Linear Models for Regression

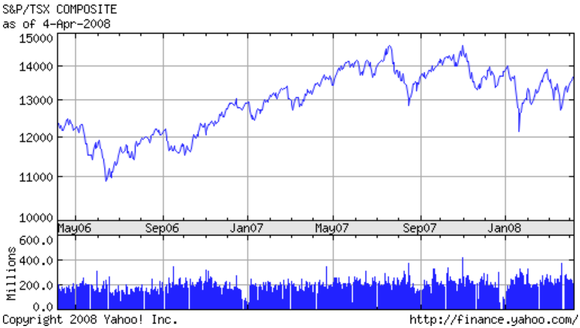
CMPT 419/726
Mo Chen
SFU Computing Science
Jan. 13, 2020

Bishop PRML Ch. 3

Outline

Outline

Regression



- Given training set $\{(\boldsymbol{x}_1, \boldsymbol{t}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{t}_N)\}$
- $\boldsymbol{t}_i$ is continuous: regression
- For now, assume $\boldsymbol{t}_i \in \mathbb{R}, \boldsymbol{x}_i \in \mathbb{R}^D$
- E.g. $\boldsymbol{t}_i$ is stock price, $\boldsymbol{x}_i$ contains company profit, debt, cash flow, gross sales, number of spam emails sent, …

Outline

Linear Functions

- A function $f(\cdot)$ is linear if

$$f(\alpha u + \beta v) = \alpha f(u) + \beta f(v)$$

- Linear functions will lead to simple algorithms, so let's see what we can do with them

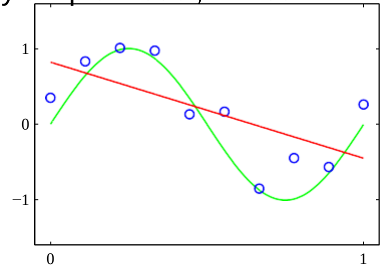
Linear Regression

- Simplest linear model for regression

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D$$

  - Remember, we're learning $\boldsymbol{w}$
  - Set $\boldsymbol{w}$ so that $y(\boldsymbol{x}, \boldsymbol{w})$ aligns with target value in training data

- This is a very simple model, limited in what it can do



Linear Basis Function Models

- Simplest linear model

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D$$

was linear in $\boldsymbol{x}$ (*) and $\boldsymbol{w}$

- Linearity in $\boldsymbol{w}$ is what will be important for simple algorithms
- Extend to include fixed non-linear functions of data

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 \phi_1(\boldsymbol{x}) + w_2 \phi_2(\boldsymbol{x}) + \cdots + w_{M-1} \phi_{M-1}(\boldsymbol{x})$$

- Linear combinations of these basis functions also linear in parameters

Linear Basis Function Models

- Bias parameter allows fixed offset in data

$$y(\boldsymbol{x}, \boldsymbol{w}) = \underbrace{w_0}_{\text{bias}} + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D$$

  - Think of simple 1-D $\boldsymbol{x}$:

$$y(\boldsymbol{x}, \boldsymbol{w}) = \underbrace{w_0}_{\text{intercept}} + \underbrace{w_1}_{\text{slope}} x_1$$

For notational convenience, define $\phi_0(\boldsymbol{x}) = 1$:

$$y(\boldsymbol{x}, \boldsymbol{w}) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^\top \boldsymbol{\phi}(\boldsymbol{x})$$

Linear Basis Function Models

- Function for regression $y(\boldsymbol{x}, \boldsymbol{w})$ is non-linear function of $\boldsymbol{x}$, but linear in $\boldsymbol{w}$:

$$y(\boldsymbol{x}, \boldsymbol{w}) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^\top \boldsymbol{\phi}(\boldsymbol{x})$$

