02/24/2016

Regression

aka fitting curves to data

- Classification: given sample x, predict class (often binary).
- Regression given sample, x, predict a numerical value.
  - Choose form of regression function $h(x;p)$ with parameters p.
    * like predictor function in classification.
  - Choose a cost function (objective function) to optimize.
    * Usually based on a loss function; e.g. risk function = expected loss.
- Some regression functions:
  1. Linear: $h(x;w,\alpha) = w^T x + \alpha$.
  2. Polynomial.
  3. Logistic: $h(x;w,\alpha) = s(w^T x + \alpha)$. Recall: logistic function $s(\gamma) = \frac{1}{1+e^{-\gamma}}$
- Some loss functions: let z be prediction $h(x)$; y be true value.
  - A. $L(z,y) = (z-y)^2$ <u>squared error</u>.
  - B. $L(z,y) = |z-y|$ <u>absolute error</u>.
  - C. $L(z,y) = -yln(z) - (1-y)ln(1-z)$ <u>logistic loss</u>.
- Some cost functions to minimize:
  - a. $J(h) = \frac{1}{n} sum_{i=1}^{n} L(h(X_i), y_i)$ <u>mean loss</u>.
  - b. $J(h) = \max_{i=1}^{n} L(h(X_i, y_i))$ <u>maximum loss</u>.
  - c. $J(h) = \sum \omega_i L(h(X_i, y_i))$ <u>weighted sum</u>.
  - d. $J(h) = \frac{1}{n} \sum L(h(X_i, y_i)) + \lambda |w|^2$ <u>ℓ_2 penalized/regularized</u>.
  - e. $J(h) = \frac{1}{n} \sum L(h(X_i, y_i)) + \lambda |w|$ <u>ℓ_1 penalized/regularized</u>.
- Some combinations:

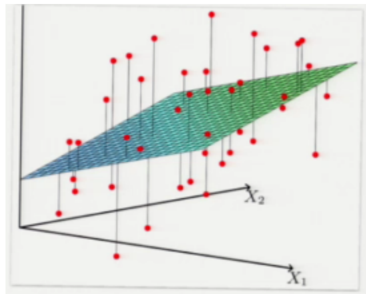
Name	Regression	Loss	Cost	Algorithm
Least-squares linear regression	1	A	a	quadratic, minimize w/calculus
Weighted least-squares linear	1	A	c	quadratic, minimize w/calculus
Ridge regression	1	A	d	quadratic, minimize w/calculus
Lasso	1	A	d	minimize w/gradient descent
Logistic regression	3	C	a	minimize w/gradient descent
Least absolute deviations	1	B	a	minimize w/linear program
Chebyshev criterion	1	B	b	minimize w/linear program

Least-Squares Linear Regression

1 + A + a.

- Optimization problem:

$$\boxed{\text{Find } w, \alpha \text{ that minimizes } \sum_{i=1}^{n}(x_i \cdot w + \alpha - y_i)^2}$$



- Convention:
  - X is nxd design matrix of samples.
  - y is d-vector of dependent scalars.
  - $X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1d} \\ x_{21} & x_{22} & \dots & \\ \vdots & \vdots & \dots & \\ x_{n1} & x_{n2} & \dots & x_{nd} \end{bmatrix}$
  - Usually $n > d$.
  - Sample row vector is X_i^T.
  - Column vector is X_{*j}.
  - $y = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$
  - Recall fictitious dimension trick: replace $x \cdot w + \alpha$ with,

$$[x_1 \; x_2 \; \dots \; x_d \; 1] \cdot \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \\ \alpha \end{bmatrix}$$

    Now X is an nx$(d+1)$ matrix; w is a $(d+1)$-vector.

- Rewritten optimization problem:

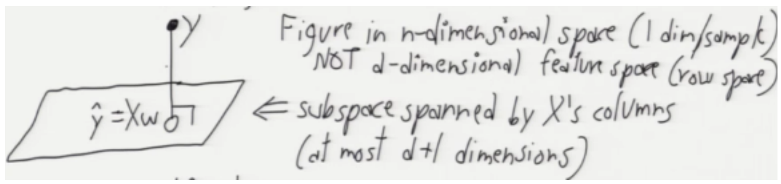
$$\boxed{\text{Find } w \text{ that minimizes } |Xw - y|^2}$$

- Optimize by calculus, minimize residual sum of squares:

$$\begin{aligned} RSS(w) &= w^T X^T X w - 2y^T X w + y^T y = 0 \\ &\implies X^T X w = X^T y \Leftarrow \text{The \underline{normal equations}} \end{aligned}$$

- If $X^T X$ is singular, problem is under-constrained.
- We use a linear solver to find $w = (X^T X)^{-1} X^T y$.
- $X^+ = (X^T X)^{-1} X^T$ is the pseudoinverse of X and is a $(d+1)$xn matrix.

