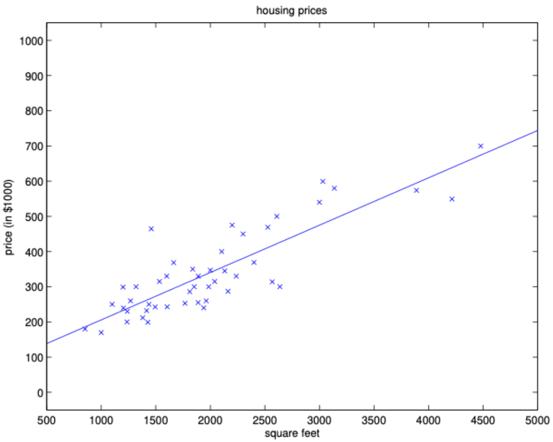
  - **Stochastic or incremental gradient descent**

$$\theta_j := \theta_j + \eta \left(y^{(i)} - h_\theta\left(\mathbf{x}^{(i)}\right)\right) x_j^{(i)}, \quad (\text{ for every } i, j)$$

or

$$\boldsymbol{\theta} := \boldsymbol{\theta} + \eta \left(y^{(i)} - h_\theta\left(\mathbf{x}^{(i)}\right)\right) \mathbf{x}^{(i)}, \quad (\text{ for every } i)$$

Regression curve



Probabilistic interpretation

Consider the model:

$$y^{(i)} = \boldsymbol{\theta}^\top \mathbf{x}^{(i)} + \epsilon^{(i)}$$

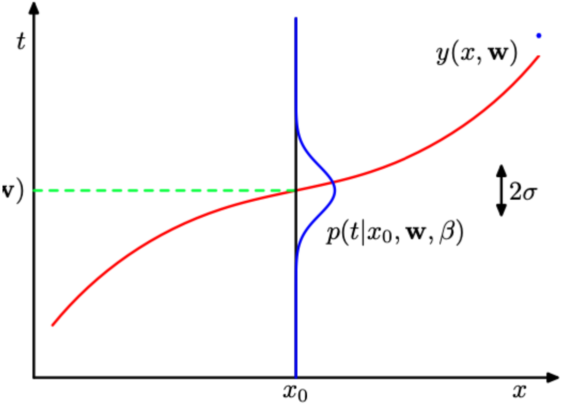
where $\epsilon^{(i)} \sim \mathcal{N}\left(0, \sigma^2\right)$. I.e., the density of $\epsilon^{(i)}$ is given by:

$$p\left(\epsilon^{(i)}\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(\epsilon^{(i)}\right)^2}{2\sigma^2}\right)$$

This implies that

$$p\left(y^{(i)} \mid \mathbf{x}^{(i)}; \theta\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y^{(i)} - \boldsymbol{\theta}^\top \mathbf{x}^{(i)}\right)^2}{2\sigma^2}\right)$$

Probabilistic interpretation



Probabilistic interpretation

The **Likelihood function** is given by:

$$L(\boldsymbol{\theta}) = L(\boldsymbol{\theta}; X, \mathbf{y}) = p(\mathbf{y} \mid X; \theta)$$

Probabilistic interpretation

The **Likelihood function** is given by:

$$L(\boldsymbol{\theta}) = L(\boldsymbol{\theta}; X, \mathbf{y}) = p(\mathbf{y} \mid X; \theta)$$

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} p\left(y^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}\right)$$

Probabilistic interpretation

The **Likelihood function** is given by:

$$L(\boldsymbol{\theta}) = L(\boldsymbol{\theta}; X, \mathbf{y}) = p(\mathbf{y} \mid X; \theta)$$

$$\begin{aligned} L(\boldsymbol{\theta}) &= \prod_{i=1}^{n} p\left(y^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}\right) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y^{(i)} - \boldsymbol{\theta}^\top \mathbf{x}^{(i)}\right)^2}{2\sigma^2}\right) \end{aligned}$$

Probabilistic interpretation

The **log Likelihood function** is given by:

$$
\begin{aligned}
\ell(\boldsymbol{\theta}) &= \log L(\boldsymbol{\theta}) \\
&= \log \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y^{(i)} - \boldsymbol{\theta}^\top \mathbf{x}^{(i)}\right)^2}{2\sigma^2}\right) \\
&= \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y^{(i)} - \boldsymbol{\theta}^\top \mathbf{x}^{(i)}\right)^2}{2\sigma^2}\right) \\
&= n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{n} \left(y^{(i)} - \boldsymbol{\theta}^\top \mathbf{x}^{(i)}\right)^2
\end{aligned}
$$

Probabilistic interpretation

The solution that minimizes $\ell(\boldsymbol{\theta})$ is:

$$\boldsymbol{\theta}_{\text{ML}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}$$

- Note that this is equivalent to obtain the parameters by minimizing:

$$\frac{1}{2} \sum_{i=1}^{n} \left(y^{(i)} - \boldsymbol{\theta}^\top \mathbf{x}^{(i)}\right)^2$$

- Note that we arrive to a solution regardless the value of σ^2. This fact is used to define an **exponential family and generalized linear models.**

Linear Basis Function Models

Linear Regression Models

Linear functions of the adjustable parameters:

$$y(\mathbf{x}, \boldsymbol{\theta}) = \theta_0 + \theta_1 x_1 + ... + \theta_D x_D$$

Linear Basis Function Models

Linear Regression Models

Linear functions of the adjustable parameters:

$$y(\mathbf{x}, \boldsymbol{\theta}) = \theta_0 + \theta_1 x_1 + ... + \theta_D x_D$$

Linear Basis Function Models

Can be extended to include nonlinear functions of the input variables using basis functions:

$$y(\mathbf{x}, \boldsymbol{\theta}) = \theta_0 + \sum_{j=1}^{M-1} \theta_j \phi_j(\mathbf{x}) = \sum_{j=0}^{M} \theta_j \phi_j(\mathbf{x}) = \boldsymbol{\theta}^\intercal \boldsymbol{\phi}(\mathbf{x})$$

Linear Basis Function Models

Linear Regression Models

Linear functions of the adjustable parameters:

$$y(\mathbf{x}, \boldsymbol{\theta}) = \theta_0 + \theta_1 x_1 + ... + \theta_D x_D$$

Linear Basis Function Models

Can be extended to include nonlinear functions of the input variables using basis functions:

$$y(\mathbf{x}, \boldsymbol{\theta}) = \theta_0 + \sum_{j=1}^{M-1} \theta_j \phi_j(\mathbf{x}) = \sum_{j=0}^{M} \theta_j \phi_j(\mathbf{x}) = \boldsymbol{\theta}^\intercal \boldsymbol{\phi}(\mathbf{x})$$

Basis Functions

Examples include polynomials, Gaussians, and sigmoidals.

