Regularization and overfitting

Sriram Sankararaman

The instructor gratefully acknowledges Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Linear regression (Ordinary Least Squares or OLS)

Setup

- Input: $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$ (covariates, predictors, features, etc)
- Output: $y \in \mathbb{R}$ (responses, targets, outcomes, outputs, etc)
- Hypotheses/Model: $h_{\boldsymbol{w},b} : \boldsymbol{x} \rightarrow y$, with
  $h_{\boldsymbol{w},b}(\boldsymbol{x}) = b + \sum_d w_d x_d = b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$

  $\boldsymbol{w} = [w_1 \; w_2 \; \cdots \; w_{\mathsf{D}}]^{\mathrm{T}}$: *weights*,

  b is called *bias*.

  $\boldsymbol{\theta} = [b \; w_1 \; w_2 \; \cdots \; w_{\mathsf{D}}]^{\mathrm{T}}$ parameters
- Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, \mathsf{N}\}$

Find parameters $\boldsymbol{\theta}$ that minimizes the cost function

- Find $\boldsymbol{\theta}$ that minimizes $J(\boldsymbol{\theta})$
- $J(\boldsymbol{\theta}) = ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}||_2^2 = \sum_n \left(y_n - \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n\right)^2$
- J is the RSS (residual sum of squares) on training data.
- Probabilistic interpretation
  - In this probabilistic interpretation, finding $\boldsymbol{\theta}$ that minimizes J is equivalent to find the maximum likelihood estimates of the parameters of the model.

Solution to linear regression

Compute the gradient of J and find the value of $\boldsymbol{\theta}$ at which the gradient vanishes.

$$\nabla J(\boldsymbol{\theta}) = \mathbf{0}$$

This leads us to solve the normal equations

$$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\theta} = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

to obtain

$$\widehat{\boldsymbol{\theta}} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

Outline

What if $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ is not invertible

Answer 1: N $<$ D. Intuitively, not enough data to estimate all the parameters.

Answer 2: $\boldsymbol{X}$ columns are not linearly independent. Intuitively, there are two features that are perfectly correlated. In this case, solution is not unique.

l_2 regularized linear regression (ridge regression)

Solution

$$\widehat{\boldsymbol{\theta}} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

l_2 regularized linear regression (ridge regression)

Solution

$$\widehat{\boldsymbol{\theta}} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

This is equivalent to adding an extra term to $J(\boldsymbol{\theta})$

$$\overbrace{||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}||_2^2}^{J(\boldsymbol{\theta})} + \underbrace{\lambda\sum_{d=1}^{D}{\theta_d}^2}_{\text{regularization}}$$

- $\sum_{d=1}^{D}{\theta_d}^2 = \sum_{d=1}^{D} w_d^2 = \|\boldsymbol{w}\|_2^2$
- l_2-regularization uses the squared 2-norm

l_2 regularized linear regression (ridge regression)

Solution

$$\widehat{\boldsymbol{\theta}} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

This is equivalent to adding an extra term to $J(\boldsymbol{\theta})$

$$\overbrace{||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}||_2^2}^{J(\boldsymbol{\theta})} + \underbrace{\lambda\sum_{d=1}^{D}{\theta_d}^2}_{\text{regularization}}$$

Trade-off two quantities: minimize the error while keeping the weights small.

- $\sum_{d=1}^{D}{\theta_d}^2 = \sum_{d=1}^{D} w_d^2 = \|\boldsymbol{w}\|_2^2$
- l_2-regularization uses the squared 2-norm

l_2 regularized linear regression (ridge regression)

Solution

$$\widehat{\boldsymbol{\theta}} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

This is equivalent to adding an extra term to $J(\boldsymbol{\theta})$

$$\overbrace{\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|_2^2}^{J(\boldsymbol{\theta})} + \underbrace{\lambda\sum_{d=1}^{D}{\theta_d}^2}_{\text{regularization}}$$

- $\sum_{d=1}^{D}{\theta_d}^2 = \sum_{d=1}^{D} w_d^2 = \|\boldsymbol{w}\|_2^2$
- l_2-regularization uses the squared 2-norm

Benefits

- Numerically more stable, invertible matrix
- Prevent overfitting — more on this later

How to choose λ?

λ is referred as *hyperparameter*

- In contrast $\boldsymbol{\theta}$ is the parameter vector
- Use validation or cross-validation to find good choice of λ

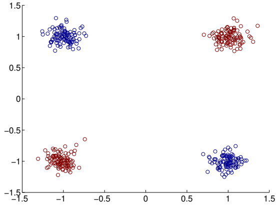
Mini-summary

- l_2 regularized linear regression (ridge regression) can be helpful to avoid numerical instability.
- Only a minor modification to the OLS estimate
- New hyperparameter λ needs to be chosen using cross-validation.

