Introduction to Machine Learning
10-315 Fall ‘19

Lecture 24:
NonLinear and Kernel regression 1

Teacher:
Gianni A. Di Caro
Least Squares Linear regression: summary and terminology

- Linear regression for **univariate models** \( f(x), \quad f: \mathbb{R} \to \mathbb{R} \). (also known **single regression**)
  Hypothesized model: \( y = f(x) = w_0 + w_1 x = w^T x \)

- Linear regression for **multivariate models** \( f(x), \quad f: \mathbb{R}^d \to \mathbb{R} \) (also known a **multiple regression**)
  Hypothesized model: \( y = f(x) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d = w^T x \)

- General case: Linear **multivariate regression** for **multivariate models** \( f(x), \quad f: \mathbb{R}^d \to \mathbb{R}^k \)
  Hypothesized model: \( y = [f_1(x), f_2(x), \ldots, f_k(x)]^T = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d = w^T x \)
Least Squares Linear regression: summary and terminology

OLS Optimization problem and solution:

\[
\min_w (Xw - Y)^T (Xw - Y) \\
\hat{w} = (X^TX)^{-1}X^TY \\
\hat{y}(x) = \hat{w}^T x
\]

- Prediction as a **dot product** between learned feature weights and input feature vector:
  \[
  \hat{y}(x) = \hat{w}^T x = \langle w, x \rangle
  \]

- Prediction as a **linear blend of the predictions** from the training set:
  \[
  \hat{y}(x) = x^T(X^TX)^{-1}X^TY = \sum_{i=1}^{m} b_i(x, X)y^{(i)}
  \]

Design matrix

\[
X = \begin{bmatrix}
  x_1^{(1)} & \cdots & x_d^{(1)} \\
  \vdots & \ddots & \vdots \\
  x_1^{(m)} & \cdots & x_d^{(m)}
\end{bmatrix}
\]

Number of features, \(d\)

Number of samples, \(m\)

\[
Y = \begin{bmatrix}
  y^{(1)} \\
  \vdots \\
  y^{(m)}
\end{bmatrix}
\]
What about situations where the functional relationships between predictor variables and the outputs is expressed by nonlinear relations?

- In general, if the function cannot be expressed by a linear combination of the coefficients, there’s no closed-form solution as in the case of linear regression and least squares.
- Numeric methods need to be applied, and the optimization problems become non-convex → many local minima + non-trivial risk of finding a biased solution.
Nonlinear regression → Approximation, Linearization

- General idea (to reuse results and methods from linear LS): **Approximate** the model function by **linearizing** it with respect to the estimation parameters \( \mathbf{w} \)

- **Taylor series** of order \( n \), for a univariate model:

\[
f(x) \approx f(x_0) + \frac{f'(x_0)}{1!} (x - x_0) + \frac{f''(x_0)}{2!} (x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!} (x - x_0)^n
\]

\[
f(x) \approx \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!} (x - x_0)^n
\]

- **Taylor series** of order \( n \), for a multivariate model:

\[
f(x) \approx \sum_{|a| \geq 0} \frac{(x - x_0)^a}{a!} (\partial^a f)(x_0) \xrightarrow{x_0=0} \sum_{|a| \geq 0} \frac{(x)^a}{a!} (\partial^a f)(x_0)
\]

(includes cross-product and partial derivatives among variables)
Nonlinear regression → Approximation, Linearization

\[ f(x) \approx \sum_{n=0}^{\infty} w_n x^n \]

\[ f(x) \approx \sum_{|a| \geq 0} \frac{(x)^a}{a!} (\partial^a f)(x_0) \]

How can we exploit these expressions to find \( f \rightarrow \) find \( w_n \) using the data?