Basis Functions

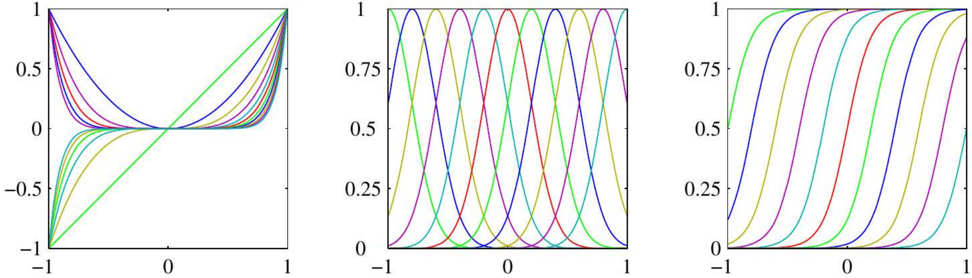
Examples of Basis Functions

$$\phi_j(x) = x^j \quad \text{(Polynomial)}$$

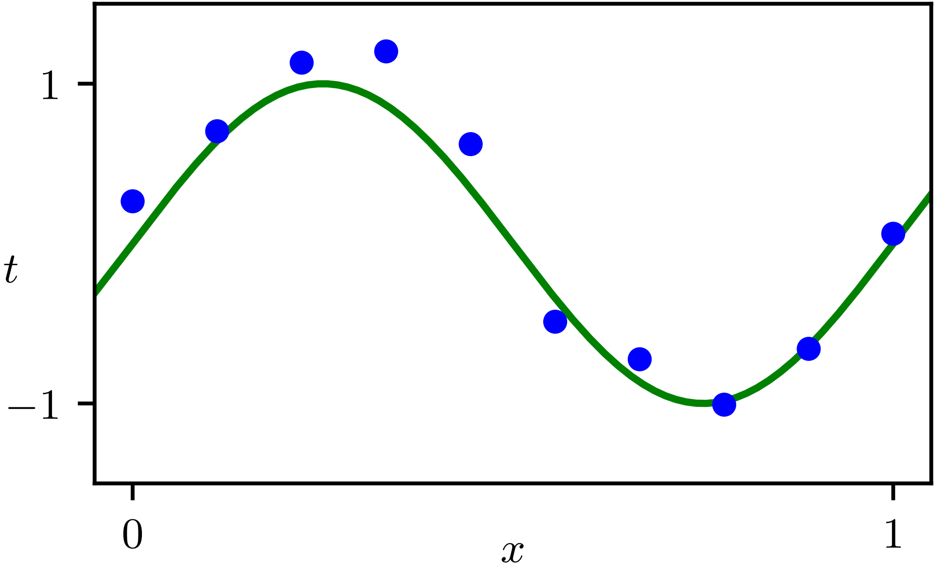
$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\} \quad \text{(Gaussian)}$$

$$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right) \quad \text{(Sigmoidal)}$$

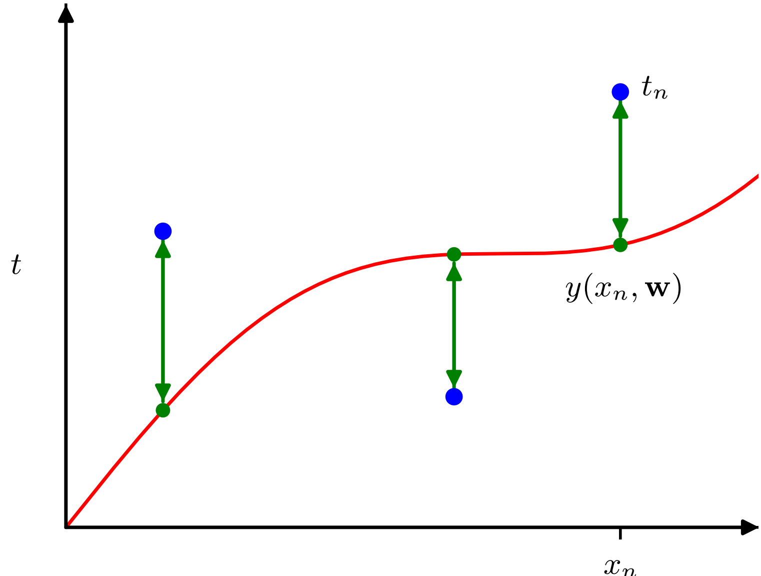
Basis Functions



Example polynomial regression (Bishop 2006)



Example polynomial regression (Bishop 2006)