Outline

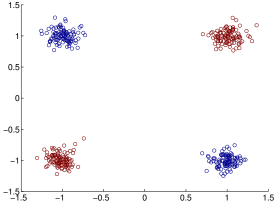
What if data is not linearly separable or fits to a line

Example of nonlinear classification

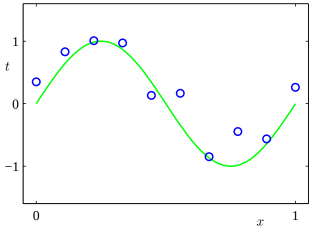


What if data is not linearly separable or fits to a line

Example of nonlinear classification



Example of nonlinear regression



Nonlinear basis for classification

Transform the input/feature

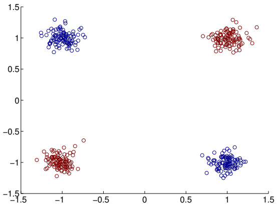
$$\phi(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^2 \rightarrow z = x_1 \cdot x_2$$

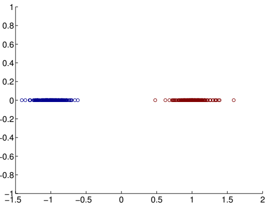
Nonlinear basis for classification

Transform the input/feature

$$\phi(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^2 \rightarrow z = x_1 \cdot x_2$$

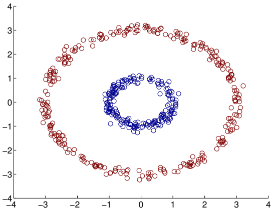
Transformed training data: linearly separable!



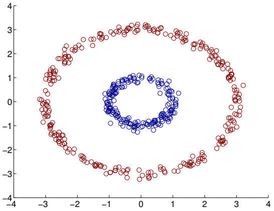


Another example

How to transform the input/feature?



Another example

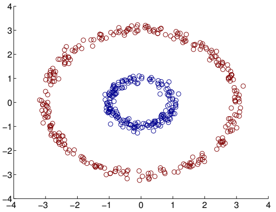


How to transform the input/feature?

$$\phi(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^2 \rightarrow \boldsymbol{z} = \left[\begin{array}{c} x_1^2 \\ x_1 \cdot x_2 \\ x_2^2 \end{array} \right] \in \mathbb{R}^3$$

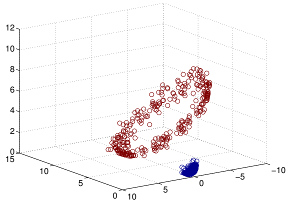
Another example

How to transform the input/feature?



$$\phi(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^2 \rightarrow \boldsymbol{z} = \begin{bmatrix} x_1^2 \\ x_1 \cdot x_2 \\ x_2^2 \end{bmatrix} \in \mathbb{R}^3$$

Transformed training data: linearly separable



General nonlinear basis functions

We can use a nonlinear mapping

$$\boldsymbol{\phi}(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^D \rightarrow \boldsymbol{z} \in \mathbb{R}^M$$

where M is the dimensionality of the new feature/input $\boldsymbol{z}$ (or $\boldsymbol{\phi}(\boldsymbol{x})$). Note that M could be either greater than D or less than or the same.

With the new features, we can apply our learning techniques to minimize our errors on the transformed training data

- linear methods: prediction is based on $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$
- other methods: nearest neighbors, decision trees, etc

Regression with nonlinear basis

Residual sum squares

$$\sum_n [\boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) - y_n]^2$$

where $\boldsymbol{\theta} \in \mathbb{R}^M$, the same dimensionality as the transformed features $\boldsymbol{\phi}(\boldsymbol{x})$.

The linear regression solution can be formulated with the new design matrix

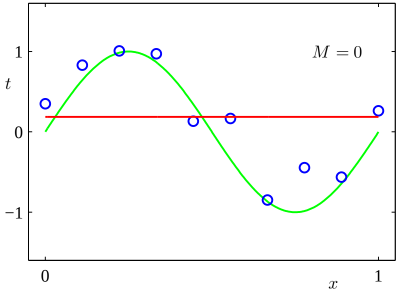
$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1)^{\mathrm{T}} \\ \boldsymbol{\phi}(\boldsymbol{x}_2)^{\mathrm{T}} \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_N)^{\mathrm{T}} \end{pmatrix} \in \mathbb{R}^{N \times M}, \quad \widehat{\boldsymbol{\theta}} = \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{y}$$

