Introduction to Machine Learning - CS725
Instructor: Prof. Ganesh Ramakrishnan
Lecture 4 - Linear Regression - Bayesian Inference and Regularization

Building on questions on Least Squares Linear Regression

1. Is there a probabilistic interpretation?
   - Gaussian Error, Maximum Likelihood Estimate
2. Addressing overfitting
   - Bayesian and Maximum Aposteriori Estimates, Regularization
3. How to minimize the resultant and more complex error functions?
   - Level Curves and Surfaces, Gradient Vector, Directional Derivative, Gradient Descent Algorithm, Convexity, Necessary and Sufficient Conditions for Optimality

Prior Distribution over **w** for Linear Regression

$$y = \mathbf{w}^T \phi(x) + \varepsilon$$
$$\varepsilon \sim \mathcal{N}(0, \sigma^2)$$

- We saw that when we try to maximize log-likelihood we end up with $\hat{\mathbf{w}}_{MLE} = (\phi^T \phi)^{-1} \phi^T \mathbf{y}$
- We can use a Prior distribution on **w** to avoid over-fitting
$$w_i \sim \mathcal{N}(0, \tfrac{1}{\lambda})$$
(that is, each component w_i is approximately bounded within $\pm \frac{3}{\sqrt{\lambda}}$ by the $3 - \sigma$ rule)
- We want to find $P(\mathbf{w}|D) = \mathcal{N}(\mu_m, \Sigma_m)$
Invoking the Bayes Estimation results from before:

Prior Distribution over **w** for Linear Regression

$$y = \mathbf{w}^T \phi(x) + \varepsilon$$
$$\varepsilon \sim \mathcal{N}(0, \sigma^2)$$

- We saw that when we try to maximize log-likelihood we end up with $\hat{\mathbf{w}}_{MLE} = (\phi^T \phi)^{-1} \phi^T \mathbf{y}$
- We can use a Prior distribution on **w** to avoid over-fitting
$$w_i \sim \mathcal{N}(0, \frac{1}{\lambda})$$
(that is, each component w_i is approximately bounded within $\pm \frac{3}{\sqrt{\lambda}}$ by the $3 - \sigma$ rule)
- We want to find $P(\mathbf{w}|D) = \mathcal{N}(\mu_m, \Sigma_m)$
Invoking the Bayes Estimation results from before:

$$\Sigma_m^{-1} \mu_m = \Sigma_0^{-1} \mu_0 + \phi^T \mathbf{y} / \sigma^2$$
$$\Sigma_m^{-1} = \Sigma_0^{-1} + \frac{1}{\sigma^2} \phi^T \phi$$

Finding μ_m & Σ_m for $\mathbf{w}$

Setting $\Sigma_0 = \frac{1}{\lambda} I$ and $\mu_0 = \mathbf{0}$

$$\begin{aligned} \Sigma_m^{-1} \mu_m &= \phi^T \mathbf{y} / \sigma^2 \\ \Sigma_m^{-1} &= \lambda I + \phi^T \phi / \sigma^2 \\ \mu_m &= \frac{(\lambda I + \phi^T \phi / \sigma^2)^{-1} \phi^T \mathbf{y}}{\sigma^2} \end{aligned}$$

or

$$\mu_m = (\lambda \sigma^2 I + \phi^T \phi)^{-1} \phi^T \mathbf{y}$$

MAP and Bayes Estimates

- $\Pr(\mathbf{w} \mid \mathcal{D}) = \mathcal{N}(\mathbf{w} \mid \mu_m, \Sigma_m)$
- The **MAP estimate** or mode under the Gaussian posterior is the mode of the posterior $\Rightarrow$

$$\hat{w}_{MAP} = \underset{\mathbf{w}}{\mathrm{argmax}}\, \mathcal{N}(\mathbf{w} \mid \mu_m, \Sigma_m) = \mu_m$$

- Similarly, the **Bayes Estimate**, or the expected value under the Gaussian posterior is the mean $\Rightarrow$

$$\hat{w}_{Bayes} = E_{\Pr(\mathbf{w}|\mathcal{D})}[\mathbf{w}] = E_{\mathcal{N}(\mu_m, \Sigma_m)}[\mathbf{w}] = \mu_m$$