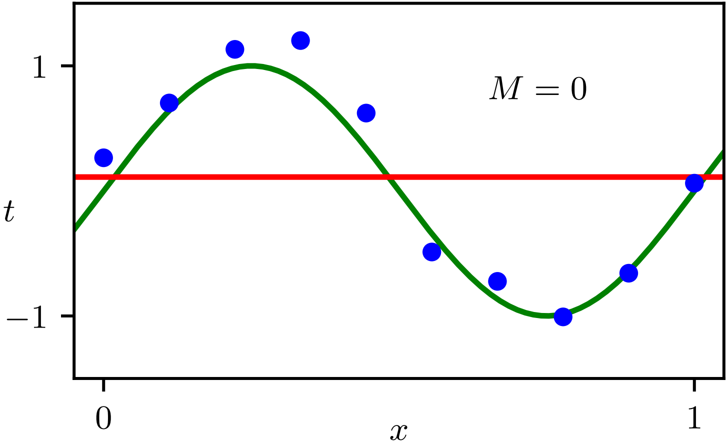
Example polynomial regression (Bishop 2006)

$$h(x, w) = w_0 + w_1 x + w_2 x^2 + ... + w_M x^M$$

$$E(w) = \sum_{n=1}^{N} \left(h\left(x_n, w\right) - t_n\right)^2$$

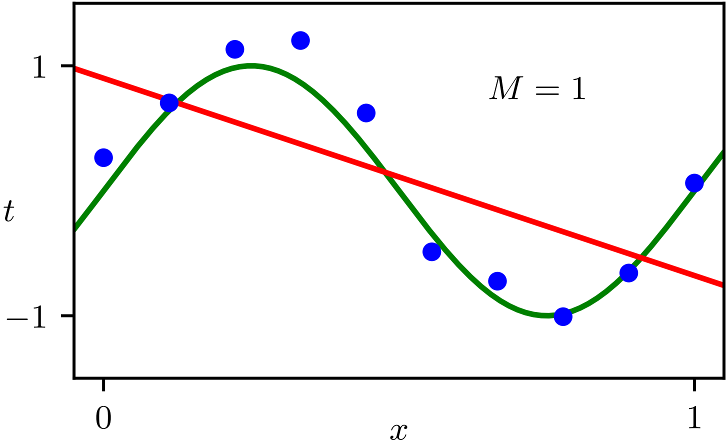
Example polynomial regression (Bishop 2006)

$$h(x, w) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$



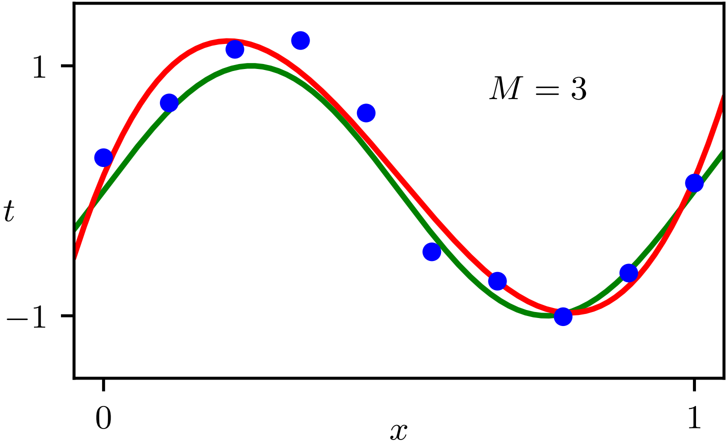
Example polynomial regression (Bishop 2006)

$$h(x, w) = w_0 + w_1 x + w_2 x^2 + ... + w_M x^M$$



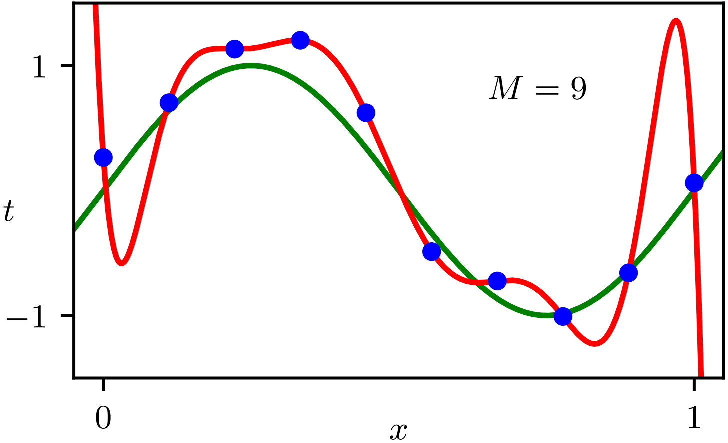
Example polynomial regression (Bishop 2006)

$$h(x, w) = w_0 + w_1 x + w_2 x^2 + ... + w_M x^M$$

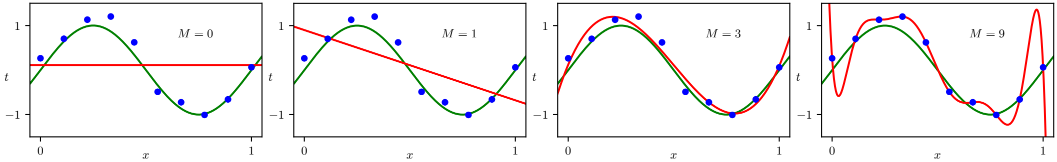


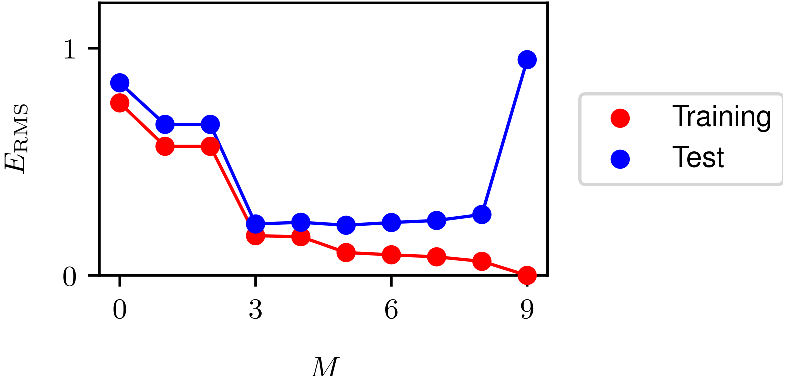
Example polynomial regression (Bishop 2006)

