

<https://bit.ly/2I3JKMY>

KYOTO UNIVERSITY

Statistical Learning Theory - Regression -

Hisashi Kashima

DEPARTMENT OF INTELLIGENCE SCIENCE
AND TECHNOLOGY

Linear Regression

Regression:

Supervised learning for predicting a real valued variable

- Regression learning is one of supervised learning problem settings with wide applications
- Goal: Obtain a function $f: \mathcal{X} \rightarrow \mathfrak{R}$ (\mathfrak{R} : real value)
 - Usually, input domain \mathcal{X} is a D -dimensional vector space
 - E.g. $x \in \mathcal{X}$ is a house and $y \in \mathfrak{R}$ is its price (housing dataset in UCI Machine Learning Repository)
- Training dataset: N pairs of an input and an output $\{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(N)}, y^{(N)})\}$
 - We use the training dataset to estimate f

Some applications of regression:

From marketing prediction to chemo-informatics

- Some applications:

- Price prediction: Predict the price y of a product x
- Demand prediction: Predict the demanded amount y of a product x
- Sales prediction: Predict the sales amount y of a product x
- Chemical activity: Predict the activity level y of a compound x

- Other applications:

- Time series prediction: Predict the value y at the next time step given the past measurements x
- Classification (has a discrete output domain)

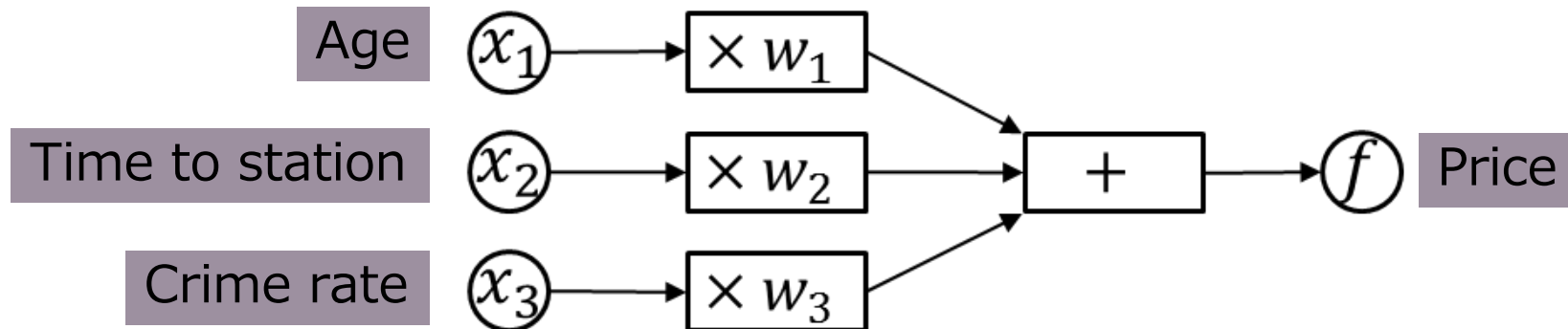
Model:

Linear regression model

- Model: How does y depend on \mathbf{x} ?
- We consider the simplest choices: Linear regression model

$$y = \mathbf{w}^\top \mathbf{x} = w_1 x_1 + w_2 x_2 + \cdots + w_D x_D$$

—Prediction model of the price of a house:



Handling discrete features:

Dummy variables

- We assume input \mathbf{x} is a real vector
 - In the house price prediction example, features can be age, walk time to the nearest station, crime rate in the area, ...
 - They are considered as real values
- How do we handle discrete features as real values?
 - Binary features: {Male, Female} are encoded as {0,1}
 - One-hot encoding: {Kyoto, Osaka, Tokyo} are encoded with (1,0,0), (0,1,0), and (0,0,1)
 - Called dummy variables

Objective function of training:

Squared loss

- Objective function (to minimize):
Disagreement measure of the model to the training dataset
 - Loss function: $\ell^{(i)}(y^{(i)}, \mathbf{w}^\top \mathbf{x}^{(i)})$ for the i -th instance
 - Objective function: $L(\mathbf{w}) = \sum_{i=1}^N \ell^{(i)}(y^{(i)}, \mathbf{w}^\top \mathbf{x}^{(i)})$