- Summarily:

$$\mu_{MAP} = \mu_{Bayes} = \mu_m = (\lambda\sigma^2 I + \phi^T\phi)^{-1}\phi^T\mathbf{y}$$

$$\Sigma_m^{-1} = \lambda I + \frac{\phi^T\phi}{\sigma^2}$$

From Bayesian Estimates to (Pure) Bayesian Prediction

	Point?	$p(x\|D)$
MLE	$\hat{\theta}_{MLE} = \text{argmax}_\theta \, LL(D\|\theta)$	$p(x\|\theta_{MLE})$
Bayes Estimator	$\hat{\theta}_B = E_{p(\theta\|D)} E[\theta]$	$p(x\|\theta_B)$
MAP	$\hat{\theta}_{MAP} = \text{argmax}_\theta \, p(\theta\|D)$	$p(x\|\theta_{MAP})$
Pure Bayesian		$p(\theta\|D) = \frac{p(D\|\theta)p(\theta)}{\int_m p(D\|\theta)p(\theta)d}$ $p(D\|\theta) = \prod_{i=1} p(x_i\|\theta)$ $p(x\|D) = \int_\theta p(x\|\theta)p(\theta\|D$

where θ is the parameter

Predictive distribution for linear Regression

- $\hat{\mathbf{w}}_{MAP}$ helps avoid overfitting as it takes regularization into account
- But we miss the modeling of uncertainty when we consider only $\hat{\mathbf{w}}_{MAP}$
- **Eg:** While predicting diagnostic results on a new patient x, along with the value y, we would also like to know the uncertainty of the prediction $\Pr(y \mid x, D)$. Recall that $y = \mathbf{w}^T \phi(x) + \varepsilon$ and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$

$$\Pr(y \mid \mathbf{x}, \mathcal{D}) = \Pr(y \mid \mathbf{x}, < \mathbf{x}_1, y_1 > \ldots < \mathbf{x}_m, y_m >)$$

Pure Bayesian Regression Summarized

- By definition, regression is about finding $(y \mid \mathbf{x}, < \mathbf{x}_1, y_1 > \ldots < \mathbf{x}_m, y_m >)$
- By Bayes Rule

$$
\begin{aligned}
\Pr(y \mid \mathbf{x}, \mathcal{D}) &= \Pr(y \mid \mathbf{x}, < \mathbf{x}_1, y_1 > \ldots < \mathbf{x}_m, y_m \\
&= \int_{\mathbf{w}} \Pr(y|\mathbf{w}; \mathbf{x}) \Pr(\mathbf{w} \mid \mathcal{D}) d\mathbf{w} \\
&\sim \mathcal{N}\left(\mu_m^T \phi(\mathbf{x}), \sigma^2 + \phi^T(\mathbf{x}) \Sigma_m \phi(\mathbf{x})\right)
\end{aligned}
$$

where

$$y = \mathbf{w}^T \phi(\mathbf{x}) + \varepsilon \text{ and } \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

$$\mathbf{w} \sim \mathcal{N}(0, \alpha I) \text{ and } \mathbf{w} \mid \mathcal{D} \sim \mathcal{N}(\mu_m, \Sigma_m)$$

$$\mu_m = (\lambda \sigma^2 I + \phi^T \phi)^{-1} \phi^T \mathbf{y} \text{ and } \Sigma_m^{-1} = \lambda I + \phi^T \phi / \sigma^2$$

$$\text{Finally} \quad y \sim \mathcal{N}(\mu_m^T \phi(\mathbf{x}), \phi^T(\mathbf{x}) \Sigma_m \phi(\mathbf{x}))$$

Penalized Regularized Least Squares Regression

- The Bayes and MAP estimates for Linear Regression coincide with *Regularized Ridge Regression*

$$\mathbf{w}_{Ridge} = \arg\min_{\mathbf{w}} \; ||\phi\mathbf{w} - \mathbf{y}||_2^2 + \lambda\sigma^2||\mathbf{w}||_2^2$$

- **Intuition:** To discourage redundancy and/or stop coefficients of $\mathbf{w}$ from becoming too large in magnitude, add a penalty to the error term used to estimate parameters of the model.
- The general **Penalized Regularized L.S Problem**:

$$\mathbf{w}_{Reg} = \arg\min_{\mathbf{w}} \; ||\phi\mathbf{w} - \mathbf{y}||_2^2 + \lambda\Omega(\mathbf{w})$$

  - $\Omega(\mathbf{w}) = ||\mathbf{w}||_2^2 \Rightarrow$ **Ridge Regression**
  - $\Omega(\mathbf{w}) = ||\mathbf{w}||_1 \Rightarrow$ **Lasso**
  - $\Omega(\mathbf{w}) = ||\mathbf{w}||_0 \Rightarrow$ **Support-based penalty**
- Some $\Omega(\mathbf{w})$ correspond to priors that can be expressed in close form. Some give good working solutions. However, for mathematical convenience, some norms are easier to handle