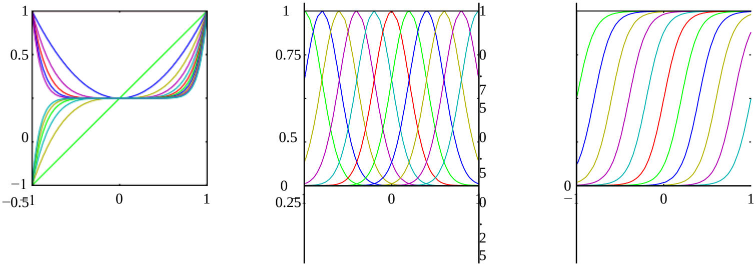
- Polynomial regression is an example of this
- Order M polynomial regression, $\phi_j(x) =$?
- $\phi_j(x) = x^j$:

$$y(x, \boldsymbol{w}) = w_0 x^0 + w_1 x^1 + \cdots + w_M x^M$$

Basis Functions: Feature Functions

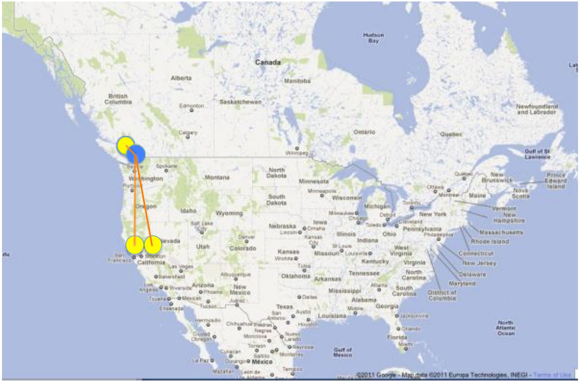
- Often we extract features from $\boldsymbol{x}$
  - An intuitve way to think of $\phi_j(\boldsymbol{x})$ is as feature functions
- E.g. Automatic CMPT 726 project report grading system
  - $\boldsymbol{x}$ is text of report: `In this project we apply the algorithm of Mori [2] to recognizing blue objects. We test this algorithm on pictures of you and I from my holiday photo collection. ...`
- $\phi_1(\boldsymbol{x})$ is count of occurrences of `Mori [`
- $\phi_2(\boldsymbol{x})$ is count of occurrences of `of you and I`
- Regression grade $y(\boldsymbol{x}, \boldsymbol{w}) = 20\phi_1(\boldsymbol{x}) - 10\phi_2(\boldsymbol{x})$

Other Non-linear Basis Functions



- Polynomial: $\phi_j(x) = x^j$
- Gaussians: $\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$
- Sigmoidal: $\phi_j(x) = \frac{1}{1+\exp\left\{\frac{\mu_j - x}{s}\right\}}$

Example - Gaussian Basis Functions: Temperature

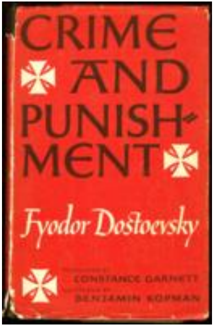


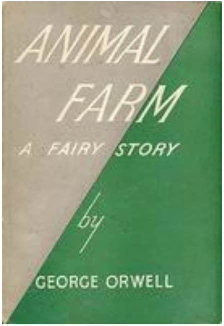
- $\mu_1 =$ Vancouver, $\mu_2 =$ San Francisco, $\mu_3 =$ Oakland
- Temperature in $x =$ Seattle?

$$y(x, w) = w_1 \exp\left\{-\frac{(x-\mu_1)^2}{2s^2}\right\} + w_2 \exp\left\{-\frac{(x-\mu_2)^2}{2s^2}\right\} + w_3 \exp\left\{-\frac{(x-\mu_3)^2}{2s^2}\right\}$$

- Compute distances to all μ, $y(x, w) \approx w_1$

Example - Gaussian Basis Functions: 726 Report Grading





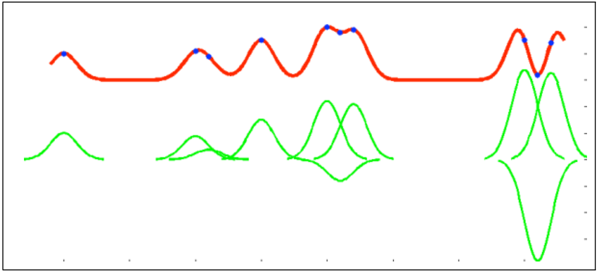
- Define:
  - μ_1 = Crime and Punishment
  - μ_2 = Animal Farm
  - μ_3 = Some paper by Mori
- Learn weights:
  - $w_1 = ?$
  - $w_2 = ?$
  - $w_3 = ?$
- Grade a project report $\boldsymbol{x}$:
  - Measure similarity of $\boldsymbol{x}$ to each μ_j, Gaussian, with weights:

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_1 \exp\left\{-\frac{(x-\mu_1)^2}{2s^2}\right\} + w_2 \exp\left\{-\frac{(x-\mu_2)^2}{2s^2}\right\} + w_3 \exp\left\{-\frac{(x-\mu_3)^2}{2s^2}\right\}$$

- The Gaussian basis function models end up similar to template matching

Example - Gaussian Basis Functions

- Could define $\exp\left\{-\frac{(x-\mu_1)^2}{2s^2}\right\}$
  - Gaussian around each training data point x_j, N of them
- Could use for modelling temperature or resource availability at spatial location x
- Overfitting - interpolates data
- Example of a kernel method



Outline

Loss Functions for Regression

- We want to find the "best" set of coefficients $\boldsymbol{w}$
- Recall, one way to define "best" was minimizing squared error:

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \boldsymbol{w}) - t_n\}^2$$

- We will now look at another way, based on maximum likelihood

Gaussian Noise Model for Regression

- We are provided with a training set $\{(\boldsymbol{x}_i, t_i)\}$
- Let's assume t arises from a deterministic function plus Gassian distributed (with precision β) noise:

$$t = y(\boldsymbol{x}, \boldsymbol{w}) + \epsilon$$

- The probability of observing a target value t is then:

$$p(t|\boldsymbol{x}, \boldsymbol{w}, \beta) = \mathcal{N}(t|y(\boldsymbol{x}, \boldsymbol{w}), \beta^{-1})$$

  - Notation: $\mathcal{N}(x|\mu, \sigma^2)$; x drawn from Gaussian with mean μ, variance σ^2

Gaussian Noise Model for Regression

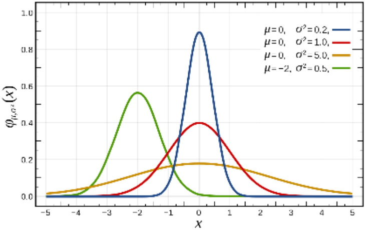
- The probability of observing a target value t is then:

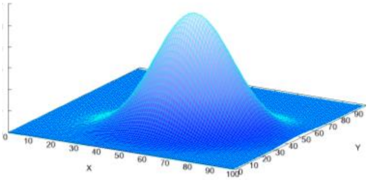
$$p(t|\boldsymbol{x}, \boldsymbol{w}, \beta) = \mathcal{N}(t|y(\boldsymbol{x}, \boldsymbol{w}), \beta^{-1})$$

  - Notation: $\mathcal{N}(x|\mu, \sigma^2)$;x drawn from Gaussian with mean μ, variance σ^2

- If $x \sim \mathcal{N}(x|\mu, \sigma^2)$, then

$$p(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$





Maximum Likelihood for Regression

- The likelihood of data $t = \{t_i\}$ using this Gaussian noise model:

$$p(\boldsymbol{t}|\boldsymbol{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_n), \beta^{-1})$$

- The log-likelihood:

$$\ln p(\boldsymbol{t}|\boldsymbol{w}, \beta) = \ln \prod_{n=1}^{N} \frac{\sqrt{\beta}}{\sqrt{2\pi}} \exp\left\{-\frac{\beta}{2}\left(t_n - \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_n)\right)^2\right\}$$

$$= \underbrace{\frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi)}_{\text{constant w.r.t. } \boldsymbol{w}} - \beta \underbrace{\frac{1}{2}\sum_{n=1}^{N}\left(t_n - \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_n)\right)^2}_{\text{squared error}}$$