$$h(x, w) = w_0 + w_1 x + w_2 x^2 + ... + w_M x^M$$

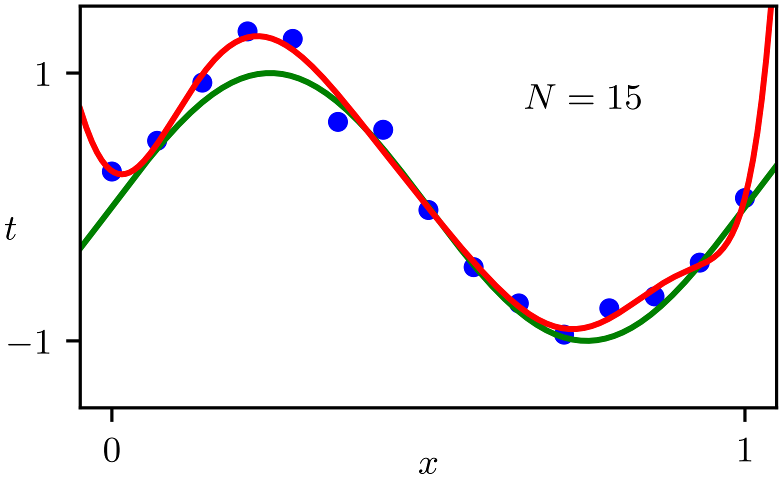


Example polynomial regression (Bishop 2006)

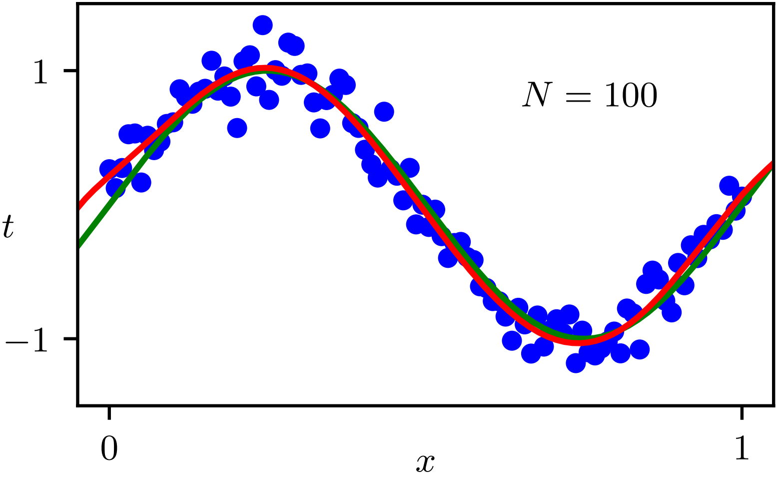




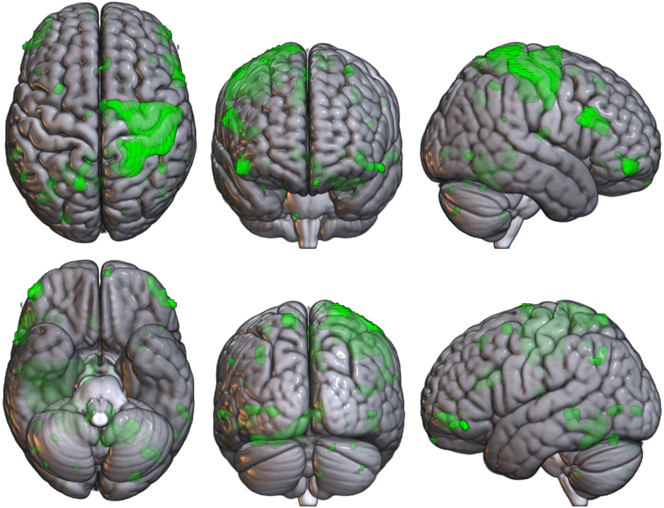
Example polynomial regression (Bishop 2006)



Example polynomial regression (Bishop 2006)



Example: task-related fMRI

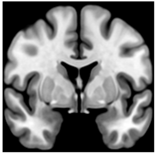


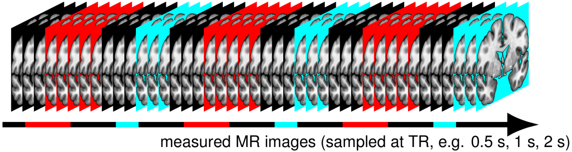
Example: task-related fMRI

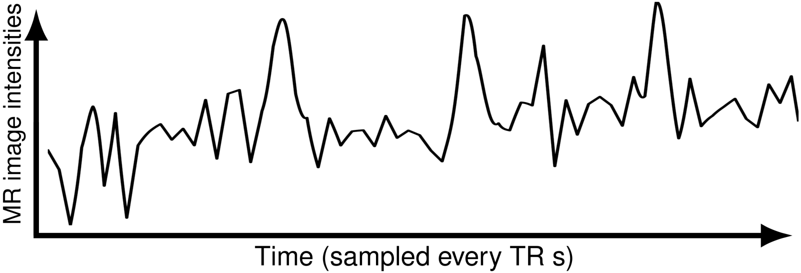
- It is a multiple regression model that quantitatively assesses whether fMRI signals exhibit BOLD-related fluctuations.
- On **one voxel**, the GLM can be expressed as:

$$\mathbf{y} = \beta_0 + \underbrace{\mathbf{x}_1\beta_1 + \cdots + \mathbf{x}_j\beta_j}_{\text{BOLD related signals}} + \underbrace{\mathbf{x}_{j+1}\beta_{j+1} + \cdots + \mathbf{x}_p\beta_p}_{\text{Nuisance signals}} + \boldsymbol{\epsilon} \qquad \beta_j \in \mathbb{R},\ \mathbf{y},\ \mathbf{x}_j\ \in \mathbb{R}^T$$