Constrained Regularized Least Squares Regression

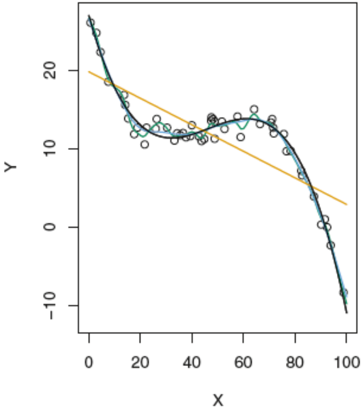
- **Intuition:** To discourage redundancy and/or stop coefficients of $\mathbf{w}$ from becoming too large in magnitude, constrain the error minimizing estimate using a penalty
- The general Constrained **Regularized L.S. Problem**:

$$\mathbf{w}_{Reg} = \arg\min_{\mathbf{w}} \; ||\phi\mathbf{w} - \mathbf{y}||_2^2$$

$$\textit{such that } \Omega(\mathbf{w}) \leq \theta$$

- Claim: For any Penalized formulation with a particular λ, there exists a corresponding Constrained formulation with a corresponding θ
  - $\Omega(\mathbf{w}) = ||\mathbf{w}||_2^2 \Rightarrow$ **Ridge Regression**
  - $\Omega(\mathbf{w}) = ||\mathbf{w}||_1 \Rightarrow$ **Lasso**
  - $\Omega(\mathbf{w}) = ||\mathbf{w}||_0 \Rightarrow$ **Support-based penalty**
- **Proof of Equivalence:** Requires tools of Optimization/duality

Polynomial regression



- Consider a degree 3 polynomial regression model as shown in the figure
- Each bend in the curve corresponds to increase in $\|w\|$
- Eigen values of $(\phi^\top \phi + \lambda I)$ are indicative of curvature. Increasing λ reduces the curvature

Do Closed-form solutions Always Exist?

- Linear regression and Ridge regression both have closed-form solutions
  - For linear regression,

$$w^* = (\phi^\top \phi)^{-1} \phi^\top y$$

  - For ridge regression,

$$w^* = (\phi^\top \phi + \lambda I)^{-1} \phi^\top y$$

  (for linear regression, $\lambda = 0$)

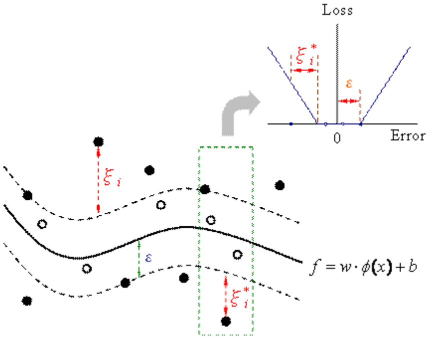
- What about optimizing the formulations (constrained/penalized) of Lasso (L_1 norm)? And support-based penalty (L_0 norm)?: Also requires tools of Optimization/duality

Why is Lasso Interesting?

Support Vector Regression

One more formulation before we look at Tools of Optimization/duality

Support Vector Regression (SVR)



- Any point in the band (of ϵ) is not penalized. Thus the loss function is known as ϵ-*insensitive loss*
- Any point outside the band is penalized, and has slackness ξ_i or ξ_i^*
- The SVR model curve may not pass through any training point

- The tolerance ϵ is fixed
- It is desirable that $\forall i$:
  - $y_i - w^\top \phi(x_i) - b \leq \epsilon + \xi_i$
  - $b + w^\top \phi(x_i) - y_i \leq \epsilon + \xi_i^*$

SVR objective

- 1-norm regularized:
  - $\min_{w,b,\xi_i,\xi_i^*} \frac{1}{2}\|w\|^2 + C\sum_i(\xi_i + \xi_i^*)$
    s.t. $\forall i$,
    $y_i - w^\top\phi(x_i) - b \leq \epsilon + \xi_i$,
    $b + w^\top\phi(x_i) - y_i \leq \epsilon + \xi_i^*$,
    $\xi_i, \xi_i^* \geq 0$
- 2-norm regularized:
  - $\min_{w,b,\xi_i,\xi_i^*} \frac{1}{2}\|w\|^2 + C\sum_i(\xi_i^2 + \xi_i^{*2})$
    s.t. $\forall i$,
    $y_i - w^\top\phi(x_i) - b \leq \epsilon + \xi_i$,
    $b + w^\top\phi(x_i) - y_i \leq \epsilon + \xi_i^*$
  - Here, the constraints $\xi_i, \xi_i^* \geq 0$ are not necessary