✓ Approximate the function by linearizing: truncate the Taylor series at the first term, get a linear approximation and work with that to estimate the parameters using OLS

- Any function can be approximated with the desired precision by using **polynomials**

- Polynomials are an example of **basis functions**: form a basis in a function space such that any function can be represented using the basis

- For a given input predictor \( x \), from the estimation point of view, the above expressions are linear in the parameters to be estimated, \( w \)
Non-linear, additive regression models

- **Main idea to model non-linearities**: Replace inputs to linear units with \( b \) feature (basis) functions \( \phi_j(x), \ j = 1, \ldots, b \), where \( \phi_j(x) \) is an arbitrary function of \( x \)

\[
y = f(x; w) = w_0 + w_1 \phi_1(x) + w_2 \phi_2(x) + \cdots + w_b \phi_b(x) = w^T \cdot \phi(x)
\]

- Fitting data to a **nonlinear model**
- **Linear as** statistical estimation problem
- The regression function \( E[y \mid x] \) is linear in the unknown parameters \( w \) that are estimated from the data.

- **If we add enough** basis function we are guaranteed to approximate the model function very well!
Examples of Feature Functions: Polynomial regression

- Higher order polynomial with one-dimensional input, \( x = (x) \)
  - \( \phi_1(x) = x, \phi_2(x) = x^2, \phi_3(x) = x^3, \ldots \)
  - The feature space is transformed and enlarged

- Quadratic polynomial with two-dimensional inputs, \( x = (x_1, x_2) \)
  - \( \phi_1(x) = x_1, \phi_2(x) = x_1^2, \phi_3(x) = x_2, \phi_4(x) = x_2^2, \phi_5(x) = x_1 x_2 \)

- E.g., if our hypothesis mode is defined by a polynomial of degree 4 defined over a two-feature space \( (x_1, x_2) \), the hypothesis function has 14 parameters:

\[
f_w = w_1 x_1^4 + w_2 x_2^4 + w_3 x_1^3 x_2 + w_4 x_1 x_2^3 + w_5 x_1^2 x_2^2 + w_6 x_1^3 + w_7 x_2^3 + w_8 x_1^2 x_2 + w_9 x_1 x_2^2 + w_{10} x_2^2 + w_{11} x_1 x_2 + w_{12} x_1 + w_{13} x_2 + w_{14}
\]
Solution using Feature Functions

- The same techniques (analytical gradient + system of equations, or gradient descent) used for the plain linear case with MSE as loss function

\[ f(x^{(i)}; w) = w_0 + w_1 \phi_1(x^{(i)}) + w_2 \phi_2(x^{(i)}) + \cdots + w_b \phi_b(x^{(i)}) = w^T \cdot \phi(x^{(i)}) \]

\[ \ell = \frac{1}{m} \sum_{i=1}^{m} (y^{(i)} - f(x^{(i)}))^2 \]

\[ \phi(x^{(i)}) = (1, \phi_1(x^{(i)}), \phi_2(x^{(i)}), \cdots, \phi_b(x^{(i)})) \]

- To find \( \min_w \ell \) we have to look where \( \nabla_w \ell = 0 \)

\[ \nabla_w \ell = -\frac{2}{m} \sum_{i=1}^{m} (y^{(i)} - f(x^{(i)})) \phi(x^{(i)}) = 0 \]
Solution using Feature Functions

\[ f(x^{(i)}; w) = w_0 + w_1 \phi_1(x^{(i)}) + w_2 \phi_2(x^{(i)}) + \cdots + w_b \phi_b(x^{(i)}) = w^T \cdot \phi(x^{(i)}) \]

\[ \phi(x^{(i)}) = (1, \phi_1(x^{(i)}), \phi_2(x^{(i)}), \ldots, \phi_b(x^{(i)})) = (\phi_0, \phi_1(x^{(i)}), \phi_2(x^{(i)}), \ldots, \phi_b(x^{(i)})) \]

\[ -\frac{2}{m} \sum_{i=1}^{m} (y^{(i)} - f(x^{(i)})) \phi(x^{(i)}) = 0 \]

\[ w_0 \sum_{i=1}^{m} 1 \phi_j(x^{(i)}) + w_1 \sum_{i=1}^{m} \phi_1(x^{(i)}) \phi_j(x^{(i)}) + \cdots + w_k \sum_{i=1}^{m} \phi_k(x^{(i)}) \phi_j(x^{(i)}) + \cdots + w_b \sum_{i=1}^{m} \phi_b(x^{(i)}) \phi_j(x^{(i)}) \]

\[ = \sum_{i=1}^{m} y_i \phi_j(x^{(i)}) \quad \forall j = 1, \ldots, b \]
Solution using Feature Functions