- Sum of squared errors is maximum likelihood under a Gaussian noise model

Announcements

Project examples

Project timeline

Outline

Finding Optimal Weights

$$\ln p(\boldsymbol{t}|\boldsymbol{w},\beta) = \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta\frac{1}{2}\sum_{n=1}^{N}\left(t_n - \boldsymbol{w}^\top\boldsymbol{\phi}(\boldsymbol{x}_n)\right)^2$$

- How do we maximize likelihood wrt $\boldsymbol{w}$ (or minimize squared error)?
- Take gradient of log-likelihood wrt $\boldsymbol{w}$:

$$\frac{\partial}{\partial w_i}\ln p(\boldsymbol{t}|\boldsymbol{w},\beta) = \beta\sum_{n=1}^{N}\left(t_n - \boldsymbol{w}^\top\boldsymbol{\phi}(\boldsymbol{x}_n)\right)\phi_i(\boldsymbol{x}_n)$$

- In vector form:

$$\nabla\ln p(\boldsymbol{t}|\boldsymbol{w},\beta) = \beta\sum_{n=1}^{N}\left(t_n - \boldsymbol{w}^\top\boldsymbol{\phi}(\boldsymbol{x}_n)\right)\boldsymbol{\phi}(\boldsymbol{x}_n)^\top$$

Finding Optimal Weights

- Set gradient to 0:

$$\mathbf{0}^\top = \sum_{n=1}^{N} t_n \boldsymbol{\phi}(\boldsymbol{x}_n)^\top - \boldsymbol{w}^\top \sum_{n=1}^{N} \boldsymbol{\phi}(\boldsymbol{x}_n)\boldsymbol{\phi}(\boldsymbol{x}_n)^\top$$

- Maximum likelihood estimate for $\boldsymbol{w}$:

$$\boldsymbol{w}_{ML} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{t}$$

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_1) & \cdots & \phi_{M-1}(\boldsymbol{x}_1) \\ \phi_0(\boldsymbol{x}_2) & \phi_1(\boldsymbol{x}_2) & \cdots & \phi_{M-1}(\boldsymbol{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\boldsymbol{x}_N) & \phi_1(\boldsymbol{x}_N) & \cdots & \phi_{M-1}(\boldsymbol{x}_N) \end{pmatrix}$$

- $\boldsymbol{\Phi}^\dagger = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top$ is known as the pseudo-inverse (`numpy.linalg.pinv` in python)

Math

$$y(\boldsymbol{x}, \boldsymbol{w}) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^\top \boldsymbol{\phi}(\boldsymbol{x})$$

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_1) & \cdots & \phi_{M-1}(\boldsymbol{x}_1) \\ \phi_0(\boldsymbol{x}_2) & \phi_1(\boldsymbol{x}_2) & \cdots & \phi_{M-1}(\boldsymbol{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\boldsymbol{x}_N) & \phi_1(\boldsymbol{x}_N) & \cdots & \phi_{M-1}(\boldsymbol{x}_N) \end{pmatrix}$$

$$\boldsymbol{0}^\top = \sum_{n=1}^{N} t_n \boldsymbol{\phi}(\boldsymbol{x}_n)^\top - \boldsymbol{w}^\top \sum_{n=1}^{N} \boldsymbol{\phi}(\boldsymbol{x}_n)\boldsymbol{\phi}(\boldsymbol{x}_n)^\top$$

$$\boldsymbol{0}^\top = \boldsymbol{t}^\top \begin{bmatrix} \boldsymbol{\phi}(x_1)^\top \\ \vdots \\ \boldsymbol{\phi}(x_N)^\top \end{bmatrix} - \boldsymbol{w}^\top [\boldsymbol{\phi}(\boldsymbol{x}_1) \quad \cdots \quad \boldsymbol{\phi}(\boldsymbol{x}_N)] \begin{bmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1)^\top \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_N)^\top \end{bmatrix}$$

(Sum → dot product)

$$\boldsymbol{0}^\top = \boldsymbol{t}^\top \boldsymbol{\Phi} - \boldsymbol{w}^\top \boldsymbol{\Phi}^\top \boldsymbol{\Phi}$$