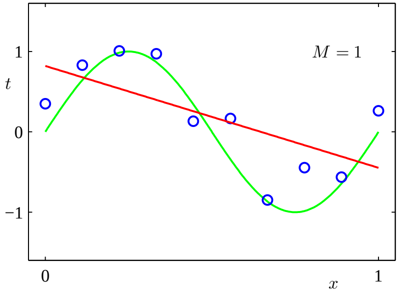
Example with regression

Polynomial basis functions

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix} \Rightarrow f(x) = \theta_0 + \sum_{m=1}^{M} \theta_m x^m$$

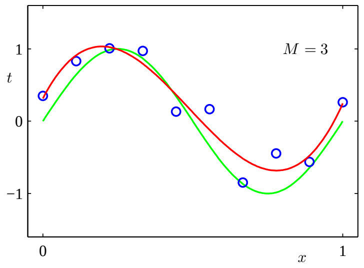
Fitting samples from a sine function: *underrfitting* as $f(x)$ is too simple



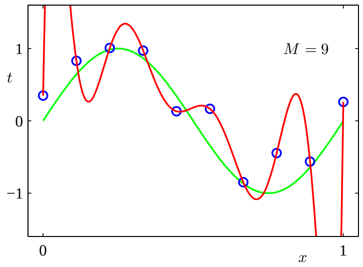


Adding high-order terms

M=3



M=9: *overfitting*



More complex features lead to better results on the training data, but potentially worse results on new data, e.g., test data!

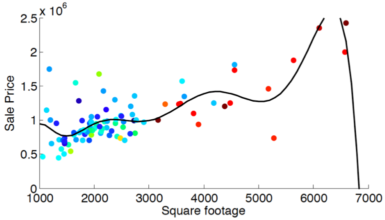
Overfiting

Parameters for higher-order polynomials are very large

	$M=0$	$M=1$	$M=3$	$M=9$
θ_0	0.19	0.82	0.31	0.35
θ_1		-1.27	7.99	232.37
θ_2			-25.43	-5321.83
θ_3			17.37	48568.31
θ_4				-231639.30
θ_5				640042.26
θ_6				-1061800.52
θ_7				1042400.18
θ_8				-557682.99
θ_9				125201.43

Overfitting can be quite disastrous

Fitting the housing price data with $M = 3$

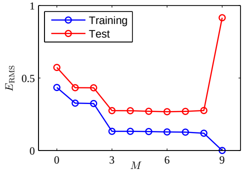


Note that the price would goes to zero (or negative) if you buy bigger ones!
This is called poor generalization/overfitting.

Detecting overfitting

Plot model complexity versus objective function

As a model increases in complexity, performance on training data keeps improving while performance on test data may first improve but eventually deteriorate.

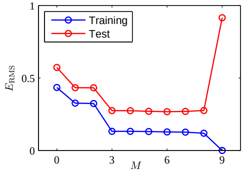


- Horizontal axis: *measure of model complexity*; in this example complexity defined by order of the polynomial basis functions.

Detecting overfitting

Plot model complexity versus objective function

As a model increases in complexity, performance on training data keeps improving while performance on test data may first improve but eventually deteriorate.



- Horizontal axis: *measure of model complexity*; in this example complexity defined by order of the polynomial basis functions.
- Vertical axis:
  1. For regression, residual sum of squares or residual mean squared (squared root of RSS)
  2. For classification, classification error rate.

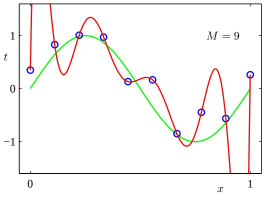
Mini-summary

- Started with linear models/hypotheses (for classification or regression) and showed how we can learn these models.
- But we can compute non-linear functions of the input features and use these as input to a linear model – removing the restriction of linearity.
- Does not change the learning algorithms!
- Very flexible
- Danger: of overfitting as features increase.

Outline

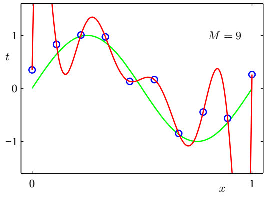
Use more training data to prevent over fitting

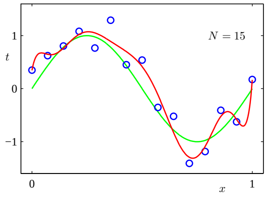
The more, the merrier

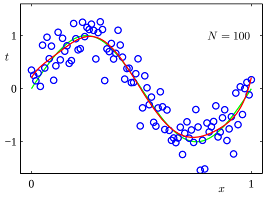


Use more training data to prevent over fitting

The more, the merrier

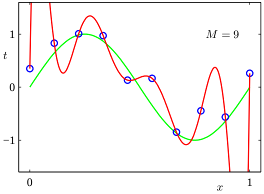


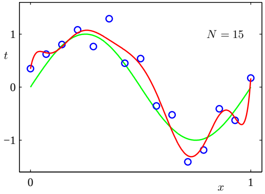


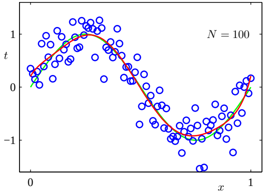


Use more training data to prevent over fitting

The more, the merrier







What if we do not have a lot of data?

