CPSC 340: Machine Learning and Data Mining

Nonlinear Regression
Fall 2022
Last Time: Linear Regression

- We discussed linear models:
  \[
  y_i = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id}
  = \sum_{j=1}^{d} w_j x_{ij}
  \]
- "Multiply feature \( x_{ij} \) by weight \( w_j \), add them to get \( y_i \)."
- We discussed squared error function:
  \[
  f(w) = \frac{1}{2} \sum_{i=1}^{n} (w^\top x_i - y_i)^2
  \]
  - Minimize ‘f’ by equating gradient of ‘f’ with zero.
- Interactive demo:

Matrix/Norm Notation (MEMORIZE/STUDY THIS)

• We typically assume that vectors are column-vectors.
  – We use ‘w’ as a “d times 1” vector containing weight ‘w_j’ in position ‘j’.
  – We use ‘y’ as an “n times 1” vector containing target ‘y_i’ in position ‘i’.
  – We use ‘x_i’ as a “d times 1” vector containing features ‘j’ of example ‘i’.
    • We’re now going to be careful to make sure these are column vectors.
  – So ‘X’ is a matrix with x_i^T in row ‘i’.
Matrix/Norm Notation (MEMORIZE/STUDY THIS)

- We showed how to express various quantities in matrix notation:
  - Linear regression prediction for one example: $\hat{y}_i = w^T x_i$
  - Linear regression prediction for all ‘n’ examples: $\hat{y} = Xw$
  - Linear regression residual vector: $r = Xw - y$
  - Sum of residuals squared in linear regression model:
    $$\mathcal{L}(w) = \sum_{i=1}^{n} (\sum_{j=1}^{d} w_j x_{ij} - y_i)^2 = \| Xw - y \|^2$$
  - Today: derive gradient and least squares solution in matrix notation.
Digression: Matrix Algebra Review

• Quick review of **linear algebra operations** we’ll use:
  – If ‘a’ and ‘b’ be vectors, and ‘A’ and ‘B’ be matrices then:

\[
\begin{align*}
\alpha^T \beta &= \beta^T \alpha \\
\| \alpha \|^2 &= \alpha^T \alpha \\
(A + B)^T &= A^T + B^T \\
(AB)^T &= B^T A^T \\
(A + B)(A + B) &= AA + BA + AB + BB \\
\alpha^T A \beta &= \beta^T A^T \alpha
\end{align*}
\]

*Sanity check:*

**ALWAYS CHECK THAT DIMENSIONS MATCH**

(if not, you did something wrong)
Linear and Quadratic Gradients

• From these rules we have (see post-lecture slide for steps):

\[
f(w) = \frac{1}{2} \sum_{i=1}^{n} (w^T x_i - y_i)^2 = \frac{1}{2} \|Xw - y\|^2 = \frac{1}{2} w^T X^T X w - w^T X^T y + \frac{1}{2} y^T y
\]

\[
= \frac{1}{2} w^T A w - w^T b + c
\]

These are scalars so dimensions match.

• How do we compute gradient?

Let’s first do it with \( d=1 \):

\[
f(w) = \frac{1}{2} w a w + w b + c
\]

\[
= \frac{1}{2} a w^2 + w b + c
\]

\[
f'(w) = a w + b + 0
\]

Here are the generalizations to \( d \) dimensions:

\[
\nabla [c] = 0 \quad \text{(zero vector)}
\]

\[
\nabla [w^T b] = b
\]

\[
\nabla \left[ \frac{1}{2} w^T A w \right] = A w \quad \text{(if A is symmetric)}
\]

Full derivations are on webpage in notes on linear and quadratic gradients.
Linear and Quadratic Gradients

• From these rules we have (see post-lecture slide for steps):

\[ f(w) = \frac{1}{2} \sum_{i=1}^{n} (w^T x_i - y_i)^2 = \frac{1}{2} \| w^T x - y \|^2 = \frac{1}{2} w^T X^T X w - w^T X^T y + \frac{1}{2} y^T y \]

\[ = \frac{1}{2} w^T A w - w^T b + c \]