(Matrix form)

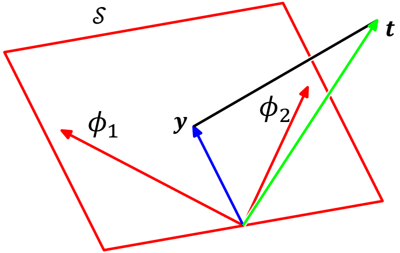
$$\boldsymbol{0} = \boldsymbol{\Phi}^\top \boldsymbol{t} - \boldsymbol{\Phi}^\top \boldsymbol{\Phi} \boldsymbol{w}$$

(Transpose, $(AB)^\top = B^\top A^\top$)

$$\boldsymbol{\Phi}^\top \boldsymbol{\Phi} \boldsymbol{w} = \boldsymbol{\Phi}^\top \boldsymbol{t} \Rightarrow \boldsymbol{w} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{t}$$

(Rearrange and take inverse)

Geometry of Least Squares



- $\boldsymbol{t} = (\boldsymbol{t_1}, \dots, \boldsymbol{t_N})$ is the target value vector
- $\mathcal{S}$ is space spanned by $\boldsymbol{\phi}_j = \left(\phi_j(\boldsymbol{x}_1), \dots, \phi_j(\boldsymbol{x}_N)\right)$
- Solution $\boldsymbol{y}$ lies in $\mathcal{S}$
- Least squares solution is orthogonal projection of $\boldsymbol{t}$ onto $\mathcal{S}$
- Can verify this by looking at $\boldsymbol{y} = \boldsymbol{\Phi w}_{ML} = \boldsymbol{\Phi\Phi}^{\dagger}\boldsymbol{t} = \boldsymbol{Pt}$
  - $\boldsymbol{P}^{\boldsymbol{2}} = \boldsymbol{P}, \boldsymbol{P} = \boldsymbol{P}^{\top}$

Math

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{w}_{ML}, \text{ where } \boldsymbol{w}_{ML} = (\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\boldsymbol{t}$$

$$\boldsymbol{y} = \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\boldsymbol{t} = \boldsymbol{P}\boldsymbol{t}, \text{ where } \boldsymbol{P} = \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top$$

verify $\boldsymbol{P}^2 = \boldsymbol{P}$

$$\boldsymbol{P}^2 = \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top$$

$$= \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top$$

$$= \boldsymbol{P}$$

Math

$$\boldsymbol{y} = \boldsymbol{\Phi} \boldsymbol{w}_{ML}, \text{ where } \boldsymbol{w}_{ML} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{t}$$

$$\boldsymbol{y} = \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{t} = \boldsymbol{P}\boldsymbol{t}, \text{ where } \boldsymbol{P} = \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top$$

verify $\boldsymbol{P} = \boldsymbol{P}^\top$

$$\boldsymbol{P}^\top = (\boldsymbol{\Phi}(\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top)^\top$$

$$= \boldsymbol{\Phi}\left((\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1}\right)^\top \boldsymbol{\Phi}^\top \qquad \text{(Transpose, } (AB)^\top = B^\top A^\top\text{)}$$

$$= \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top$$

$$\left(\left(\boldsymbol{A}^{-1}\right)^\top = (\boldsymbol{A}^\top)^{-1}, \text{ since } \left(\boldsymbol{A}^{-1}\right)^\top \boldsymbol{A}^\top = \left(\boldsymbol{A}\boldsymbol{A}^{-1}\right)^\top = \boldsymbol{I}\right)$$

Sequential Learning

- In practice N might be huge, or data might arrive online
- Can use a gradient descent method:
  - Start with initial guess for $\boldsymbol{w}$
  - Update by taking a step in gradient direction ∇E of error function
- Modify to use stochastic / sequential gradient descent:
  - If error function $E = \sum_n E_n$ (e.g. least squares)
  - Update by taking a step in gradient direction ∇E_n for one example
  - Details about step size are important – decrease step size at the end