Regularization methods

Intuition: For a linear model for regression

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$$

we can try to identify 'simpler' models. But what does it mean for a model to *be simple*?

Regularization methods

Intuition: For a linear model for regression

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$$

we can try to identify 'simpler' models. But what does it mean for a model to *be simple*?

Assumption (inductive bias)
A simpler model is one where most of the weights are zero.

A simpler model is one with smaller weights.

Why are smaller weights associated with simpler models?

Simpler functions are smoother, *i.e.*, nearby values of $\boldsymbol{x}$ have similar outputs $\hat{y}$.

Two values $\boldsymbol{x}$ and $\boldsymbol{x}'$ that differ in the first component by a small value ϵ.

Their predictions $\hat{y}$ and $\hat{y'}$ differ by ϵw_1.

Smaller w_1 (closer to zero), more similar are the predictions.

Regularized linear regression

A new cost function or error function to minimize

$$J(\boldsymbol{w}, b) = \sum_n (y_n - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n - b)^2 + \lambda\|\boldsymbol{w}\|_2^2$$

where $\lambda > 0$. This extra term $\|\boldsymbol{w}\|_2^2$ is called regularization/regularizer and controls the model complexity.

Regularized linear regression

A new cost function or error function to minimize

$$J(\boldsymbol{w}, b) = \sum_n (y_n - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - b)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

where $\lambda > 0$. This extra term $\|\boldsymbol{w}\|_2^2$ is called regularization/regularizer and controls the model complexity.

Intuitions

Regularized linear regression

A new cost function or error function to minimize

$$J(\boldsymbol{w}, b) = \sum_n (y_n - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n - b)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

where $\lambda > 0$. This extra term $\|\boldsymbol{w}\|_2^2$ is called regularization/regularizer and controls the model complexity.

Intuitions

- If $\lambda \to +\infty$, then

$$\widehat{\boldsymbol{w}} \to \mathbf{0}$$

Regularized linear regression

A new cost function or error function to minimize

$$J(\boldsymbol{w}, b) = \sum_n (y_n - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - b)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

where $\lambda > 0$. This extra term $\|\boldsymbol{w}\|_2^2$ is called regularization/regularizer and controls the model complexity.

Intuitions

- If $\lambda \to +\infty$, then

$$\widehat{\boldsymbol{w}} \to \mathbf{0}$$

- If $\lambda \to 0$, then we trust our data more. Numerically,

$$\widehat{\boldsymbol{w}} \to \arg\min \sum_n (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n + b - y_n)^2$$

Closed-form solution

For regularized linear regression: the solution changes very little (in form) from the OLS (Ordinary Least Squares) solution

$$\arg\min \sum_n (y_n - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n - b)^2 + \lambda\|\boldsymbol{w}\|_2^2 \Rightarrow \widehat{\boldsymbol{w}} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

and reduces to the OLS solution when $\lambda = 0$, as expected.

Closed-form solution

For regularized linear regression: the solution changes very little (in form) from the OLS (Ordinary Least Squares) solution

$$\arg\min \sum_n (y_n - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n - b)^2 + \lambda\|\boldsymbol{w}\|_2^2 \Rightarrow \widehat{\boldsymbol{w}} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

and reduces to the OLS solution when $\lambda = 0$, as expected.

If we have to use numerical procedure, the gradient would change nominally too,

$$\nabla J(\boldsymbol{w}) = 2(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} + \lambda\boldsymbol{w})$$

As long as $\lambda \geq 0$, the optimization is convex.

Example: fitting data with polynomials

Our regression model

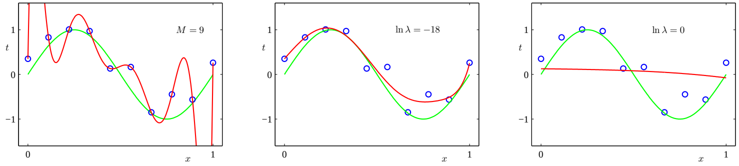
$$y = \sum_{m=1}^{M} w_m x^m$$

Regularization would discourage large parameter values as we saw with the OLS solution, thus potentially preventing overfitting.