• Gradient is given by:

\[ \nabla f(w) = A w - b + \delta \]

• Using definitions of ‘A’ and ‘b’: 

\[ = X^T X w - X^T y \]

---

Sanity check: all dimensions match

\((d \times n) (n \times d) (d \times 1) - (d \times n) (n \times 1)\)
Normal Equations for Least Squares Solution

- Set gradient equal to zero to find the “critical” points:
  \[ \mathbf{X}^\top \mathbf{X}_w - \mathbf{X}^\top \mathbf{y} = \mathbf{0} \]

- We now move terms not involving ‘w’ to the other side:
  \[ \mathbf{X}^\top \mathbf{X}_w = \mathbf{X}^\top \mathbf{y} \]

- This is a set of ‘d’ linear equations called the normal equations.
  - This a linear system like “Ax = b” from Math 152.
    - You can use Gaussian elimination to solve for ‘w’.
  - In Julia, the “\" command can be used to solve linear systems:
    \[
    \text{Train: } \mathbf{w} = (\mathbf{X}^\top \mathbf{X}) \backslash (\mathbf{X}^\top \mathbf{y})
    \]
    \[
    \text{Predict: } \hat{\mathbf{y}} = \mathbf{X}_{\text{test}} \times \mathbf{w}
    \]
Incorrect Solutions to Least Squares Problem

The least squares objective is \( f(w) = \frac{1}{2} \| Xw - y \|^2 \)

The minimizers of this objective are solutions to the linear system:

\[
X^T X w = X^T y
\]

The following are not the solutions to the least squares problem:

\[
w = (X^T X)^{-1}(X^T y) \quad \text{(only true if } X^T X \text{ is invertible)}
\]

\[
w X^T X = X^T y \quad \text{(matrix multiplication is not commutative, dimensions don't even match)}
\]

\[
w = \frac{X^T y}{X^T X} \quad \text{(you cannot divide by a matrix)}
\]
Least Squares Cost

- **Cost** of solving “normal equations” $X^TXw = X^Ty$?
- Forming $X^Ty$ vector costs $O(nd)$.
  - It has ‘d’ elements, and each is an inner product between ‘n’ numbers.
- Forming matrix $X^TX$ costs $O(nd^2)$.
  - It has $d^2$ elements, and each is an inner product between ‘n’ numbers.
- Solving a $d \times d$ system of equations costs $O(d^3)$.
  - Cost of Gaussian elimination on a $d$-variable linear system.
  - Other standard methods have the same cost.
- Overall cost is $O(nd^2 + d^3)$.
  - Which term dominates depends on ‘n’ and ‘d’.
Least Squares Issues

• Issues with least squares model:
  – Solution might not be unique.
  – It is sensitive to outliers.
  – It always uses all features.
  – Data might so big we cannot store $X^TX$.
    • If you have 10 million features, this requires $O(d^2)$.
    • Or you cannot afford the $O(nd^2 + d^3)$ cost.
  – It might predict outside range of $y_i$ values.
    • For some applications, only positive $y_i$ values are valid.
  – It assumes a linear relationship between $x_i$ and $y_i$. 

\[ X \text{ is } n \times d \]
\[ \text{so } X^T \text{ is } d \times n \]
\[ \text{and } X^TX \text{ is } d \times d \]
Non-Uniqueness of Least Squares Solution

• Why is the solution vector ‘w’ not unique?
  – Imagine having two features that are identical for all examples.
  – I can increase weight on one feature, and decrease it on the other, without changing predictions.
  \[
  \hat{y}_i = w_1 x_{i1} + w_2 x_{i2} = (w_1 + w_2)x_{i1} + 0x_{i2}
  \]
  – In this setting, if \((w_1, w_2)\) is a solution then \((w_1 + w_2, 0)\) is another solution.
  – This is special case of features being “collinear”:
    • One feature is a linear function of the others.