- Squared loss function:

$$\ell^{(i)}(y^{(i)}, \mathbf{w}^\top \mathbf{x}^{(i)}) = (y^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)})^2$$

- Absolute loss, Huber loss: more robust choices
- Optimal parameter \mathbf{w}^* is the one that minimizes $L(\mathbf{w})$:

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} L(\mathbf{w})$$

Solution of linear regression problem:

One dimensional case

- Let us start with a case where inputs and outputs are both one-dimensional
- Objective function to minimize:

$$L(w) = \sum_{i=1}^N (y^{(i)} - wx^{(i)})^2$$

- Solution: $w^* = \frac{\sum_{i=1}^N y^{(i)}x^{(i)}}{\sum_{i=1}^N x^{(i)2}} = \frac{\text{Cov}(x,y)}{\text{Var}(x)}$

- Solve $\frac{\partial L(w)}{\partial w} = 0$

Solution of linear regression problem: General multi-dimensional case

- Matrix and vector notations:

- Design matrix $\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}]^\top$

- Target vector $\mathbf{y} = (y^{(1)}, y^{(2)}, \dots, y^{(N)})^\top$

- Objective function:

$$\begin{aligned} L(\mathbf{w}) &= \sum_{i=1}^N (y^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)})^2 = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \\ &= (\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) \end{aligned}$$

- Solution: $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} L(\mathbf{w}) = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$

Example:

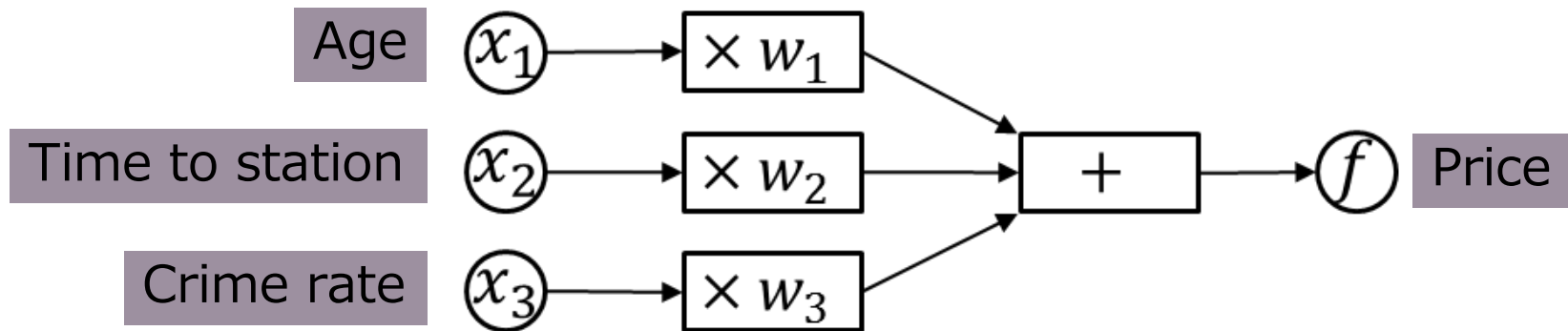
House price prediction

- Design matrix:

$$\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \mathbf{x}^{(4)}]^T = \left[\begin{pmatrix} 15 \\ 10 \\ 1.0 \end{pmatrix}, \begin{pmatrix} 3 \\ 1 \\ 0.1 \end{pmatrix}, \begin{pmatrix} 35 \\ 5 \\ 7.0 \end{pmatrix}, \begin{pmatrix} 40 \\ 70 \\ 1.0 \end{pmatrix} \right]^T$$

- Target vector:

$$\mathbf{y} = (y^{(1)}, y^{(2)}, y^{(3)}, y^{(4)})^T = (140, 85, 220, 115)^T$$



Regularization

Ridge regression:

Include penalty on the norm of \mathbf{w} to avoid instability

- Existence of the solution $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ requires that $\mathbf{X}^\top \mathbf{X}$ is non-singular, i.e. full-rank
 - This is often secured when the number of data instances N is much larger than the number of dimensions D
- Regularization: Adding some constant $\lambda > 0$ to the diagonals of $\mathbf{X}^\top \mathbf{X}$ for numerical stability
 - Modified solution: $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$
- Back to its objective function, the new solution corresponds to
$$L(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$
 - $\lambda \|\mathbf{w}\|_2^2$ is called a (L2-)regularization term