\[ w_0 \sum_{i=1}^{m} \phi_j(x^{(i)}) + w_1 \sum_{i=1}^{m} \phi_1(x^{(i)}) \phi_j(x^{(i)}) + \ldots \]
\[ + w_k \sum_{i=1}^{m} \phi_k(x^{(i)}) \phi_j(x^{(i)}) \ldots + w_b \sum_{i=1}^{m} \phi_b(x^{(i)}) \phi_j(x^{(i)}) = \sum_{i=1}^{m} y_i \phi_j(x^{(i)}) \quad \forall j = 1, \ldots, b \]

In matrix form:

\[ (A^T A)w = A^T Y \]

Unregularized LS

\[ w = (A^T A)^{-1} A^T Y \]

Ridge regularized LS

\[ w = (A^T A + \lambda I_m)^{-1} A^T Y \]

\[ w = (A^T A + \lambda I_m)^{-1} A^T Y \]

\[ \hat{f}(x) = \langle \phi(x), w \rangle \quad \phi(x) = [\phi_0(x) \phi_1(x) \ldots \phi_b(x)] \]

\[ \hat{f}(x) = \langle b(\phi(x), A), Y \rangle \quad b = \phi^T(x)(A^T A)^{-1} A^T \]
Example of polynomial regression

Univariate (1-dim) \( f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_m X^m = X\beta \)

where \( X = [1 \ X \ X^2 \ldots X^m] \), \( \beta = [\beta_1 \ldots \beta_m]^T \)

\[ \hat{\beta} = (A^T A)^{-1} A^T Y \text{ or } (A^T A + \lambda I)^{-1} A^T Y \]

\( \hat{f}_n(X) = X \hat{\beta} \)

where \( A = \begin{bmatrix} 1 & X_1 & X_1^2 & \cdots & X_1^m \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & X_n & X_n^2 & \cdots & X_n^m \end{bmatrix} \)

Multivariate (p-dim) \( f(X) = \beta_0 + \beta_1 X^{(1)} + \beta_2 X^{(2)} + \cdots + \beta_p X^{(p)} \)

\[ + \sum_{i=1}^{p} \sum_{j=1}^{p} \beta_{ij} X^{(i)} X^{(j)} + \sum_{i=1}^{p} \sum_{j=1}^{p} \sum_{k=1}^{p} X^{(i)} X^{(j)} X^{(k)} \]

\[ + \ldots \text{terms up to degree m} \]
Other basis functions? Correlation and locality issues

\[ f(X) = \sum_{j=0}^{m} \beta_j \phi_j(X) \]

Basis coefficients \( \beta_j \) \( \phi_j(X) \) (Linear combinations yield meaningful spaces)

Polynomial Basis
- \( \phi_0(X) \)
- \( \phi_1(X) \)
- \( \phi_2(X) \)

Fourier Basis
- \( \phi_0(X) \)
- \( \phi_1(X) \)
- \( \phi_2(X) \)

Wavelet Basis
- \( \phi_0(X) \)
- \( \phi_1(X) \)
- \( \phi_2(X) \)

Good representation for periodic functions

Good representation for local functions
Electricity example: linear vs. nonlinear data

New data: it doesn’t look linear anymore
New hypothesis: which model complexity?

The complexity of the model grows: one parameter for each feature transformed according to a polynomial of order 2 (at least 3 parameters vs. 2 of original hypothesis)
New hypothesis: which model complexity?

At least 5 parameters (if we had multiple predicting features, all their order $d$ products should be considered, resulting into a number of additional parameters)
New hypothesis: which model complexity?

The number of parameters is now larger than the data points, such that the polynomial can almost precisely fit the data → **Overfitting**
Another example: Polynomial model

- The red curve is the true function (which is not a polynomial)

- The data points are samples from the curve with added noise in y.