• But, any ‘w’ where \(\nabla f(w) = 0\) is a global minimizer of ‘f’.
  – This is due to convexity of ‘f’, which we will discuss later.
Next Topic: Non-Linear Regression
Motivation: Non-Linear Regression

• Many relationships are approximated well by linear function.

“the shorter your sleep, the shorter your lifespan”
Motivation: Non-Linear Regression

- Many relationships are approximated well by linear function.
  - But many are also highly non-linear.

For example, you could have a “u-shape” when too much/little is not good.

(effect of hyper-parameters usually looks like this)
Motivation: Non-Linear Regression

• Many relationships are approximated well by linear function.
  – But many are also highly non-linear.

https://commons.wikimedia.org/wiki/File:CDC_growth_chart_boys_birth_to_36_mths_cj41c017.pdf

Slope could slowly change or reach asymptote.
Motivation: Non-Linear Regression

• Many relationships are approximated well by linear function.
  – But many are also highly non-linear.

“geometric decay”

“exponential growth”

“sigmoidal growth”.

http://www.biology.arizona.edu/biomath/tutorials/Applications/Carbon.html
Motivation: Non-Linear Regression

• Many relationships are approximated well by linear function.
  – But many are also highly non-linear.

“Piecewise linear”: different pieces follow different linear functions.
(or be linear up to asymptote or phase transition)

100 meter times will not go negative!

http://www.at-a-lanta.nl/weia/Progressie.html
http://www.britannica.com/biography/Florence-Griffith-Joyner
Motivation: Non-Linear Regression

• Many relationships are approximated well by linear function.
  – But many are also highly non-linear.

“Periodic” signals.

Electrocardiogram (ECG)

Electroencephalography (EEG)

Motivation: Non-Linear Regression

- Many relationships are approximated well by linear function.
  - But many are also highly non-linear.

“Spike then recover”

[Neuron action potential]

[Response of organs to exercise]

https://en.wikipedia.org/wiki/Biological_neuron_model
Adapting Counting/Distance-Based Methods

• Can adapt classification methods to perform non-linear regression:

http://www.at-atlanta.nl/weia/Progressie.html
Adapting Counting/Distance-Based Methods

• Can adapt classification methods to perform non-linear regression:
  – Regression tree: tree with mean value or linear regression at leaves.
Adapting Counting/Distance-Based Methods

• Can adapt classification methods to perform non-linear regression:
  – Regression tree: tree with mean value or linear regression at leaves.
  – Probabilistic models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
    • Take CPSC 440.

https://en.wikipedia.org/wiki/Multivariate_normal_distribution
Adapting Counting/Distance-Based Methods

- Can adapt classification methods to perform non-linear regression:
  - Regression tree: tree with mean value or linear regression at leaves.
  - **Probabilistic** models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  - **Non-parametric models**:
    - KNN regression:
      - Find ‘k’ nearest neighbours of $x_i$.
      - Return the mean of the corresponding $y_i$.

Adapting Counting/Distance-Based Methods

• Can adapt classification methods to perform non-linear regression:
  – Regression tree: tree with mean value or linear regression at leaves.
  – **Probabilistic** models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  – Non-parametric models:
    • KNN regression.
    • Could be **weighted by distance**.
      – Close points ‘j’ get more “weight” $w_{ij}$.

Adapting Counting/Distance-Based Methods

• Can adapt classification methods to perform non-linear regression:
  – Regression tree: tree with mean value or linear regression at leaves.
  – Probabilistic models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  – Non-parametric models:
    • KNN regression.
    • Could be weighted by distance.
    • ‘Nadaraya-Watson’: weight all $y_i$ by distance to $x_i$.
      \[
      \hat{y}_i = \frac{\sum_{j=1}^{n} v_{ij} y_j}{\sum_{j=1}^{n} v_{ij}}
      \]

http://www.mathworks.com/matlabcentral/fileexchange/35316-kernel-regression-with-variable-window-width/content/ksr_vw.m
Adapting Counting/Distance-Based Methods

• Can adapt classification methods to perform non-linear regression:
  – Regression tree: tree with mean value or linear regression at leaves.
  – Probabilistic models: fit $p(x_i \mid y_i)$ and $p(y_i)$ with Gaussian or other model.
  – Non-parametric models:
    • KNN regression.
    • Could be weighted by distance.
    • ‘Nadaraya-Watson’: weight all $y_i$.
    • ‘Locally linear regression’: for each $x_i$, fit a linear model weighted by distance.
      (Better than KNN and NW at boundaries.)