- *Claim:*
  Error obtained on training data after minimizing ridge regression $\geq$ error obtained on training data after minimizing linear regression
- *Goal:*
  Do well on unseen (test) data as well. Therefore, high training error might be acceptable if test error can be lower

Solving Least Square Linear Regression Model

- Intuitively: Minimize by setting derivative (gradient) to 0 and find closed form solution.
- For most optimization problems, finding closed form solution is difficult
- Even for linear regression (for which closed form solution exists), are there alternative methods?
- Eg: Consider, $\mathbf{y} = \phi\mathbf{w}$,where ϕ is a matrix with full column rank, the least squares solution, $\mathbf{w}^* = (\phi^T\phi)^{-1}\phi^T\mathbf{y}$. Now, imagine that ϕ is a very large matrix. with say, 100,000 columns and 1,000,000 rows. Computation of closed form solution might be challenging.
- How about an iterative method?

- A level curve of a function $\mathbf{f}(\mathbf{x})$ is defined as a curve along which the value of the function remains unchanged while we change the value of its argument x.
- Formally we can define a level curve as :

$$L_c(\mathbf{f}) = \left\{ \mathbf{x} | \mathbf{f}(\mathbf{x}) = \mathbf{c} \right\} \tag{1}$$

where c is a constant.

Level curves and surfaces

- The image below is an example of different level curves for a single function

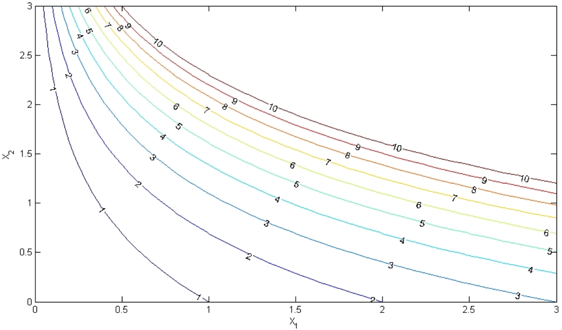


Figure 1: 10 level curves for the function $\mathbf{f}(\mathbf{x_1}, \mathbf{x_2}) = \mathbf{x_1 e_2^x}$ (Figure 4.12 from `https://www.cse.iitb.ac.in/~cs709/notes/BasicsOfConvexOptimization.pdf`)

Directional Derivatives

- Directional derivative: Rate at which the function changes at a given point in a given direction
- The *directional derivative* of a function f in the direction of a unit vector $\mathbf{v}$ at a point $\mathbf{x}$ can be defined as :

$$D_{\mathbf{v}}(f) = \lim_{h \to 0} \frac{f(\mathbf{x} + \mathbf{hv}) - \mathbf{f}(\mathbf{x})}{h} \tag{2}$$

$$||\mathbf{v}|| = \mathbf{1} \tag{3}$$

Gradient Vector

- Magnitude (euclidean norm) of gradient vector at any point indicates maximum value of directional derivative at that point
- Direction of gradient vector indicates direction of this maximal directional derivative at that point.
- The *gradient vector* of a function f at a point $\mathbf{x}$ is defined as:

$$\nabla f_{\mathbf{x}^*} = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \frac{\partial f(\mathbf{x})}{\partial x_2} \\ . \\ . \\ \frac{\partial f(\mathbf{x})}{\partial x_n} \end{bmatrix} \epsilon \mathbb{R}^n \tag{4}$$

Gradient Vector

- Magnitude (euclidean norm) of gradient vector at any point indicates maximum value of directional derivative at that point
- The *gradient vector* of a function f at a point $\mathbf{x}$ is defined as:

$$\nabla f_{\mathbf{x}^*} = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \frac{\partial f(\mathbf{x})}{\partial x_2} \\ . \\ . \\ \frac{\partial f(\mathbf{x})}{\partial x_n} \end{bmatrix} \epsilon \mathbb{R}^n \tag{5}$$

- Thus, at the point of minimum of a differentiable minimization objective (such as least squares for regression), ...