Overfitting:

Degradation of predictive performance for future data

- Previously, we introduced the regularization term to avoid numerical stability
- Another interpretation: To avoid *overfitting* to the training data
 - Our goal is to make correct predictions for future data, not for the training data
 - Overfitting: Too much adaptation to the training data degrades predictive performance on future data
- When the number of data instances N is less than the number of dimensions D , the solution is not unique
 - Infinite number of solutions exist

Occam's razor:

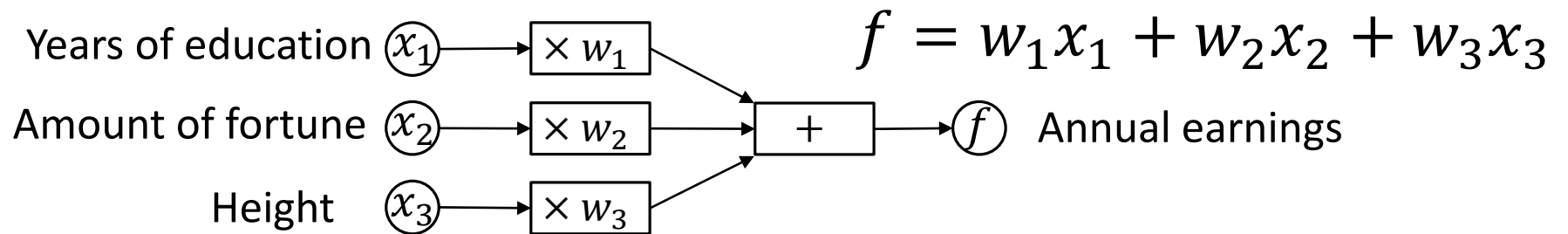
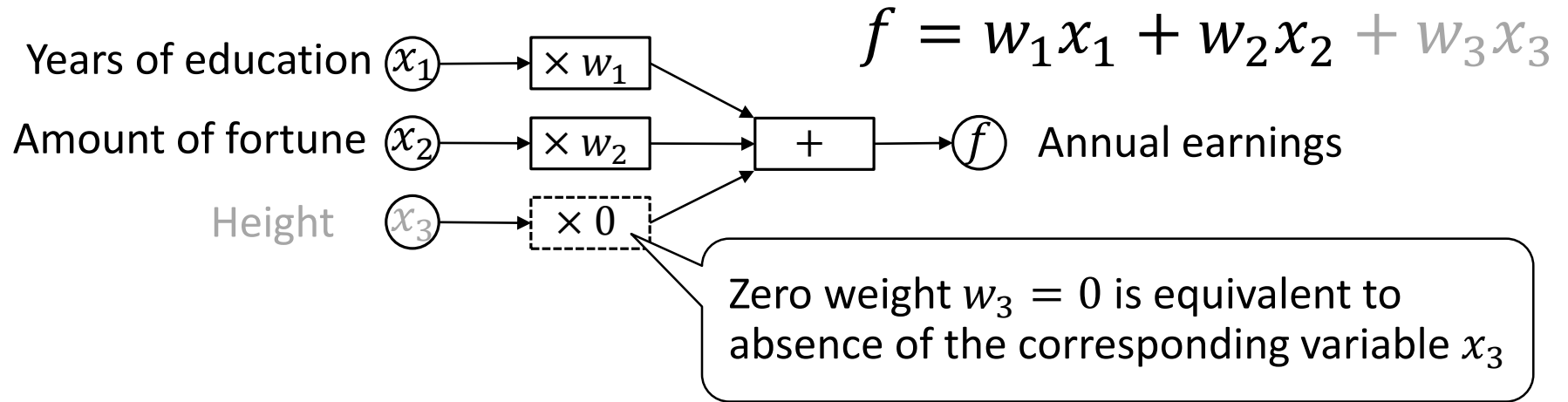
Adopt the simplest model

- We have infinite number of models that equally fit to the training data (=minimize the loss function)
 - Some perform well, some perform badly
- Which is the “best” model among them?
- Occam's razor principle: “Take the simplest model”
 - We will discuss why the simple model is good later in the “statistical learning theory”
- What is the measure of simplicity?
For example, number of features = the number of non-zero elements in \mathbf{w}

Occam's razor:

Prefers models with smaller number of variables

- Occam's razor principle prefers



0-norm regularization:

Reduces the number of non-zero elements in \mathbf{w}

- Number of non-zero elements in \mathbf{w} = “0-norm of \mathbf{w} ”

Number of
features used in
the model

- Use 0-norm constraint:

$$\text{minimize}_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \text{ s. t. } \|\mathbf{w}\|_0 \leq \eta$$

or 0-norm penalty:

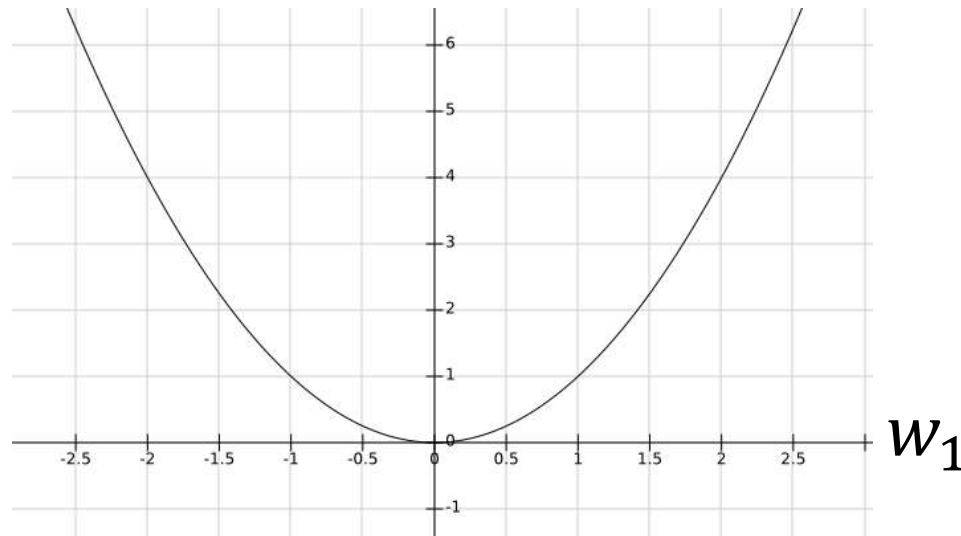
$$\text{minimize}_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_0$$

- There is some one-to-one correspondence between η and λ
- However, they are non-convex optimization problems ...
 - Hard to find the optimal solution

Ridge regression :

2-norm regularization as a convex surrogate for 0-norm

- Instead of the zero-norm $\|\mathbf{w}\|_0$, we use 2-norm $\|\mathbf{w}\|_2^2$



Convex 😊

- Ridge regression: $L(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$

–Can be seen as a relaxed(?) version of

$$L(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_0$$

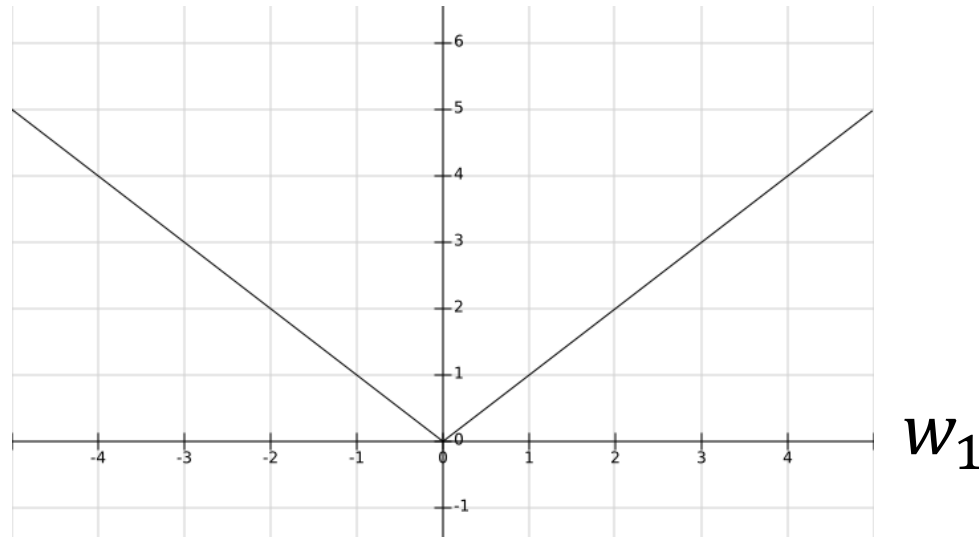
Non-convex 😞

–The closed form solution: $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$