Adapting Counting/Distance-Based Methods

• Can adapt classification methods to perform non-linear regression:
  – Regression tree: tree with mean value or linear regression at leaves.
  – Probabilistic models: fit \( p(x_i \mid y_i) \) and \( p(y_i) \) with Gaussian or other model.
  – Non-parametric models:
    • KNN regression.
    • Could be weighted by distance.
    • ‘Nadaraya-Watson’: weight all \( y_i \) by distance to \( x_i \).
    • ‘Locally linear regression’: for each \( x_i \), fit a linear model weighted by distance.
      (Better than KNN and NW at boundaries.)
  – Ensemble methods:
    • Can improve performance by averaging predictions across regression models.
Adapting Counting/Distance-Based Methods

• Applications of non-linear regression (we will see many more):
  – Regression forests for fluid simulation:
  – KNN for image completion:
    • Combined with “graph cuts” and “Poisson blending”.
    • See also “PatchMatch”.
  – KNN regression for “voice photoshop”:
    • Combined with “dynamic time warping” and “Poisson blending”.

• We will first focus on linear models with non-linear transforms.
  – These are the building blocks for more advanced methods.
Why don’t we have a y-intercept?

- Linear model is $\hat{y}_i = wx_i$ instead of $\hat{y}_i = wx_i + w_0$ with y-intercept $w_0$.
- Without an intercept, if $x_i = 0$ then we must predict $\hat{y}_i = 0$.
Why don’t we have a y-intercept?

– Linear model is $\hat{y}_i = wx_i$ instead of $\hat{y}_i = wx_i + w_0$ with y-intercept $w_0$.
– Without an intercept, if $x_i = 0$ then we must predict $\hat{y}_i = 0$. 
Adding a Y-Intercept (“Bias”) Variable

- Simple trick to add a y-intercept (“bias”) variable:
  - Make a new matrix “Z” with an extra feature that is always “1”.

\[
X = \begin{bmatrix}
-0.1 \\
0.3 \\
0.2
\end{bmatrix}
\quad Z = \begin{bmatrix}
1 & -0.1 \\
1 & 0.3 \\
1 & 0.2
\end{bmatrix}
\]

- Now use “Z” as your features in linear regression.
  - We will use ‘v’ instead of ‘w’ as regression weights when we use features ‘Z’.

\[
\hat{y}_i = v_1 Z_{i1} + v_2 Z_{i2} = w_0 + w_1 x_{i1}
\]

- So we can have a non-zero y-intercept by changing features.
  - This means we can ignore the y-intercept to make cleaner derivations/code.
Motivation: Limitations of Linear Models

• On many datasets, $y_i$ is not a linear function of $x_i$.

$$\hat{y}_i = w_0 + w_1 x_i$$

• A quadratic function would be a better fit for this dataset.
Non-Linear Feature Transforms

• Can we use linear least squares to fit a quadratic model?
  \[ \hat{y}_i = w_0 + w_1 x_i + w_2 x_i^2 \]
  – Notice that this is a non-linear function of \( x_i \) but a linear function of \( w \).

• So you can implement this by changing the features:
  \[
  X = \begin{bmatrix}
  0.2 \\
  -0.5 \\
  4 \\
  
  \end{bmatrix}
  Z = \begin{bmatrix}
  1 & 0.2 & (0.2)^2 \\
  1 & -0.5 & (-0.5)^2 \\
  1 & 4 & (4)^2 \\
  y - \text{inf} & x & x^2 \\
  \end{bmatrix}
  \]
  – Fit new parameters \( 'v' \) under “change of basis”: solve \( Z^T Z v = Z^T y \).