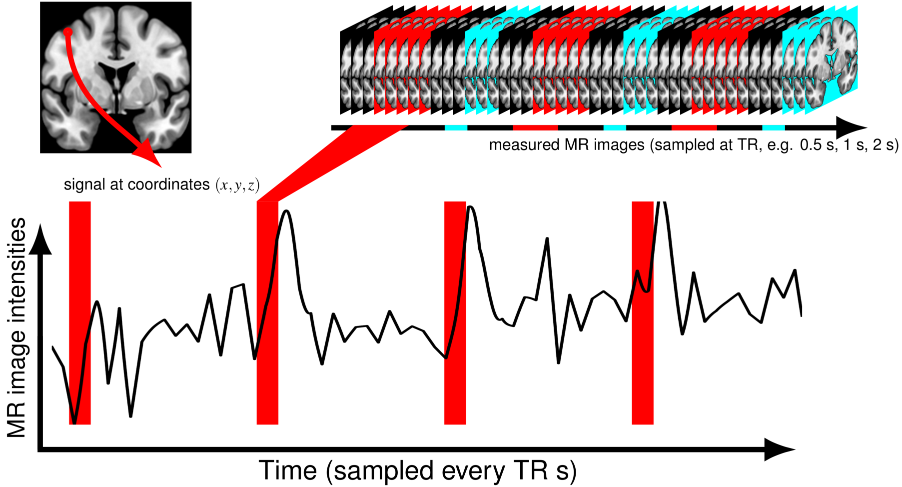
Example: task-related fMRI



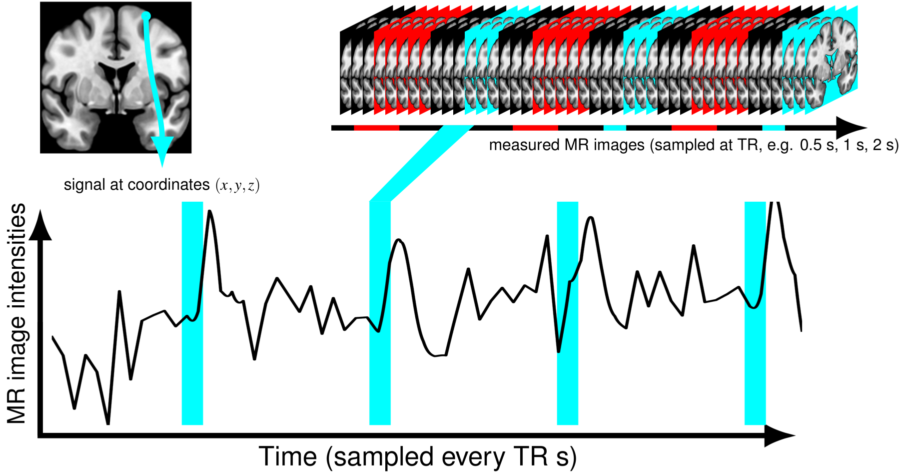




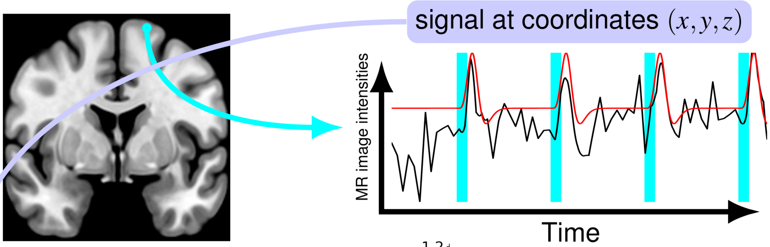
Example: task-related fMRI



Example: task-related fMRI

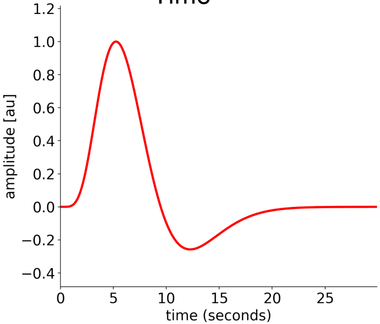


The BOLD response



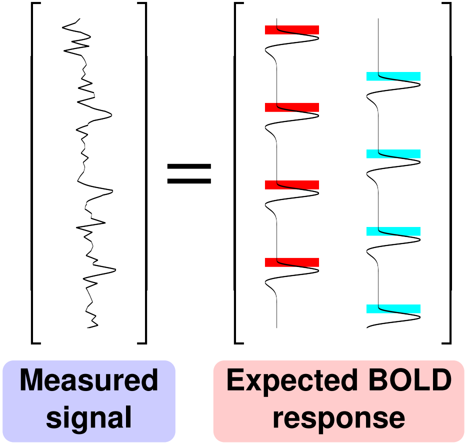
$$\mathbf{y} = \mathbf{h} * \mathbf{x} + \varepsilon$$

Hemodynamic Response Function



The General Linear Model

$$\mathbf{y} = \left[\, \mathbf{h} * \mathbf{x}_1 \quad \mathbf{h} * \mathbf{x}_2 \,\right] \theta + \mathbf{W}\eta + \epsilon$$



Regularized least squares

$$E_D(\boldsymbol{\theta}) + \lambda E_W(\boldsymbol{\theta})$$

where λ is the regularization coefficient that controls the relative importance of the data-dependent error $E_D(\boldsymbol{\theta})$ and the regularization term $E_W(\boldsymbol{\theta})$.

Regularized least squares

$$E_D(\boldsymbol{\theta}) + \lambda E_W(\boldsymbol{\theta})$$

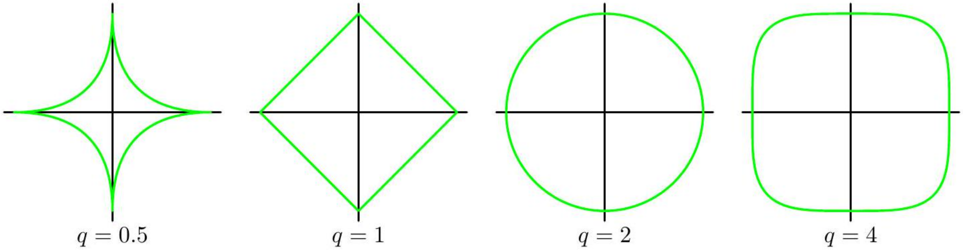
where λ is the regularization coefficient that controls the relative importance of the data-dependent error $E_D(\boldsymbol{\theta})$ and the regularization term $E_W(\boldsymbol{\theta})$.