Lasso :

1-norm regularization further induces sparsity

- Instead, we can use 1-norm $\|\mathbf{w}\|_1 = |w_1| + |w_2| + \cdots + |w_D|$



- Lasso: $L(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_1$
 - Convex optimization, but no closed form solution
- Sparsity inducing norm: 1-norm induces sparse \mathbf{w}^*

Statistical Interpretation

Interpretation as statistical inference :

Regression as maximum likelihood estimation

- So far we have formulated the regression problem in *loss minimization framework*
 - Function (prediction model) $f: \mathcal{X} \rightarrow \mathfrak{R}$ is deterministic
 - Least squares: Minimization of the sum of squared losses
- We have not considered any statistical inference
- Actually, we can interpret the previous formulation in a statistical inference framework, namely, *maximum likelihood estimation*

Maximum likelihood estimation (MLE):

Find the parameter that best reproduces training data

- We consider f as a conditional distribution $f_{\mathbf{w}}(y|\mathbf{x})$

Conditional probability

- Maximum likelihood estimation (MLE):

- Find \mathbf{w} that maximizes the likelihood function:

$$L(\mathbf{w}) = \prod_{i=1}^N f_{\mathbf{w}}(y^{(i)}|\mathbf{x}^{(i)})$$

- Likelihood function: Probability that the training data is reproduced by the model
- We assume i.i.d. (which will be explained next)

- It is often convenient to use *log* likelihood instead:

$$L(\mathbf{w}) = \sum_{i=1}^N \log f_{\mathbf{w}}(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w})$$

Important assumption on data:

Identically and independently distributed

- We assume data are *identically and independently distributed*:
 - Data instances are generated from the same data generation mechanism (i.e. probability distribution)
 - Furthermore, past data (training data) and future data (test data) have the same property
 - Data instances are independent of each other

Probabilistic version of the linear regression model:

Gaussian linear model

- Probabilistic version of the linear regression model $y = \mathbf{w}^\top \mathbf{x}$
- $y \sim \mathcal{N}(\mathbf{w}^\top \mathbf{x}, \sigma^2)$: Gaussian distribution with mean $\mathbf{w}^\top \mathbf{x}$ and variance σ^2

$$f_{\mathbf{w}}(y|\mathbf{x}) = \mathcal{N}(\mathbf{w}^\top \mathbf{x}, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y - \mathbf{w}^\top \mathbf{x})^2}{2\sigma^2}\right)$$

– In other words, $y = \mathbf{w}^\top \mathbf{x} + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$

Linear
regression
model

Relation between least squares and MLE:

Maximum likelihood is equivalent to least squares

- Log-likelihood function:

$$\begin{aligned} L(\mathbf{w}) &= \sum_{i=1}^N \log f_{\mathbf{w}}(y^{(i)} | \mathbf{x}^{(i)}) \\ &= \sum_{i=1}^N \log \frac{1}{\sqrt{2\pi}\sigma} \exp \left(-\frac{(y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2}{2\sigma^2} \right) \\ &= -\frac{1}{2\sigma^2} \sum_{i=1}^N (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2 + \text{const.} \end{aligned}$$

- Maximization of $L(\mathbf{w})$ is equivalent to minimization of the squared loss $\sum_{i=1}^N (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2$

Some More Applications

Time series prediction:

Auto regressive (AR) model

- Time series data: A sequence of real valued data $x_1, x_2, \dots, x_t, \dots \in \mathbb{R}$ associated with time stamps $t = 1, 2, \dots$
- Time series prediction: Given x_1, x_2, \dots, x_{t-1} , predict x_t
- Auto regressive (AR) model:

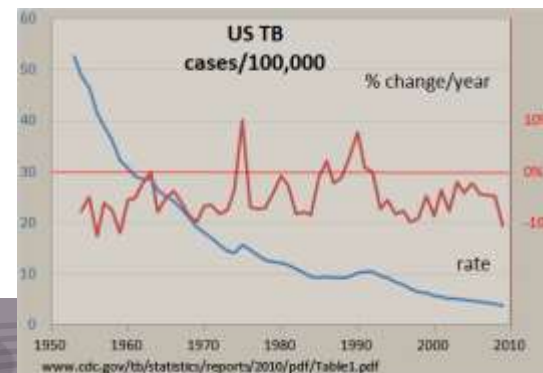
$$x_t = w_1 x_{t-1} + w_2 x_{t-2} + \dots + w_D x_{t-D}$$