• It’s a linear function of \( w \), but a quadratic function of \( x_i \).
  \[ \hat{y}_i = V^T Z_i = v_1 Z_{i1} + v_2 Z_{i2} + v_3 Z_{i3} \]
  \[ = w_0 + w_1 x_i + w_2 x_i^2 \]
Non-Linear Feature Transforms

\[ y_i = w_0 + w_1 x_i \]

\[ y_i = \sum \alpha_i z_i + \sum \beta_i z_i^2 \]

To predict on new data \( \tilde{X} \), form \( \tilde{Z} \) from \( \tilde{X} \) and take \( y = \tilde{Z} v \).
General Polynomial Features (d=1)

• We can have a polynomial of degree ‘p’ by using these features:

\[
Z = \begin{bmatrix}
1 & x_1 & (x_1)^2 & \cdots & (x_1)^p \\
1 & x_2 & (x_2)^2 & \cdots & (x_2)^p \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_n & (x_n)^2 & \cdots & (x_n)^p 
\end{bmatrix}
\]

• There are polynomial basis functions that are numerically nicer:
  – Such as Lagrange polynomials (see CPSC 303).
General Polynomial Features

If you have more than one feature, you can include interactions:

– With $p=2$, in addition to $(x_{i1})^2$ and $(x_{i2})^2$ you could include $x_{i1}x_{i2}$. 
“Change of Basis” Terminology

• Instead of “nonlinear feature transform”, in machine learning it is common to use the expression “change of basis”.
  – The $z_i$ are the “coordinates in the new basis” of the training example.

• “Change of basis” means something different in math:
  – Math: basis vectors must be linearly independent (in ML we don’t care).
  – Math: change of basis must span the same space (in ML we change space).

• Unfortunately, saying “change of basis” in ML is common.
  – When I say “change of basis”, just think “nonlinear feature transform”.

Linear Basis vs. Nonlinear Basis

Usual linear Regression

Train:
- Use $X$ and $y$ to find $w$

Test:
- Use $\hat{X}$ and $w$ to find $\hat{y}$

Linear regression with change of basis

Train:
- Use $X$ to find $Z$
- Use $Z$ and $y$ to find $w$

Test:
- Use $\hat{X}$ to find $\hat{Z}$
- Use $\hat{Z}$ and $w$ to find $\hat{y}$
• Linear regression with original features:
  – We use ‘X’ as our “n by d” data matrix, and ‘w’ as our parameters.
  – We can find d-dimensional ‘w’ by minimizing the squared error:
    \[ \frac{1}{\lambda} \| Xw - y \|^2 \]

• Linear regression with nonlinear feature transforms:
  – We use ‘Z’ as our “n by k” data matrix, and ‘v’ as our parameters.
  – We can find k-dimensional ‘v’ by minimizing the squared error:
    \[ \frac{1}{\lambda} \| Zv - y \|^2 \]

• Notice that in both cases the target is still ‘y’.
Degree of Polynomial and Fundamental Trade-Off

• As the polynomial degree increases, the training error goes down.

• But approximation error goes up: we start overfitting with large ‘p’.

• Usual approach to selecting degree: validation or cross-validation.

Beyond Polynomial Transformations

• Polynomials are not the only possible transformation:
  – Exponentials, logarithms, trigonometric functions, and so on.
  – The right non-linear transform will vastly improve performance.
    • Later we will see “deep learning” where you try to learn a transformation.

xkcd
Summary

- **Matrix notation** for expressing least squares problem.
- **Normal equations**: solution of least squares as a linear system.
  - Solve $(X^TX)w = (X^Ty)$.
- Solution might not be unique because of **collinearity**.
  - But any solution is optimal because of “convexity”.
- **Tree/probabilistic/non-parametric/ensemble** regression methods.
- **Non-linear transforms**:
  - Allow us to model non-linear relationships with linear models.
- Next time: how to do least squares with a million features.
Linear Least Squares: Expansion Step