Gradient Vector

- The figure below gives an example of gradient vector

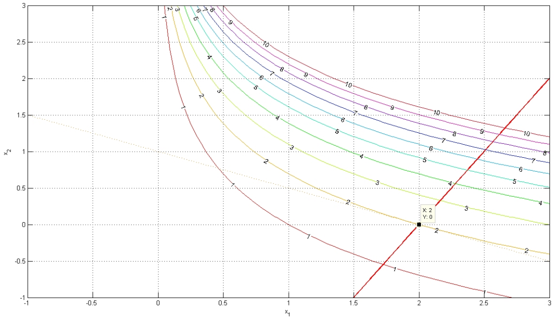


Figure 2: The level curves from Figure 1 along with the gradient vector at (2, 0). Note that the gradient vector is perpenducular to the level curve $x_1 e^{x_2} = 2$ at (2, 0)

Hyperplanes

- A hyperplane in an n-dimensional Euclidean space is a flat, n-1 dimensional subset of that space that divides the space into two disconnected parts.
- Technically, a hyperplane is a set of points whose direction *w.r.t.* a point $\mathbf{p}$ is orthogonal to a vector $\mathbf{v}$.
- Formally:

$$H_{\mathbf{v},\mathbf{p}} = \left\{ \mathbf{q} \mid (\mathbf{p} - \mathbf{q})^{\mathbf{T}} \mathbf{v} = \mathbf{0} \right\} \tag{6}$$

Tangential Hyperplanes

There are two definitions of *tangential hyperplane* ($TH_{\mathbf{x}^*}$) to *level surface* ($L_{f(\mathbf{x}^*)}(f)$) of f at $\mathbf{x}^*$:

- Plane consisting of all tangent lines at $\mathbf{x}^*$ to any parametric curve $c(t)$ on level surface.
- Plane orthogonal to the gradient vector at $\mathbf{x}^*$.

$$TH_{\mathbf{x}^*} = \left\{ \mathbf{p} \mid (\mathbf{p} - \mathbf{x}^*)^{\mathbf{T}} \nabla \mathbf{f}(\mathbf{x}^*) = \mathbf{0} \right\} \tag{7}$$

Gradient Descent Algorithm

Gradient descent is based on the observation that if the multi-variable function F(**x**) is defined and differentiable in a neighborhood of a point **a** , then F(**x**) decreases fastest if one goes from **a** in the direction of the negative gradient of F at **a** ,i.e. -∇ F(**a**).
Therefore,

$$\Delta \mathbf{w}^{(\mathbf{k})} = - \nabla \mathbf{E}(\mathbf{w}^{(\mathbf{k})}) \tag{8}$$

Hence,

$$\mathbf{w}^{(\mathbf{k+1})} = \mathbf{w}^{(\mathbf{k})} + \mathbf{2t}^{(\mathbf{k})}(\phi^{\mathbf{T}}\mathbf{y} - \phi^{\mathbf{T}}\phi\mathbf{w}^{(\mathbf{k})}) \tag{9}$$

Gradient Descent Algorithm

Find starting point $\mathbf{w^{(0)}} \epsilon \mathcal{D}$

- $\Delta \mathbf{w^k} = -\nabla \varepsilon(\mathbf{w^{(k)}})$
- Choose a step size $t^{(k)} > 0$ using exact or backtracking ray search.
- Obtain $\mathbf{w^{(k+1)}} = \mathbf{w^{(k)}} + \mathbf{t^{(k)}} \mathbf{\Delta w^{(k)}}$.
- Set $k = k + 1$. **until** stopping criterion (such as $\|\nabla \varepsilon(\mathbf{x^{(k+1)}})\| \leq \epsilon$) is satisfied

Gradient Descent Algorithm

Exact line search algorithm to find $t^{(k)}$

- The line search approach first finds a descent direction along which the objective function f will be reduced and then computes a step size that determines how far **x** should move along that direction.
- In general,

$$t^{(k)} = \arg\min_t f\left(\mathbf{w^{(k+1)}}\right) \tag{10}$$

- Thus,

$$t^{(k)} = \arg\min_t \left(\mathbf{w^{(k)}} + \mathbf{2t}\left(\phi^{\mathbf{T}}\mathbf{y} - \phi^{\mathbf{T}}\phi\mathbf{w^{(k)}}\right)\right) \tag{11}$$

Example of Gradient Descent Algorithm

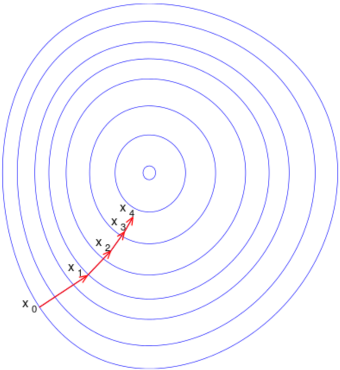


Figure 3: A red arrow originating at a point shows the direction of the negative gradient at that point. Note that the (negative) gradient at a point is orthogonal to the level curve going through that point. We see that gradient descent leads us to the bottom of the bowl, that is, to the point where the value of the function F is minimal. Sources: Wikipidea