— x_t is determined by the recent length- D history

- AR model as a linear regression model $y = \mathbf{w}^\top \mathbf{x}$:

— $\mathbf{w} = (w_1, w_2, \dots, w_D)^\top$

— $\mathbf{x} = (x_{t-1}, x_{t-2}, \dots, x_{t-D})^\top$



Classification as regression:

Regression is also applicable to classification

- Binary classification: $y \in \{+1, -1\}$
- Apply regression to predict $y \in \{+1, -1\}$
- Rigorously, such application is not valid
 - Since an output is either $+1$ or -1 , the Gaussian noise assumption does not hold
 - However, since solution of regression is often easier than that of classification, this application can be compromise
- Fisher discriminant: Instead of $\{+1, -1\}$, use $\left\{+\frac{1}{N^+}, -\frac{1}{N^-}\right\}$
 - $N^+(N^-)$ is the number of positive (negative) data

Nonlinear Regression

Nonlinear regression:

Introducing nonlinearity in linear models

- So far we have considered only linear models
- How to introduce non-linearity in the models?
 1. Introduce nonlinear basis functions:
 - Transformed features: e.g. $x \rightarrow \log x$
 - Cross terms: e.g. $x_1, x_2 \rightarrow x_1 x_2$
 - Kernels: $\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x})$ (some nonlinear mapping to a high-dimensional space)
 2. Intrinsically nonlinear models:
 - Regression tree / random forest
 - Neural network

Nonlinear transformation of features:

Simplest way to introduce nonlinearity in linear models

- Nonlinear basis function: $x \rightarrow \log x, e^x, x^2, \frac{1}{x}, \dots$
 - Sometimes used for converting the range
 - E.g. $\log: \mathbb{R}^+ \rightarrow \mathbb{R}$, $\exp: \mathbb{R} \rightarrow \mathbb{R}^+$
- Interpretations of log transformation:

	y	$\log y$
x	$y = \beta x + \alpha$ Increase of x by 1 will increase y by β	$\log y = \beta x + \alpha$ Increase of x by 1 will multiply y by $1 + \beta$
$\log x$	$y = \beta \log x + \alpha$ Doubling x will increase y by β	$\log y = \beta \log x + \alpha$ Doubling x will multiply y by $1 + \beta$

Cross terms:

Can include synergetic effects among different features

- Not only the original features x_1, x_2, \dots, x_D , use their cross terms products $\{x_d x_{d'}\}_{d,d'}$

- Model has a matrix parameter \mathbf{W} :

$$y = \text{Trace} \left(\begin{bmatrix} w_{1,1} & \cdots & w_{1,D} \\ \vdots & \ddots & \vdots \\ w_{D,1} & \cdots & w_{D,D} \end{bmatrix}^\top \begin{bmatrix} x_1^2 & x_1 x_2 & \cdots & x_1 x_D \\ x_2 x_1 & x_2^2 & \cdots & x_2 x_D \\ \vdots & \vdots & \ddots & \vdots \\ x_D x_1 & x_D x_2 & \cdots & x_D^2 \end{bmatrix} \right)$$
$$= \mathbf{x}^\top \mathbf{W}^\top \mathbf{x}$$

- $L(\mathbf{W}) = \sum_{i=1}^N \left(y^{(i)} - \mathbf{x}^{(i)\top} \mathbf{W}^\top \mathbf{x}^{(i)} \right)^2 + \lambda \|\mathbf{W}\|_F^2$

(e.g. factorization machines)

Kernels:

Linear model in a high-dimensional feature space

- High dimensional non-linear mapping: $\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x})$
 - $\boldsymbol{\phi}: \mathcal{R}^D \rightarrow \mathcal{R}^{\bar{D}}$ is some nonlinear mapping from D -dimensional space to a \bar{D} -dimensional space ($D \ll \bar{D}$)
- Linear model $y = \bar{\mathbf{w}}^\top \boldsymbol{\phi}(\mathbf{x})$
- Kernel regression model: $y = \sum_{i=1}^N \alpha^{(i)} k(\mathbf{x}^{(i)}, \mathbf{x})$
 - Kernel function $k(\mathbf{x}^{(i)}, \mathbf{x}) = \langle \boldsymbol{\phi}(\mathbf{x}^{(i)}), \boldsymbol{\phi}(\mathbf{x}) \rangle$: inner product
 - Kernel trick: Instead of working in the \bar{D} -dimensional space, we use an equivalent form in an N -dimensional space
 - Foundation of kernel machines, e.g. SVM, Gaussian process, ...