Want $w'$ that minimizes

$$f(w) = \frac{1}{2} \sum_{i=1}^{D} (w^T x_i - y_i)^2 = \frac{1}{2} \|Xw - y\|^2$$

Let's expand then compute gradient.

$$= \frac{1}{2} (Xw - y)^T (Xw - y)$$
$$= \frac{1}{2} (w^T X^T - y^T)(Xw - y)$$
$$= \frac{1}{2} (w^T X^T (Xw - y) - y^T (Xw - y))$$
$$= \frac{1}{2} (w^T X^T x_w - w^T X^T y - y^T Xw + y^T y)$$

$$= \frac{1}{2} w^T X^T x_w - w^T X^T y + \frac{1}{2} y^T y$$

Rule:

$$\|a\|^2 = a^T a$$
$$(A + b^T)^T = (A^T + b)^T$$
$$(AB)^T = B^T A^T$$
$$(A + B)C = AC + BC$$
$$A(B + C) = AB + BC$$
$$a^T Ab = b^T A^T a$$

Sanity check: all of these are scalars.
Vector View of Least Squares

• We showed that least squares minimizes:
  \[ f(\mathbf{w}) = \frac{1}{2} \| \mathbf{Xw} - \mathbf{y} \|^2 \]

• The ½ and the squaring don’t change solution, so equivalent to:
  \[ f(\mathbf{w}) = \| \mathbf{Xw} - \mathbf{y} \| \]

• From this viewpoint, least square minimizes Euclidean distance between vector of labels ‘\( y \)’ and vector of predictions \( \mathbf{Xw} \).
• **Householder notation:** set of (fairly-logical) conventions for math.

Use **Greek** letters for **scalars**: \( \alpha = 1, \beta = 3, \gamma = \pi \)

Use **first/last lowercase** letters for **vectors**: 
\[
\begin{align*}
  w &= \begin{bmatrix} 0.1 \end{bmatrix}, \\
  x &= \begin{bmatrix} 0 \end{bmatrix}, \\
  y &= \begin{bmatrix} -1 \end{bmatrix}, \\
  a &= \begin{bmatrix} 1 \end{bmatrix}, \\
  b &= \begin{bmatrix} 0.5 \end{bmatrix}
\end{align*}
\]

Assumed to be **column-vectors**.

Use **first/last uppercase** letters for **matrices**: \( X, Y, W, A, B \)

**Indices** use \( i, j, k \).

**Sizes** use \( m, n, d, p \) and \( k \) (hopefully meaning of \( k \) is obvious from context)

**Sets** use \( S, T, U, V \)

**Functions** use \( f, g, h \)

When I write \( x_i \), I mean “grab row \( i \) of \( X \) and make a **column-vector** with its values.”
Bonuu Slid: Householder(-ish) Notation

- **Householder notation**: set of (fairly-logical) conventions for math:

  Our ultimate least squares notation:
  \[ f(w) = \frac{1}{2} \| Xw - y \|^2 \]

  But if we agree on notation we can quickly understand:
  \[ g(x) = \frac{1}{2} \| Ax - b \|^2 \]

  If we use random notation we get things like:
  \[ h(\beta) = \frac{1}{2} \| R\beta - p \|^2 \]

  Is this the same model?
When does least squares have a unique solution?

• We said that least squares solution is not unique if we have repeated columns.
• But there are other ways it could be non-unique:
  – One column is a scaled version of another column.
  – One column could be the sum of 2 other columns.
  – One column could be three times one column minus four times another.

• Least squares solution is unique if and only if all columns of X are “linearly independent”.
  – No column can be written as a “linear combination” of the others.
  – Many equivalent conditions (see Strang’s linear algebra book):
    • X has “full column rank”, $X^TX$ is invertible, $X^TX$ has non-zero eigenvalues, $\det(X^TX) > 0$.
    – Note that we cannot have independent columns if $d > n$. 