Common regularizers:

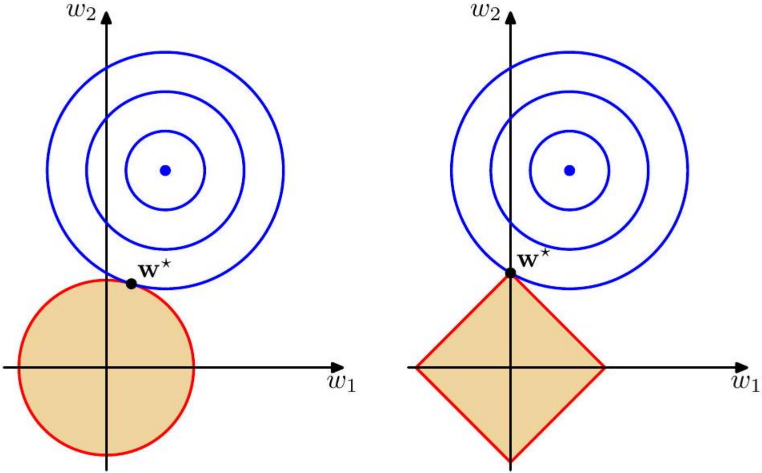
(ridge) $\frac{1}{2}\|\boldsymbol{\theta}\|_2^2 = \frac{1}{2}\boldsymbol{\theta}^\mathsf{T}\boldsymbol{\theta}$

(lasso) $\|\boldsymbol{\theta}\|_1$

Regularization



Regularization



Ridge regression

$$\frac{1}{2}\sum_{n=1}^{N}\left\{y_n - \boldsymbol{\theta}^\mathsf{T}\boldsymbol{\phi}\left(\mathbf{x}_n\right)\right\}^2 + \frac{\lambda}{2}\boldsymbol{\theta}^\mathsf{T}\boldsymbol{\theta}$$

Bayesian regression approach

Bayesian regression is a probabilistic approach that treats model parameters as random variables.

It combines prior beliefs with data to obtain posterior distributions of the parameters, allowing for uncertainty quantification and flexible modeling.

- Prior Distribution: Initial belief about the model parameters.
- Likelihood: Information from the observed data.
- Posterior Distribution: Updated belief after combining prior and likelihood.

Bayesian Regression Formula

The posterior distribution is calculated using Bayes' theorem:

$$P(\boldsymbol{\theta}|\mathbf{x}) \propto P(\mathbf{x}|\boldsymbol{\theta}) \cdot P(\boldsymbol{\theta})$$

where:

- $P(\boldsymbol{\theta}|\mathbf{t})$ is the posterior distribution of parameters $\boldsymbol{\theta}$ given data.
- $P(\mathbf{x}|\boldsymbol{\theta})$ is the likelihood of observing data $\mathbf{x}$ given parameters $\boldsymbol{\theta}$.
- $P(\boldsymbol{\theta})$ is the prior distribution of parameters $\boldsymbol{\theta}$.

Ridge Regression as a Bayesian Model

Ridge regression can be viewed as a Bayesian model with a normal prior on the weights:

$$\frac{1}{2}\sum_{n=1}^{N}\left\{t_n - \boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{\phi}\left(\mathbf{x}_n\right)\right\}^2 + \frac{\lambda}{2}\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{\theta}$$

This corresponds to a likelihood term (first part) and a prior term (second part), where λ controls the strength of the prior.

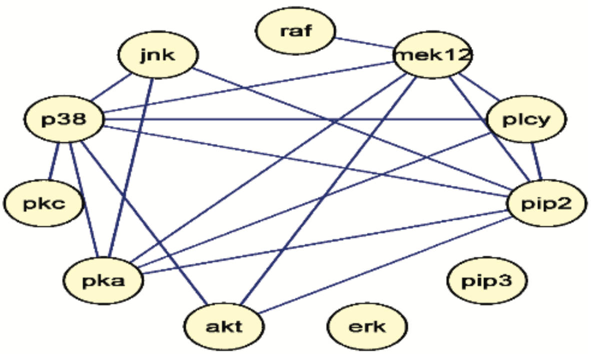
$$\begin{aligned} P(\boldsymbol{\theta}|\mathbf{x},\lambda) &\propto P(\mathbf{x}|\boldsymbol{\theta}) \cdot P(\boldsymbol{\theta}|\lambda) \\ &\propto \exp\left(-\frac{1}{2}\sum_{n=1}^{N}\left\{t_n - \boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{\phi}\left(\mathbf{x}_n\right)\right\}^2\right) \cdot \exp\left(-\frac{\lambda}{2}\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{\theta}\right) \end{aligned}$$

Interpreting Model Uncertainty

Bayesian methods provide a comprehensive view of uncertainty by quantifying both **aleatoric** (data variability) and **epistemic** (parameter uncertainty) uncertainties.

- **Credible Intervals**: Provide a range within which the true parameter value is likely to lie.
- **Posterior Predictive Distribution**: Reflects both types of uncertainty, allowing for informed decision-making.
- **Uncertainty Quantification**: Essential for understanding the reliability of predictions and managing risks.