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
w_0	0.19	0.82	0.31	0.35
w_1		-1.27	7.99	232.37
w_2			-25.43	-5321.83
w_3			17.37	48568.31
w_4				-231639.30
w_5				640042.26
w_6				-1061800.52
w_7				1042400.18
w_8				-557682.99
w_9				125201.43

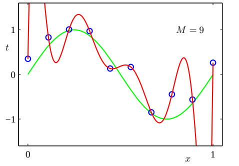
Overfitting in terms of λ

Overfitting is reduced from complex model to simpler one with the help of increasing regularizer

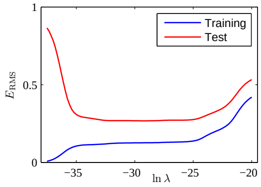


Overfitting in terms of λ

Overfitting is reduced from complex model to simpler one with the help of increasing regularizer



λ vs. residual error shows the difference of the model performance on training and testing dataset



The effect of λ

Large λ attenuates parameters towards 0

	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
w_0	0.35	0.35	0.13
w_1	232.37	4.74	-0.05
w_2	-5321.83	-0.77	-0.06
w_3	48568.31	-31.97	-0.06
w_4	-231639.30	-3.89	-0.03
w_5	640042.26	55.28	-0.02
w_6	-1061800.52	41.32	-0.01
w_7	1042400.18	-45.95	-0.00
w_8	-557682.99	-91.53	0.00
w_9	125201.43	72.68	0.01

l_2 regularized logistic regression

Adding regularizer to the cost function for logistic regression

$$J(\boldsymbol{w}, b) = -\sum_n \{y_n \log h_{\boldsymbol{w},b}(\boldsymbol{x}_n) + (1-y_n)\log[1-h_{\boldsymbol{w},b}(\boldsymbol{x}_n)]\} + \underbrace{\lambda\|\boldsymbol{w}\|_2^2}_{\text{regularization}}$$

$$h_{\boldsymbol{w},b}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b)$$

l_2 regularized logistic regression

Adding regularizer to the cost function for logistic regression

$$J(\boldsymbol{w}, b) = -\sum_n \{y_n \log h_{\boldsymbol{w},b}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n)]\} + \underbrace{\lambda \|\boldsymbol{w}\|_2^2}_{\text{regularization}}$$

$$h_{\boldsymbol{w},b}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b)$$

Numerical optimization

- Objective functions remains convex as long as $\lambda \geq 0$.
- Gradients and Hessians are changed marginally and can be easily derived.

How to choose the right amount of regularization?

Can we tune λ on the training dataset?

How to choose the right amount of regularization?

Can we tune λ on the training dataset?
No: as this will set λ to zero, i.e., without regularization, defeating our intention to use it to control model complexity and to gain better generalization.

λ **is thus a hyperparmeter**. To tune it,

- We can use a development/validation dataset independent of training and testing dataset.

The procedure is similar to choosing K in the nearest neighbor classifiers.

How to choose the right amount of regularization?

Can we tune λ on the training dataset?
No: as this will set λ to zero, i.e., without regularization, defeating our intention to use it to control model complexity and to gain better generalization.

λ is thus a hyperparmeter. To tune it,

- We can use a development/validation dataset independent of training and testing dataset.

The procedure is similar to choosing K in the nearest neighbor classifiers.

For different λ, we get $\widehat{\boldsymbol{w}}$ and evaluate the model on the development/validation dataset.

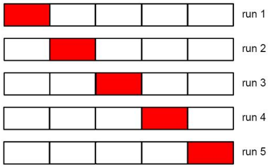
We then plot the curve λ versus prediction error (accuracy, classification error) and find the place that the performance on the validation is the best.

Use cross-validation to choose λ

Procedure

- Randomly partition training data into K *disjoint* parts
  Normally, K is chosen to be 10, 5, etc.
- For each possible value of λ
  1. Use one part as validation; use other $(K-1)$ parts as training
  2. Evaluate the model on the validation
  3. Do this K times, and average the performance on the validations
- Choose the λ with the best performance

When $K = N$ (the number of training examples), this becomes LOO.