Outline

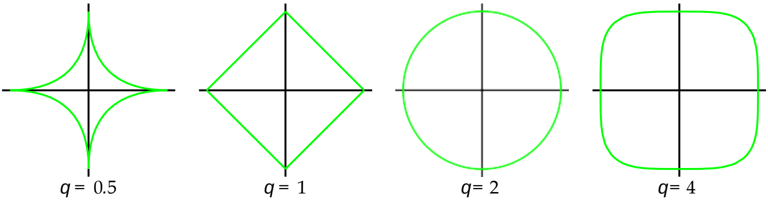
Regularization

- Last week we discussed regularization as a technique to avoid over-fitting:

$$\tilde{E}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \boldsymbol{w}) - t_n\}^2 + \underbrace{\frac{\lambda}{2}||\boldsymbol{w}||^2}_{\text{regularizer}}$$

- Next on the menu:
  - Other regularlizers
  - Bayesian learning and quadratic regularizer

Other Regularizers



- Can use different norms for regularizer:

$$\tilde{E}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \boldsymbol{w}) - t_n\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$

- e.g. $q = 2$ – ridge regression
- e.g. $q = 1$ – lasso
- math is easiest with ridge regression

Optimization with a Quadratic Regularizer

- With $q = 2$, total error still a nice quadratic:

$$\tilde{E}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \boldsymbol{w}) - t_n\}^2 + \frac{\lambda}{2}\boldsymbol{w}^\top\boldsymbol{w}$$

- Calculus ...

$$\boldsymbol{w} = \underbrace{(\lambda \boldsymbol{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})}_{\text{regularized}}{}^{-1}\boldsymbol{\Phi}^\top\boldsymbol{t}$$

- Similar to unregularlized least squares
- Advantage: $(\lambda \boldsymbol{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})$ is well conditioned so inversion is stable

Math

First, recall that without regularization,

$$\ln p(\mathbf{t}|\mathbf{w},\beta) = \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta\underbrace{\frac{1}{2}\sum_{n=1}^{N}\left(t_n - \mathbf{w}^\top\boldsymbol{\phi}(\mathbf{x}_n)\right)^2}_{E(\mathbf{w})}$$

$$\Rightarrow \mathbf{0}^\top = \sum_{n=1}^{N} t_n\boldsymbol{\phi}(\mathbf{x}_n)^\top - \mathbf{w}^\top\sum_{n=1}^{N}\boldsymbol{\phi}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n)^\top$$

Now, with regularization,

$$\tilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w}$$

$$\mathbf{0}^\top = -\sum_{n=1}^{N} t_n\boldsymbol{\phi}(\mathbf{x}_n)^\top + \mathbf{w}^\top\sum_{n=1}^{N}\boldsymbol{\phi}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n)^\top + \lambda\mathbf{w}$$

Math

Now, with regularization,

$$\tilde{E}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \boldsymbol{w}) - t_n\}^2 + \frac{\lambda}{2}\boldsymbol{w}^\top\boldsymbol{w}$$

$$\mathbf{0}^\top = -\sum_{n=1}^{N} t_n \boldsymbol{\phi}(\boldsymbol{x}_n)^\top + \boldsymbol{w}^\top \sum_{n=1}^{N} \boldsymbol{\phi}(\boldsymbol{x}_n)\boldsymbol{\phi}(\boldsymbol{x}_n)^\top + \lambda\boldsymbol{w}$$

(because why not)

$$\mathbf{0}^\top = -\boldsymbol{t}^\top \begin{bmatrix} \boldsymbol{\phi}(x_1)^\top \\ \vdots \\ \boldsymbol{\phi}(x_N)^\top \end{bmatrix} + \boldsymbol{w}^\top [\boldsymbol{\phi}(\boldsymbol{x}_1) \quad \cdots \quad \boldsymbol{\phi}(\boldsymbol{x}_N)] \begin{bmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1)^\top \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_N)^\top \end{bmatrix} + \lambda\boldsymbol{I}\boldsymbol{w}$$

(Sum → dot product)

$$\mathbf{0}^\top = -\boldsymbol{t}^\top\boldsymbol{\Phi} + \boldsymbol{w}^\top\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\boldsymbol{I}\boldsymbol{w}$$