- There is a choice in both the degree, M, of the basis functions used, and in the strength of the regularization

\[
f(x, \mathbf{w}) = \sum_{j=0}^{M} w_j x^j = \mathbf{w}^\top \Phi(x)
\]

\[
\Phi : x \rightarrow \Phi(x) \quad \mathbb{R} \rightarrow \mathbb{R}^{M+1}
\]

\[
\tilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} \{f(x_i, \mathbf{w}) - y_i\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2
\]

w is a M+1 dimensional vector
Another example: Polynomial model

N = 9 samples, M = 7

Regularized solution for different levels of regularization

Least Squares solution for different polynomials

M = 3  
M = 5
Another example: Gaussian basis functions

• The red curve is the true function (which is not a polynomial)
• The data points are samples from the curve with added noise in y.

• Basis functions are centred on the training data (N points)
• There is a choice in both the scale, sigma, of the basis functions used, and in the strength of the regularization

\[
f(x, \mathbf{w}) = \sum_{i=1}^{N} w_i e^{-(x-x_i)^2/\sigma^2} = \mathbf{w}^\top \Phi(x)\quad \Phi : x \to \Phi(x) \quad \mathbb{R} \to \mathbb{R}^N
\]

\[
\tilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} \{f(x_i, \mathbf{w}) - y_i\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2 \quad \text{w is a N-vector}
\]
Another example: Gaussian basis functions

$N = 9$ samples, $\sigma = 0.334$
Right Bias-Variance tradeoff?

3 Independent training datasets

Large bias, Small variance – poor approximation but robust/stable

Small bias, Large variance – good approximation but unstable
Selecting Model Complexity

- Dataset with 10 points, 1D features: which hypothesis function should we use?
  - **Linear regression**: \( y = f(x; w) = w_0 + w_1 x \)
  - **Polynomial regression**, cubic: \( y = f(x; w) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 \)
- MSE for the loss functions
- Which model would give the smaller error in terms of MSE / least squares fit?
Selecting Model Complexity

- Cubic regression provides a better fit to the data, and a smaller MSE
- Should we stick with the hypothesis $f(x; \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$?
- Since a higher order polynomial seems to provide a better fit, why don’t we use a polynomial of order higher than 3?
- What is the highest order that makes sense for the given problem?
Selecting Model Complexity

- For 10 data points, a degree 9 polynomial gives a perfect fit (Lagrange interpolation): 0 error
- Is it always good to minimize (even reduce to zero) the training error?
- Related (and more important) question: How do we (will) perform on new, unseen data?
Overfitting

- The 9-polynomial model totally fails the prediction for the new point!

- **Overfitting**: Situation when the *training error is low* and the *generalization error is high*. Causes of the phenomenon:
  - Highly complex hypothesis model, with a large number of parameters (degrees of freedom)
  - Small data size (as compared to the complexity of the model)

- The learned function has enough degrees of freedom to (over)fit all data perfectly
Training and Validation Loss

- **Fundamental problem**: we are looking for parameters that optimize the generalization loss, using the empirical loss.

- Solving the minimization problems for the empirical loss not necessarily brings the same optimal generalization loss because of:
  - *unrealizability*: the true hypothesis is not included in the considered universe
  - *variance*: resulting from sampling different subsets of the possible data
  - *noise*: predictions can differ for the same samples
  - *computational complexity*: it might not be feasible to solve the problem to optimality

- Divide data into **training set** (used to find parameters for a fixed hypothesis class \( h_\theta \)), and **validation set** (used to choose hypothesis class)

- What is the negative effect of doing this?
Splitting dataset in two

![Graph showing split dataset in two](image)
Performance on Validation set

Training set and validation set, fourth degree polynomial
Performance on Validation set

Training set and validation set, 30th degree polynomial: how the loss looks like?
Increasing model complexity

In this case, the small size of the dataset favors an easy overfitting by increasing the degree of the polynomial (i.e., hypothesis complexity). For a large multi-dimensional dataset this effect is less strong / evident.
Training vs. Validation Loss

- General intuition for training and validation loss

- We would like to choose hypothesis class that is at the “sweet spot” of minimizing validation loss
Model Selection and Evaluation Process

1. Break all available data into training and testing sets (e.g., 70% / 30%)

2. Break training set into training and validation sets (e.g., 70% / 30%)

3. Loop:
   i. Set a hyperparameter value (e.g., degree of polynomial → model complexity)
   ii. Train the model using training sets
   iii. Validate the model using validation sets
   iv. Exit loop if (validation errors keep growing && training errors go to zero)

4. Choose hyperparameters using validation set results: hyperparameter values corresponding to lowest validation errors

5. (Optional) With the selected hyperparameters, retrain the model using all training data sets

6. Evaluate (generalization) performance on the testing sets
Model Selection and Evaluation Process

Dataset → Training set → Internal training set → Learn 1 → Validate 1

Dataset → Training set → Validation set → Learn 2 → Validate 2

Dataset → Training set → Learn n → Validate n

Testing set → Learn * → Select best model → Model *
Can we use kernels?