Outline

Review: Probabilistic interpretation for OLS (ordinary least squares)

Review: Probabilistic interpretation for OLS (ordinary least squares)

- OLS (a.k.a linear regression): $Y = \boldsymbol{w}^\top \boldsymbol{x} + \eta$
  - Assume bias $b = 0$
  - $\eta \sim N(0, \sigma^2)$ is a Gaussian random variable
  - Thus, $Y \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}, \sigma^2)$
- We assume that $\boldsymbol{w}$ is fixed.
- We define $p(y|\boldsymbol{x}, \boldsymbol{w}, \sigma^2)$ as the distribution of y given fixed values for the parameters $\boldsymbol{w}, \sigma^2$
- The likelihood function maps parameters to probabilities

$$L(\boldsymbol{w}, \sigma^2) = \prod_n p(y_n|\boldsymbol{x}_n, \boldsymbol{w}, \sigma^2)$$

- Maximizing likelihood with respect to $(\boldsymbol{w}, \sigma^2)$ minimizes RSS and yields the OLS solution:

$$\boldsymbol{w}^{\text{OLS}} = \arg\max_{\boldsymbol{w}, \sigma^2} L(\boldsymbol{w}, \sigma^2)$$

Probabilistic interpretation of Ridge Regression

- Ridge Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{x} + \eta$
  - Bias $b = 0$ (as before)
  - $Y \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{X}, \sigma^2)$ is a Gaussian random variable (as before)
  - $w_d \sim \mathcal{N}(0, \sigma_0^2)$ are i.i.d. Gaussian random variables (unlike before)

Probabilistic interpretation of Ridge Regression

- Ridge Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{x} + \eta$
  - Bias $b = 0$ (as before)
  - $Y \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{X}, \sigma^2)$ is a Gaussian random variable (as before)
  - $w_d \sim \mathcal{N}(0, \sigma_0^2)$ are i.i.d. Gaussian random variables (unlike before)
- $\boldsymbol{w}$ is a random variable with a prior distribution (*Bayesian* interpretation)

Probabilistic interpretation of Ridge Regression

- Ridge Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{x} + \eta$
  - Bias $b = 0$ (as before)
  - $Y \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{X}, \sigma^2)$ is a Gaussian random variable (as before)
  - $w_d \sim \mathcal{N}(0, \sigma_0^2)$ are i.i.d. Gaussian random variables (unlike before)
- $\boldsymbol{w}$ is a random variable with a prior distribution (*Bayesian* interpretation)
- To find $\boldsymbol{w}$ given data $\mathcal{D}$, we can compute posterior distribution of $\boldsymbol{w}$ using Bayes' theorem:

$$p(\boldsymbol{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w})}{p(\mathcal{D})} = \frac{p(\mathcal{D}, \boldsymbol{w})}{p(\mathcal{D})}$$

Probabilistic interpretation of Ridge Regression

- Ridge Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{x} + \eta$
  - Bias $b = 0$ (as before)
  - $Y \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{X}, \sigma^2)$ is a Gaussian random variable (as before)
  - $w_d \sim \mathcal{N}(0, \sigma_0^2)$ are i.i.d. Gaussian random variables (unlike before)
- $\boldsymbol{w}$ is a random variable with a prior distribution (*Bayesian* interpretation)
- To find $\boldsymbol{w}$ given data $\mathcal{D}$, we can compute posterior distribution of $\boldsymbol{w}$ using Bayes' theorem:

$$p(\boldsymbol{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w})}{p(\mathcal{D})} = \frac{p(\mathcal{D}, \boldsymbol{w})}{p(\mathcal{D})}$$

- Maximum a posterior (MAP) estimate:

$$\boldsymbol{w}^{\text{MAP}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\mathcal{D}) = \arg\max_{\boldsymbol{w}} p(\mathcal{D}, \boldsymbol{w})$$

Probabilistic interpretation of Ridge Regression

- Ridge Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{x} + \eta$
  - Bias $b = 0$ (as before)
  - $Y \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{X}, \sigma^2)$ is a Gaussian random variable (as before)
  - $w_d \sim \mathcal{N}(0, \sigma_0^2)$ are i.i.d. Gaussian random variables (unlike before)
- $\boldsymbol{w}$ is a random variable with a prior distribution (*Bayesian* interpretation)
- To find $\boldsymbol{w}$ given data $\mathcal{D}$, we can compute posterior distribution of $\boldsymbol{w}$ using Bayes' theorem:

$$p(\boldsymbol{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w})}{p(\mathcal{D})} = \frac{p(\mathcal{D}, \boldsymbol{w})}{p(\mathcal{D})}$$

- Maximum a posterior (MAP) estimate:

$$\boldsymbol{w}^{\text{MAP}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\mathcal{D}) = \arg\max_{\boldsymbol{w}} p(\mathcal{D}, \boldsymbol{w})$$

- What's the relationship between MAP and MLE?