- Observe: $X^+X = (X^TX)^{-1}X^TX = I \Leftarrow (d+1)\text{x}(d+1)$.
- Observe: the predicted values of y are $\hat{y} = h(x;) \Rightarrow = Xw = XX^+y = Hy$.
- where $H = XX^+$ is the <u>hat matrix</u> because it puts the hat on y.
- Interpretation as a projection:
  - $\hat{y} = Xw \in \mathbb{R}^n$ is a linear combination of columns of X.
  - For fixed X, varying w, Xw is a subspace of $\mathbb{R}^n$ spanned by columns.



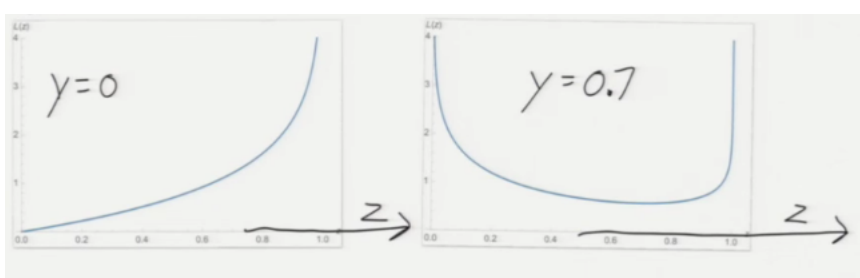
  - Minimizing $|\hat{y} - y|$ find point $\hat{y}$ nearest y on subspace $\Rightarrow$ project y onto subspace.
  - Error is smallest when line is perpendicular to subspace: $X^T(Xw - y) = 0 \Rightarrow$ the normal equations!
  - Hat matrix H (also called projection matrix) does the projecting.
- Advantages:
  - Easy to compute; just solve a linear system.
  - Unique, stable solution.
- Disadvantages:
  - Very sensitive to outliers, because error is squared!
  - Fails if X^TX is singular.

Logistic Regression

(David Cox, 1953) 3 + C + a

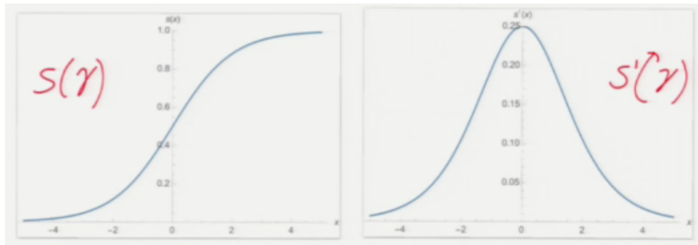
- Fits "probabilities" in range (0, 1).
- Usually used for classification. The input y_i's can be probabilities, but in most applications they are all 0 or 1.
- QDA, LDA: generative models.
- Logistic regression: discriminative model.
- Optimization problem:

$$\boxed{\text{Find } w, \alpha \text{ that minimizes } J = \sum_{i=1}^{n} (y_i \ln s(X_i \cdot w + \alpha) + (1 - y_i) \ln(1 - s(X_i \cdot w + \alpha))}$$



- $-J(w, \alpha)$ is convex. Solve by gradient ascent.

$$
\begin{aligned}
s'(\gamma) &= \frac{d}{d\gamma} \frac{1}{1+e^{-\gamma}} = \frac{e^{-\gamma}}{(1+e^{-\gamma})^2} \\
&= s(\gamma)(1 - s(\gamma))
\end{aligned}
$$



- Let $s_i = s(X_i \cdot w + \alpha)$,

$$
\begin{aligned}
\nabla_w J &= \sum \Big(\frac{y_i}{s_i} \nabla s_i - \frac{1-y_i}{1-s_i} \nabla s_i \Big) \\
&= \sum \Big(\frac{y_i}{s_i} - \frac{1-y_i}{1-s_i} \Big) s_i (1-s_i) X_i \\
&= \sum (y_i - s_i) X_i
\end{aligned}
$$

- Gradient ascent rule: $w \leftarrow w + \epsilon \sum_{i=1}^{n} (y_i - s(X_i \cdot w + \alpha)) X_i$
- Stochastic gradient ascent: $w \leftarrow w + \epsilon (y_i - s(X_i \cdot w + \alpha)) X_i$ works best if we shuffle samples in random order; process one by one.
- For very large n, sometimes converges before we visit all samples!
- Starting from $w = 0, \alpha = 0$ works well in practice.