(Matrix form)

$$\mathbf{0} = -\boldsymbol{\Phi}^\top\boldsymbol{t} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{w} + \lambda\boldsymbol{I}\boldsymbol{w}$$

(Transpose, $(AB)^\top = B^\top A^\top$)

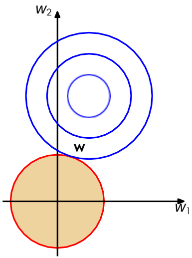
$$(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\boldsymbol{I})\boldsymbol{w} = \boldsymbol{\Phi}^\top\boldsymbol{t}$$

(Rearrange)

$$\boldsymbol{w} = (\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^\top\boldsymbol{t}$$

(Take inverse)

Ridge Regression vs. Lasso



- Ridge regression aka parameter shrinkage
  - Weights $\boldsymbol{w}$ shrink back towards origin
- Lasso leads to sparse models
  - Components of $\boldsymbol{w}$ tend to 0 with large λ (strong regularization)
  - Intuitively, once minimum achieved at large radius, minimum is on $w_1 = 0$

Outline

Bayesian Linear Regression

- Last week we saw an example of a Bayesian approach
  - Coin tossing - prior on parameters
- We will now do the same for linear regression
  - Prior on parameter $\boldsymbol{w}$
- There will turn out to be a connection to regularlization

Bayesian Linear Regression

- Start with a prior over parameters $\boldsymbol{w}$
  - Conjugate prior is a Gaussian:

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \alpha^{-1}\boldsymbol{I})$$

    - This simple form will make math easier; can be done for arbitrary Gaussian too
- Data likelihood, Gaussian model as before:

$$p(t|\boldsymbol{x}, \boldsymbol{w}, \beta) = \mathcal{N}(t|y(\boldsymbol{x}, \boldsymbol{w}), \beta^{-1})$$

Bayesian Linear Regression

- Posterior distribution on $\boldsymbol{w}$:

$$p(\boldsymbol{w}|\boldsymbol{t}) \propto \left(\prod_{n=1}^{N} p(t_n|\boldsymbol{x}_n, \boldsymbol{w}, \beta)\right) p(\boldsymbol{w})$$

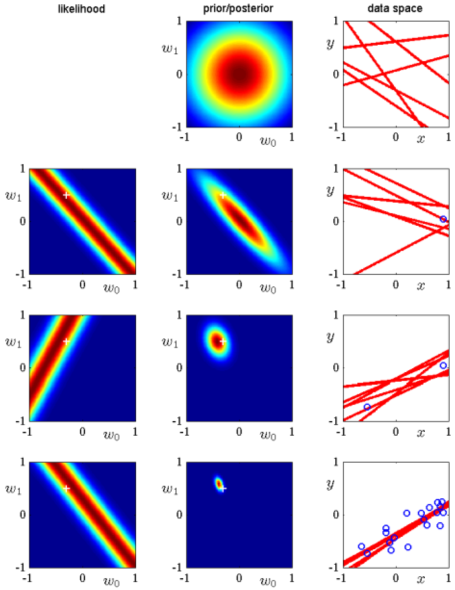
$$= \prod_{n=1}^{N} \frac{\sqrt{\beta}}{\sqrt{2\pi}} \exp\left\{-\frac{\beta}{2}\left(t_n - \boldsymbol{w}^\top \phi(\boldsymbol{x}_n)\right)^2\right\} \left(\frac{\alpha}{2\pi}\right)^{\frac{M}{2}} \exp\left\{-\frac{\alpha}{2}\boldsymbol{w}^\top \boldsymbol{w}\right\}$$

- Take the log:

$$-\ln p(\boldsymbol{w}|\boldsymbol{t}) = \frac{\beta}{2}\sum_{n=1}^{N}\left(t_n - \boldsymbol{w}^\top \phi(\boldsymbol{x}_n)\right)^2 + \frac{\alpha}{2}\boldsymbol{w}^\top \boldsymbol{w} + \text{const}$$