Example: Causal discovery



Linear Models for Classification

The classification problem

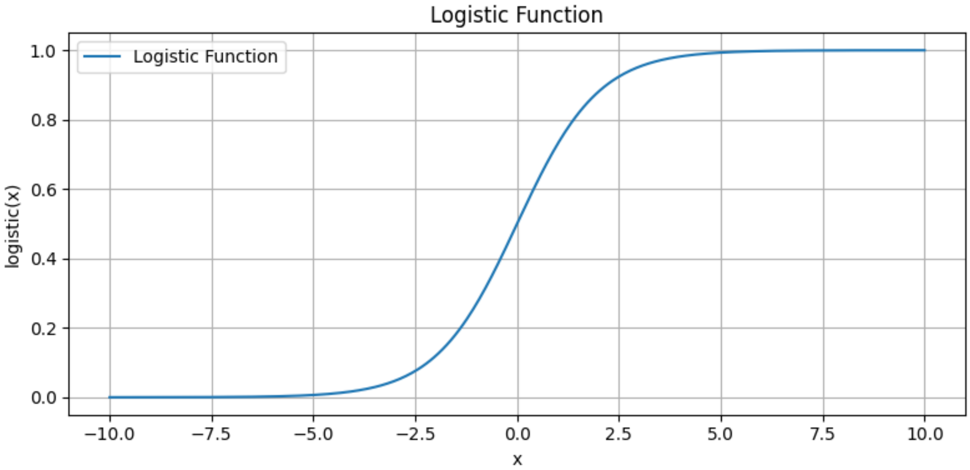
- Classification predicts discrete values for y (unlike regression which predicts continuous values)
- Focus on binary classification where $y \in \{0, 1\}$:
  - 0 = Negative class (denoted by ".")
  - 1 = Positive class (denoted by "+")
- Applications generalize to multi-class problems (more than two categories)
- Example: Email spam classification
  - $x^{(i)}$ = Features of email
  - $y^{(i)} = 1$ for spam, 0 for non-spam
- Training example pair: Input $x^{(i)}$ with corresponding label $y^{(i)}$

Logistic Regression vs. Linear Regression for Classification

- Linear regression unsuitable for classification.
    - Produces values outside $[0, 1]$ range
    - Poor performance on discrete $y \in \{0, 1\}$
    - Solution: Modified hypothesis using logistic function
- Logistic Regression Hypothesis:
    - $h_\theta(x) = g(\theta^T x)$
    - Logistic/Sigmoid function definition:

$$g(z) = \frac{1}{1 + e^{-z}}$$

Logistic function for Classification



Logistic function for Classification

- Bounded output: $0 < h_\theta(x) < 1$
- Natural probabilistic interpretation
- Smooth, S-shaped curve (sigmoid)
- Useful property of the derivative:

$$\begin{aligned} g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\ &= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\ &= \frac{1}{(1+e^{-z})}\cdot\left(1-\frac{1}{(1+e^{-z})}\right) \\ &= g(z)(1-g(z)). \end{aligned}$$

Maximum likelihood estimator

- Let us assume that

$$P(y = 1 \mid x; \theta) = h_\theta(x)$$
$$P(y = 0 \mid x; \theta) = 1 - h_\theta(x)$$

- Note that this can be written more compactly as

$$p(y \mid x; \theta) = (h_\theta(x))^y \,(1 - h_\theta(x))^{1-y}$$

- Likelihood function:

$$\begin{aligned} L(\theta) &= p(\vec{y} \mid X; \theta) \\ &= \prod_{i=1}^{n} p\left(y^{(i)} \mid x^{(i)}; \theta\right) \\ &= \prod_{i=1}^{n} \left(h_\theta\left(x^{(i)}\right)\right)^{y^{(i)}} \left(1 - h_\theta\left(x^{(i)}\right)\right)^{1-y^{(i)}} \end{aligned}$$

Maximum likelihood estimator

- As in the regression model, it will be easier to maximize the log likelihood:

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} y^{(i)} \log h\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h\left(x^{(i)}\right)\right)$$

- Gradient ascent update:

$$\theta := \theta + \alpha \nabla_{\theta} \ell(\theta)$$

- Stochastic gradient ascent rule?

Maximum likelihood estimator

- Stochastic gradient ascent rule:

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j}\ell(\theta) &= \left(y\frac{1}{g\left(\theta^T x\right)} - (1-y)\frac{1}{1-g\left(\theta^T x\right)}\right)\frac{\partial}{\partial \theta_j} g\left(\theta^T x\right) \\
&= \left(y\frac{1}{g\left(\theta^T x\right)} - (1-y)\frac{1}{1-g\left(\theta^T x\right)}\right) g\left(\theta^T x\right)\left(1-g\left(\theta^T x\right)\right)\frac{\partial}{\partial \theta_j}\theta^T x \\
&= \left(y\left(1-g\left(\theta^T x\right)\right) - (1-y) g\left(\theta^T x\right)\right) x_j \\
&= \left(y - h_\theta(x)\right) x_j
\end{aligned}
$$

$$
\therefore \theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right) x_j^{(i)}
$$

Logistic Loss Definition

Alternative notation for logistic regression loss:

- Logistic loss function definition:

$$\ell_{\text{logistic}} : \mathbb{R} \times \{0,1\} \to \mathbb{R}_{\geq 0}$$

$$\ell_{\text{logistic}}(t, y) \triangleq y \log(1 + \exp(-t)) + (1 - y) \log(1 + \exp(t))$$

- Connection to negative log-likelihood:

$$-\ell(\theta) = \ell_{\text{logistic}}(\theta^\top x, y)$$

- $\theta^\top x$ is called the **logit**

Derivative Analysis

- First derivative of logistic loss:

$$\frac{\partial \ell_{\text{logistic}}(t,y)}{\partial t} = y\frac{-\exp(-t)}{1+\exp(-t)} + (1-y)\frac{1}{1+\exp(-t)}$$
$$= \frac{1}{1+\exp(-t)} - y$$

- Chain rule application for parameter gradient:

$$\frac{\partial}{\partial \theta_j}\ell(\theta) = -\frac{\partial \ell_{\text{logistic}}(t,y)}{\partial t} \cdot \frac{\partial t}{\partial \theta_j}$$
$$= (y - \frac{1}{1+\exp(-t)}) \cdot x_j$$
$$= (y - h_\theta(x))x_j$$

Multi-class Classification

- Response variable y can take on any one of k values: $y \in \{1, 2, \ldots, k\}$
- $p(y \mid x; \theta)$ is a distribution over k possible discrete outcomes (multinomial distribution)
- Multinomial distribution involves k probabilities $\phi_1, \ldots, \phi_k$, where $\sum_{i=1}^{k} \phi_i = 1$
- Goal: Design a parameterized model that outputs $\phi_1, \ldots, \phi_k$ given input x