Probabilistic interpretation of Ridge Regression

- Ridge Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{x} + \eta$
  - Bias $b = 0$ (as before)
  - $Y \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{X}, \sigma^2)$ is a Gaussian random variable (as before)
  - $w_d \sim \mathcal{N}(0, \sigma_0^2)$ are i.i.d. Gaussian random variables (unlike before)
- $\boldsymbol{w}$ is a random variable with a prior distribution (*Bayesian* interpretation)
- To find $\boldsymbol{w}$ given data $\mathcal{D}$, we can compute posterior distribution of $\boldsymbol{w}$ using Bayes' theorem:

$$p(\boldsymbol{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w})}{p(\mathcal{D})} = \frac{p(\mathcal{D}, \boldsymbol{w})}{p(\mathcal{D})}$$

- Maximum a posterior (MAP) estimate:

$$\boldsymbol{w}^{\text{MAP}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\mathcal{D}) = \arg\max_{\boldsymbol{w}} p(\mathcal{D}, \boldsymbol{w})$$

- What's the relationship between MAP and MLE?
  - MAP reduces to MLE if we assume uniform prior for $p(\boldsymbol{w})$

Estimating $\boldsymbol{w}$

- Let $Y_1, \ldots, Y_N$ be independent with $y_n|\boldsymbol{w}, \boldsymbol{x}_n \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \sigma^2)$
- Let w_d be IID with $w_d \sim \mathcal{N}(0, \sigma_0^2)$

Joint likelihood of data and parameters (given σ_0, σ)

$$p(\mathcal{D}, \boldsymbol{w}) = p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w}) \quad = \prod_n p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) \prod_d p(w_d)$$

Joint log likelihood Plugging in Gaussian PDF, we get:

$$\begin{aligned}\log p(\mathcal{D}, \boldsymbol{w}) &= \sum_n \log p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) + \sum_d \log p(w_d) \\ &= -\frac{\sum_n (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - y_n)^2}{2\sigma^2} - \sum_d \frac{1}{2\sigma_0^2} w_d^2 + \text{const}\end{aligned}$$

Estimating $\boldsymbol{w}$

- Let $Y_1, \ldots, Y_N$ be independent with $y_n|\boldsymbol{w}, \boldsymbol{x}_n \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \sigma^2)$
- Let w_d be IID with $w_d \sim \mathcal{N}(0, \sigma_0^2)$

Joint likelihood of data and parameters (given σ_0, σ)

$$p(\mathcal{D}, \boldsymbol{w}) = p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w}) \quad = \prod_n p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) \prod_d p(w_d)$$

Joint log likelihood Plugging in Gaussian PDF, we get:

$$\begin{aligned}\log p(\mathcal{D}, \boldsymbol{w}) &= \sum_n \log p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) + \sum_d \log p(w_d) \\ &= -\frac{\sum_n (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - y_n)^2}{2\sigma^2} - \sum_d \frac{1}{2\sigma_0^2} w_d^2 + \text{const}\end{aligned}$$

MAP estimate: $\boldsymbol{w}^{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} \log p(\mathcal{D}, \boldsymbol{w})$

- As with OLS, set gradient equal to zero and solve (for $\boldsymbol{w}$)

What is the relationship between minimizing J in ridge regression and maximizing the posterior?

Ridge regression:

$$\begin{aligned}\boldsymbol{w} &= \arg\min_{\boldsymbol{w}} J(\boldsymbol{w}) \\ &= \arg\min_{\boldsymbol{w}} \sum_n (\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n - y_n)^2 + \lambda\|\boldsymbol{w}\|_2^2\end{aligned}$$

MAP estimate:

$$\begin{aligned}\boldsymbol{w}^{\mathrm{MAP}} &= \arg\max_{\boldsymbol{w}} \log p(\mathcal{D}, \boldsymbol{w}) \\ &= \arg\max_{\boldsymbol{w}} \left[-\frac{\sum_n (\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n - y_n)^2}{2\sigma^2} - \sum_d \frac{1}{2\sigma_0^2} w_d^2 + \text{const} \right]\end{aligned}$$

Mini-summary

- Regularization: prefering a simpler model by penalizing large weights or many non-zero weights.
- Helpful to reduce overfitting and to prevent numerical instability.
- Can be added to any linear model.
- Only a minor modification to the unregularized algorithms.
- New hyperparameter λ needs to be chosen using cross-validation.
- Has a probabilistic interpretation.