- L_2 regularization is maximum a posteriori (MAP) with a Gaussian prior.
  - $\lambda = \alpha / \beta$

Bayesian Linear Regression - Intuition



- Simple example $x, t \in \mathbb{R}$, $y(x, \mathbf{w}) = w_0 + w_1 x$
- Start with Gaussian prior in parameter space
- Samples shown in data space
- Receive data points (blue circles in data space)
- Compute likelihood
- Posterior is prior (or prev. posterior) times likelihood

Predictive Distribution

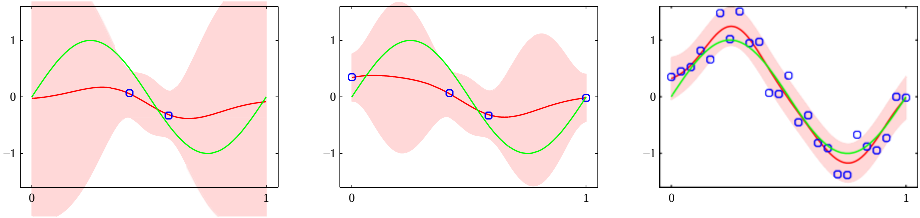
- Single estimate of $\boldsymbol{w}$ (ML or MAP) doesn't tell whole story
- We have a distribution over $\boldsymbol{w}$, and can use it to make predictions
- Given a new value for $\boldsymbol{x}$, we can compute a *distribution* over t:

$$p(t|\mathbf{t},\alpha,\beta) = \int p(t,\mathbf{w}|\mathbf{t},\alpha,\beta)d\mathbf{w}$$

$$p(t|\mathbf{t},\alpha,\beta) = \int \underbrace{p(t|\mathbf{w},\beta)}_{\text{predict}}\underbrace{p(\mathbf{w}|\mathbf{t},\alpha,\beta)}_{\text{probability}}\underbrace{d\mathbf{w}}_{\text{sum}}$$

  - i.e. For each value of $\boldsymbol{w}$, let it make a prediction, multiply by its probability, sum over all $\boldsymbol{w}$
  - For arbitrary models as the distributions, this integral may not be computationally tractable

Predictive Distribution



- With the Gaussians we've used for these distributions, the predicitve distribution will also be Gaussian
  - (math on convolutions of Gaussians spared)
- Green line is true (unobserved) curve, blue data points, red line is mean, pink one standard deviation
  - Uncertainty small around data points
  - Pink region shrinks with more data

Bayesian Model Selection

- So what do the Bayesians say about model selection?
  - Model selection is choosing model $\mathcal{M}_i$ e.g. degree of polynomial, type of basis function $\boldsymbol{\phi}$
- Don't select, just integrate

$$p(t|\boldsymbol{x}, \mathcal{D}) = \sum_{i=1}^{L} p(t|\boldsymbol{x}, \mathcal{M}_i, \mathcal{D}) p(\mathcal{M}_i|\mathcal{D})$$

- Average together the results of all models
- Could choose most likely model a posteriori $p(\mathcal{M}_i|\mathcal{D})$
  - More efficient, approximation

Bayesian Model Selection

- How do we compute the posterior over models?

$$p(\mathcal{M}_i|\mathcal{D}) \propto p(\mathcal{D}|\mathcal{M}_i)p(\mathcal{M}_i)$$

- Another likelihood + prior combination
- Likelihood:

$$p(\mathcal{D}|\mathcal{M}_i) = \int p(\mathcal{D}|\mathbf{w}, \mathcal{M}_i)p(\mathbf{w}|\mathcal{M}_i)d\mathbf{w}$$

Conclusion

- Readings: Ch. 3.1, 3.1.1-3.1.4, 3.3.1-3.3.2, 3.4
- Linear Models for Regression
  - Linear combination of (non-linear) basis functions
- Fitting parameters of regression model
  - Least squares
  - Maximum likelihood (can be = least squares)
- Controlling over-fitting
  - Regularization
  - Bayesian, use prior (can be = regularization)
- Model selection
  - Cross-validation (use held-out data)
  - Bayesian (use model evidence, likelihood)