Machine Learning and Data Mining

Linear regression

(adapted from) Prof. Alexander Ihler
Supervised learning

- **Notation**
  - Features $x$
  - Targets $y$
  - Predictions $\hat{y}$
  - Parameters $\theta$

**Program ("Learner")**
Characterized by some "parameters" $\theta$
Procedure (using $\theta$) that outputs a prediction

**Learning algorithm**
Change $\theta$
Improve performance

**Training data (examples)**
- Features
- Feedback / Target values

**Score performance ("cost function")**
Linear regression

- Define form of function $f(x)$ explicitly
- Find a good $f(x)$ within that family

“Predictor”:
Evaluate line:
$$r = \theta_0 + \theta_1 x_1$$
return $r$
\[ \hat{y}(x) = \theta \cdot x^T \]

\[ \theta = [\theta_0 \ \theta_1 \ \theta_2] \]

\[ x = [1 \ x_1 \ x_2] \]

(c) Alexander Ihler
Notation

\[ \hat{y}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots \]

Define “feature” \( x_0 = 1 \) (constant)
Then
\[ \hat{y}(x) = \theta^T x \]

\[ \theta = [\theta_0, \ldots, \theta_n] \]
\[ x = [1, x_1, \ldots, x_n] \]
Measuring error

Observation $y$
Prediction $\hat{y}$

Error or “residual”

$y - \hat{y}(x) = (y - \theta \cdot x^T)$

(c) Alexander Ihler
Mean squared error

• How can we quantify the error?

\[
\text{MSE, } J(\theta) = \frac{1}{m} \sum_j (y(j) - \hat{y}(x(j)))^2
\]

\[
= \frac{1}{m} \sum_j (y(j) - \theta \cdot x(j)^T)^2
\]

• Could choose something else, of course…
  – Computationally convenient (more later)
  – Measures the variance of the residuals
  – Corresponds to likelihood under Gaussian model of “noise”

\[
\mathcal{N}(y ; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma^2} \exp \left\{ -\frac{1}{2\sigma^2} (y - \mu)^2 \right\}
\]

(c) Alexander Ihler
MSE cost function

MSE, \( J(\theta) = \frac{1}{m} \sum_{j} (y^{(j)} - \hat{y}(x^{(j)}))^2 \)

\[ = \frac{1}{m} \sum_{j} (y^{(j)} - \theta \cdot x^{(j)T})^2 \]

- Rewrite using matrix form

\( \theta = [\theta_0, \ldots, \theta_n] \)

\( y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}^T \)

\( X = \begin{bmatrix} x^{(1)}_0 & \cdots & x^{(1)}_n \\ \vdots & \ddots & \vdots \\ x^{(m)}_0 & \cdots & x^{(m)}_n \end{bmatrix} \)

\( J(\theta) = \frac{1}{m} (y^T - \theta X^T) \cdot (y^T - \theta X^T)^T \)

(Matlab) \( \gg e = y^T - \theta \cdot x^T; \quad J = e \cdot e^T / m; \)

(c) Alexander Ihler
Visualizing the cost function

\[ J(\theta) \]

\( \theta_1 \quad \theta_0 \)
Supervised learning

- Notation
  - Features \( x \)
  - Targets \( y \)
  - Predictions \( \hat{y} \)
  - Parameters \( \theta \)

Program ("Learner")
Characterized by some "parameters" \( \theta \)
Procedure (using \( \theta \)) that outputs a prediction

Learning algorithm
Change \( \theta \)
Improve performance

Training data (examples)
Features
Feedback / Target values

Score performance ("cost function")
Finding good parameters

- Want to find parameters which minimize our error…

- Think of a cost “surface”: error residual for that $\theta$…

\[ \hat{\theta} = \arg\min_{\theta} J(\theta) \]
Machine Learning and Data Mining

Linear regression: direct minimization

(adapted from) Prof. Alexander Ihler
MSE Minimum

- Consider a simple problem
  - One feature, two data points
  - Two unknowns: $\mu_0, \mu_1$
  - Two equations:
    \[
    y^{(1)} = \theta_0 + \theta_1 x^{(1)} \\
    y^{(2)} = \theta_0 + \theta_1 x^{(2)}
    \]