Parameter Groups

- Introduce k groups of parameters $\theta_1, \dots, \theta_k$, each $\theta_i \in \mathbb{R}^d$
- Aim to use $\theta_1^\top x, \dots, \theta_k^\top x$ to represent probabilities $P(y = 1 \mid x; \theta), \dots, P(y = k \mid x; \theta)$
- Challenges:
  - $\theta_j^\top x$ may not be within $[0, 1]$
  - Sum of $\theta_j^\top x$'s may not equal 1
- Solution: Use softmax function

Softmax Function

Definition

The softmax function softmax $: \mathbb{R}^k \to \mathbb{R}^k$ is defined as:

$$\text{softmax}\left(t_1, \ldots, t_k\right) = \begin{bmatrix} \frac{\exp(t_1)}{\sum_{j=1}^{k} \exp\left(t_j\right)} \\ \vdots \\ \frac{\exp(t_k)}{\sum_{j=1}^{k} \exp\left(t_j\right)} \end{bmatrix}$$

- Inputs t to softmax are called logits
- Output is always a probability vector (non-negative entries summing to 1)

Probabilistic Model with Softmax

- Define logits as $t_i = \theta_i^\top x$
- Apply softmax to get probabilities:

$$\begin{bmatrix} P(y=1 \mid x;\theta) \\ \vdots \\ P(y=k \mid x;\theta) \end{bmatrix} = \begin{bmatrix} \frac{\exp(t_1)}{\sum_{j=1}^{k} \exp(t_j)} \\ \vdots \\ \frac{\exp(t_k)}{\sum_{j=1}^{k} \exp(t_j)} \end{bmatrix}$$

- Compact form:

$$P(y=i \mid x;\theta) = \phi_i = \frac{\exp(t_i)}{\sum_{j=1}^{k} \exp(t_j)} = \frac{\exp(\theta_i^\top x)}{\sum_{j=1}^{k} \exp(\theta_j^\top x)}$$

Negative Log-Likelihood Derivation

Single Example Loss - Negative log-likelihood for (x, y)

$$
\begin{aligned}
-\log p(y \mid x, \theta) &= -\log \left(\frac{\exp(t_y)}{\sum_{j=1}^{k} \exp(t_j)} \right) \\
&= -t_y + \log \left(\sum_{j=1}^{k} \exp(t_j) \right) \\
&= -\log \left(\frac{\exp(\theta_y^\top x)}{\sum_{j=1}^{k} \exp(\theta_j^\top x)} \right) \quad \text{[expressed with parameters]}
\end{aligned}
$$

where $\exp(\theta_y^\top x)$ is the correct class score and $\sum_{j=1}^{k} \exp(\theta_j^\top x)$ is the sum of all class scores.

Overall Loss Function

Negative Log-Likelihood Loss

The total loss over n training examples $(x^{(i)}, y^{(i)})$ and k possible classes:

$$\ell(\theta) = \sum_{i=1}^{n} -\log\left(\frac{\exp\left(\theta_{y^{(i)}}^\top x^{(i)}\right)}{\sum_{j=1}^{k} \exp\left(\theta_j^\top x^{(i)}\right)}\right)$$

where $\theta_j \in \mathbb{R}^d$.

Measures discrepancy between predicted probabilities and true labels

Cross-Entropy Loss Definition

Modular Component

Define cross-entropy loss for any logits and label:

$$\ell_{\text{ce}} : \mathbb{R}^k \times \{1, \dots, k\} \to \mathbb{R}_{\geq 0}$$

$$\ell_{\text{ce}}\left((t_1, \dots, t_k), y\right) = -\log\left(\frac{\exp(t_y)}{\sum_{j=1}^{k} \exp(t_j)}\right)$$

- Inputs: logits $t_j = \theta_j^\top x$ and true label y
- Output: Non-negative loss value

Modular Loss Formulation

Combined Expression

Using cross-entropy loss notation:

$$\ell(\theta) = \sum_{i=1}^{n} \ell_{\text{ce}}\left((\theta_1^\top x^{(i)}, \ldots, \theta_k^\top x^{(i)}), y^{(i)}\right) = \sum_{\text{examples}} \ell_{\text{ce}}(\text{logits}, \text{label})$$

Gradient of Cross-Entropy Loss

For cross-entropy loss with softmax probabilities $\phi_i = \frac{\text{exp}(t_i)}{\sum_j \text{exp}(t_j)}$:

$$\frac{\partial \ell_{\text{ce}}(t, y)}{\partial t_i} = \phi_i - \mathbb{1}\{y = i\}$$

$\mathbb{1}\{y = i\}$ is the indicator function (1 if true, 0 otherwise)

Gradient of Cross-Entropy Loss

Vectorized form

$$\frac{\partial \ell_{\text{ce}}(t, y)}{\partial t} = \phi - e_y$$

- $e_y \in \mathbb{R}^k$: y-th natural basis vector (one-hot encoding)
- $\phi \in \mathbb{R}^k$: predicted probability vector

Parameter Gradients - SGD/mini-batch updates?

- For one example (x, y):

$$\frac{\partial \ell_{\text{ce}}}{\partial \theta_i} = \frac{\partial \ell}{\partial t_i} \cdot \frac{\partial t_i}{\partial \theta_i}$$
$$= \underbrace{(\phi_i - \mathbb{1}\{y = i\})}_{\text{loss gradient}} \cdot \underbrace{x}_{\text{input features}}$$

- Across all examples $(x^{(j)}, y^{(j)})$, $j = 1, \ldots, n$:

$$\frac{\partial \ell(\theta)}{\partial \theta_i} = \sum_{j=1}^{n} \left(\phi_i^{(j)} - \mathbb{1}\{y^{(j)} = i\}\right) \cdot x^{(j)}$$

where $\phi_i^{(j)}$ is the model's predicted probability for class i on example j.

Practical implementation

- Compute gradients for each class separately
- Update rule for gradient descent:

$$\theta_i \leftarrow \theta_i - \eta \frac{\partial \ell(\theta)}{\partial \theta_i}$$