- Recall the ridge regression problem

\[ \mathbf{w} = \arg \min_{\mathbf{w}} \sum_{n=1}^{N} (y_n - \mathbf{w}^\top \mathbf{x}_n)^2 + \lambda \mathbf{w}^\top \mathbf{w} \]

- The solution to this problem was

\[ \mathbf{w} = \left( \sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^\top + \lambda \mathbf{I}_D \right) \left( \sum_{n=1}^{N} y_n \mathbf{x}_n \right) = \left( \mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_D \right)^{-1} \mathbf{X}^\top \mathbf{y} \]

- Inputs don’t appear as inner-products here. They actually do! :-)

- Matrix inversion lemma: $(\mathbf{F} \mathbf{H}^{-1} \mathbf{G} - \mathbf{E})^{-1} \mathbf{F} \mathbf{H}^{-1} = \mathbf{E}^{-1} \mathbf{F} (\mathbf{G} \mathbf{E}^{-1} \mathbf{F} - \mathbf{H})^{-1}$

- The lemma allows us to rewrite $\mathbf{w}$ as

\[ \mathbf{w} = \mathbf{X}^\top (\mathbf{X} \mathbf{X}^\top + \lambda \mathbf{I}_N)^{-1} \mathbf{y} = \mathbf{X}^\top \alpha = \sum_{n=1}^{N} \alpha_n \mathbf{x}_n \]

where $\alpha = (\mathbf{X} \mathbf{X}^\top + \lambda \mathbf{I}_N)^{-1} \mathbf{y} = (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{y}$ is an $N \times 1$ vector of dual variables, and $K_{nm} = \mathbf{x}_n^\top \mathbf{x}_m$

\[ \mathbf{w} = \sum_{n=1}^{N} \alpha_n \mathbf{x}_n \] is known as “dual” form of ridge regression solution. Linear in the samples.
Can we use kernels?

- With the dual form \( \mathbf{w} = \sum_{n=1}^{N} \alpha_n \mathbf{x}_n \), we can kernelize ridge regression.

- Choosing some kernel \( k \) with an associated feature map \( \phi \), we can write
  \[ \mathbf{w} = \sum_{n=1}^{N} \alpha_n \phi(\mathbf{x}_n) = \sum_{n=1}^{N} \alpha_n k(\mathbf{x}_n, \cdot) \]
  where \( \alpha = (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{y} \) and \( K_{nm} = \phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) \).

- Prediction for a new test input \( \mathbf{x} \) will be
  \[ y = \mathbf{w}^\top \phi(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n \phi(\mathbf{x}_n)^\top \phi(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n k(\mathbf{x}_n, \mathbf{x}) \]

- Thus, using the kernel, we effectively learn a nonlinear regression model.

Prediction costs go as \( O(N) \)
Choosing kernels

\[ \hat{f}_n(X) = K_X (K + \lambda I)^{-1} Y \]

where

\[ K_X(i) = \phi(X) \cdot \phi(X_i) \]

\[ K(i, j) = \phi(X_i) \cdot \phi(X_j) \]

Work with kernels, never need to write out the high-dim vectors

Examples of kernels:

Polynomials of degree exactly \( d \)

\[ K(u, v) = (u \cdot v)^d \]

Polynomials of degree up to \( d \)

\[ K(u, v) = (u \cdot v + 1)^d \]

Gaussian/Radial kernels

\[ K(u, v) = \exp \left( -\frac{||u - v||^2}{2\sigma^2} \right) \]

Ridge Regression with (implicit) nonlinear features \( \phi(X) \)

\[ f(X) = \phi(X) \beta \]