- Can solve this system directly:
  \[
  y^T = \theta X^T \quad \Rightarrow \quad \hat{\theta} = y^T (X^T)^{-1}
  \]

- However, most of the time, $m > n$
  - There may be no linear function that hits all the data exactly
  - Instead, solve directly for minimum of MSE function

(c) Alexander Ihler
SSE Minimum

\[ \nabla J(\theta) = -\left(y^T - \theta X^T\right) \cdot X = 0 \]

- Reordering, we have

\[ y^T X - \theta X^T \cdot X = 0 \]
\[ y^T X = \theta X^T \cdot X \]
\[ \theta = y^T X (X^T X)^{-1} \]

- \( X (X^T X)^{-1} \) is called the “pseudo-inverse”

- If \( X^T \) is square and independent, this is the inverse
- If \( m > n \): overdetermined; gives minimum MSE fit

(c) Alexander Ihler
Matlab SSE

- This is easy to solve in Matlab…

\[
\theta = y^T X (X^T X)^{-1}
\]

\[
\begin{align*}
\text{Solution 1: } & \text{ “manual”} \\
& \text{th} = y^T X \text{ inv}(X^T X);
\end{align*}
\]

\[
\begin{align*}
\text{Solution 2: } & \text{ “mldivide”} \\
& \text{th} = y^T / X; \quad \text{th} \times X = y \Rightarrow \text{th} = y/X
\end{align*}
\]
Effects of MSE choice

• Sensitivity to outliers

\[ 16^2 \text{ cost for this one datum} \]

Heavy penalty for large errors

(c) Alexander Ihler
L1 error

\[ \ell_1(\theta) = \sum_j |y^{(j)} - \hat{y}(x^{(j)})| \]
\[ = \sum_j |y - \theta \cdot x^T| \]

(c) Alexander Ihler
Cost functions for regression

\[ \ell_2 : (y - \hat{y})^2 \quad \text{(MSE)} \]

\[ \ell_1 : |y - \hat{y}| \quad \text{(MAE)} \]

Something else entirely…

\[ c - \log(\exp(-(y - \hat{y})^2) + c) \quad \text{(???)} \]

“Arbitrary” functions can’t be solved in closed form…

- use gradient descent
Machine Learning and Data Mining

Linear regression: nonlinear features

(adapted from) Prof. Alexander Ihler
Nonlinear functions

- What if our hypotheses are not lines?
  - Ex: higher-order polynomials
Nonlinear functions

- Single feature \( x \), predict target \( y \):

\[
D = \{ (x^{(j)}, y^{(j)}) \} \quad \hat{y}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3
\]

Add features:

\[
D = \{ ([x^{(j)}, (x^{(j)})^2, (x^{(j)})^3], y^{(j)}) \} \quad \hat{y}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3
\]

Linear regression in new features

- Sometimes useful to think of “feature transform”

\[
\Phi(x) = [1, x, x^2, x^3, \ldots] \quad \hat{y}(x) = \theta \cdot \Phi(x)
\]
Higher-order polynomials

- Fit in the same way
- More “features”
Features

• In general, can use any features we think are useful

• Other information about the problem
  – Sq. footage, location, age, …

• Polynomial functions
  – Features \([1, x, x^2, x^3, \ldots]\)

• Other functions
  – \(1/x, \sqrt{x}, x_1 \cdot x_2, \ldots\)

• “Linear regression” = linear in the parameters
  – Features we can make as complex as we want!
Higher-order polynomials

- Are more features better?
- “Nested” hypotheses
  - 2nd order more general than 1st,
  - 3rd order ““ than 2nd,...
- Fits the observed data better
Overfitting and complexity

- More complex models will always fit the training data better.
- But they may “overfit” the training data, learning complex relationships that are not really present.
Test data

- After training the model
- Go out and get more data from the world
  - New observations (x,y)
- How well does our model perform?
Training versus test error

- Plot MSE as a function of model complexity
  - Polynomial order
- Decreases
  - More complex function fits training data better
- What about new data?
- 0\textsuperscript{th} to 1\textsuperscript{st} order
  - Error decreases
  - Underfitting
- Higher order
  - Error increases
  - Overfitting

(c) Alexander Ihler