Bayesian Statistical Interpretation

Bayesian interpretation of regression:

Ridge regression as MAP estimation

- We consider another statistical interpretation of linear regression in terms of Bayesian statistics
 - Which justifies ridge regression
- Ridge regression as MAP estimation
 - Posterior distribution of parameters
 - Maximum A Posteriori (MAP) estimation

Bayesian modeling:

Posterior distribution instead of likelihood

- In maximum likelihood estimation (MLE), we obtain \mathbf{w} that maximizes data *likelihood*:

$$P(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = \prod_{i=1}^N f(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w})$$

$$\text{or } \log P(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = \sum_{i=1}^N \log f(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w})$$

- The probability of the data reproduced with the parameter:
 $P(\text{Data} \mid \text{Parameters})$
- In Bayesian modeling, we consider the *posterior distribution*
 $P(\text{Parameters} \mid \text{Data})$
 - Posterior distribution is the distribution over model parameters given data

Posterior distribution:

Log posterior = log likelihood + log prior

■ Posterior distribution:

$$P(\text{Parameters} \mid \text{Data}) = \frac{P(\text{Data} \mid \text{Parameters})P(\text{Parameters})}{P(\text{Data})}$$

(Bayes' formula)

■ Log posterior:

$$\begin{aligned} \log P(\text{Parameters} \mid \text{Data}) \\ = \underbrace{\log P(\text{Data} \mid \text{Parameters})}_{\text{Likelihood}} + \underbrace{\log P(\text{Parameters})}_{\text{Prior}} \end{aligned}$$

$$-\log P(\text{Data})$$

- $P(\text{Data})$ is a constant term and often neglected because it does not depend on the parameters

Maximum a posteriori (MAP) estimation:

Find parameter that maximizes the posterior

- Maximum a posteriori (MAP) estimation finds the parameter that maximizes the posterior:

$$\text{Parameters}^* = \operatorname{argmax}_{\text{Parameters}} \log P(\text{Parameters} \mid \text{Data})$$

- Maximization of the log posterior:

$$\log P(\text{Parameters} \mid \text{Data})$$

$$= \log P(\text{Data} \mid \text{Parameters}) + \log P(\text{Parameters}) + \text{const.}$$

—MLE considers only $\log P(\text{Data} \mid \text{Parameters})$ part

—Additional term (log prior) : $\log P(\text{Parameters})$

Ridge regression as MAP estimation:

Find parameter that maximizes the posterior

- Log posterior: $\log P(\text{Parameters} \mid \text{Data}) = \underbrace{\log P(\text{Data} \mid \text{Parameters})}_{\text{Log likelihood}} + \underbrace{\log P(\text{Parameters})}_{\text{Log prior}} + \text{const.}$

- Log-likelihood: $\sum_{i=1}^N \log \frac{1}{\sqrt{2\pi}\sigma'} \exp \left(-\frac{(y^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)})^2}{2\sigma'^2} \right)$

- Prior $P(\mathbf{w}) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left(-\frac{\mathbf{w}^\top \mathbf{w}}{2\sigma^2} \right)$

- Ridge regression:

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} \frac{1}{2\sigma^2} \sum_{i=1}^N (y^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)})^2 + \frac{1}{2\sigma'^2} \|\mathbf{w}\|_2^2$$

Regression:

Supervised learning for predicting a real valued variable

- A supervised learning problem to make real-valued predictions
- Regression problem is often formulated as a least-square minimization problem
 - Closed form solution is given
- Regularization framework to avoid overfitting
 - Reduce the number of features: 0-norm, 2-norm (ridge regression), 1-norm (lasso)
- Nonlinear regression
- Statistical interpretations: maximum likelihood estimation, maximum a posteriori (MAP) estimation