Outline

Supervised learning

We aim to build a function $h(\boldsymbol{x})$ to predict the true value y associated with $\boldsymbol{x}$. If we make a mistake, we incur a *loss*

$$\ell(y, h(\boldsymbol{x}))$$

Supervised learning

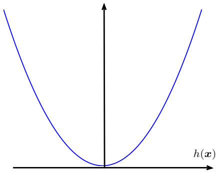
We aim to build a function $h(\boldsymbol{x})$ to predict the true value y associated with $\boldsymbol{x}$. If we make a mistake, we incur a *loss*

$$\ell(y, h(\boldsymbol{x}))$$

Example:squared loss function for regression when y is continuous

$$\ell(y, h(\boldsymbol{x})) = [h(\boldsymbol{x}) - y]^2$$

Ex: when $y = 0$



Other types of loss functions

For classification: $0/1$ loss

$$\ell(y, h(\boldsymbol{x})) = \mathbf{1}\{y \neq h(\boldsymbol{x})\}$$

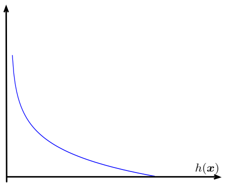
Assumes h outputs values in $\{-1, +1\}$.

Other types of loss functions

For classification: $logistic$ loss

$\ell(y, h(\boldsymbol{x})) = -y \log h(\boldsymbol{x}) - (1-y) \log[1-h(\boldsymbol{x})]$

Assumes h outputs values in $[0, 1]$.
Ex: when $y = 1$



Measure how good our hypothesis h is

Risk/ expected test loss: assume we know the true distribution of data $p(\boldsymbol{x}, y)$, the *risk* is

$$\mathcal{R}[h(\boldsymbol{x})] = \sum_{\boldsymbol{x},y} \ell(h(\boldsymbol{x}), y) p(\boldsymbol{x}, y)$$

Measure how good our hypothesis h is

Risk/ expected test loss: assume we know the true distribution of data $p(\boldsymbol{x}, y)$, the *risk* is

$$\mathcal{R}[h(\boldsymbol{x})] = \sum_{\boldsymbol{x},y} \ell(h(\boldsymbol{x}), y) p(\boldsymbol{x}, y)$$

However, we cannot compute $R[h(\boldsymbol{x})]$, so we use *empirical risk/training error*, given a training dataset $\mathcal{D}$

$$\mathcal{R}^{\text{EMP}}[h(\boldsymbol{x})] = \frac{1}{N} \sum_{n} \ell(h(\boldsymbol{x}_n), y_n)$$

Measure how good our hypothesis h is

Risk/ expected test loss: assume we know the true distribution of data $p(\boldsymbol{x}, y)$, the *risk* is

$$\mathcal{R}[h(\boldsymbol{x})] = \sum_{\boldsymbol{x},y} \ell(h(\boldsymbol{x}), y) p(\boldsymbol{x}, y)$$

However, we cannot compute $R[h(\boldsymbol{x})]$, so we use *empirical risk/training error*, given a training dataset $\mathcal{D}$

$$\mathcal{R}^{\text{EMP}}[h(\boldsymbol{x})] = \frac{1}{N} \sum_{n} \ell(h(\boldsymbol{x}_n), y_n)$$

Intuitively, as $N \rightarrow +\infty$,

$$\mathcal{R}^{\text{EMP}}[h(\boldsymbol{x})] \rightarrow \mathcal{R}[h(\boldsymbol{x})]$$

Pick a good hypothesis h

A good hypothesis h is one that minizes risk/expected test loss $\mathcal{R}[h(\boldsymbol{x})]$ but we cannot even compute it!

Instead pick h that minimizes empirical risk/training error $\mathcal{R}^{\text{EMP}}[h(\boldsymbol{x})]$

This strategy is known as empirical risk minimization.

How this relates to what we have learned?

So far, we have been doing empirical risk minimization (ERM)

- For linear regression, $h_{\boldsymbol{w},b}(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$, and we use squared loss
- For logistic regression, $h_{\boldsymbol{w},b}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b)$, and we use logistic loss
- Finding the best h is achieved by searching for $(\boldsymbol{w}, b)$ that minimizes the trainiing error/empirical risk.

How this relates to what we have learned?

So far, we have been doing empirical risk minimization (ERM)

- For linear regression, $h_{\boldsymbol{w},b}(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$, and we use squared loss
- For logistic regression, $h_{\boldsymbol{w},b}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b)$, and we use logistic loss
- Finding the best h is achieved by searching for $(\boldsymbol{w}, b)$ that minimizes the trainiing error/empirical risk.

ERM might be problematic

- If $h(\boldsymbol{x})$ is complicated enough,

$$\mathcal{R}^{\mathrm{EMP}}[h(\boldsymbol{x})] \rightarrow 0$$

- But then $h(\boldsymbol{x})$ is unlikely to do well in predicting things out of the training dataset $\mathcal{D}$
- This is called *poor generalization* or *overfitting*. We have just discussed approaches to address this issue.

Regularizer

Instead of $\mathcal{R}^{\text{emp}}$, use

$$\begin{aligned}&\arg\min_{\boldsymbol{w},b} \mathcal{R}^{\text{EMP}}[h_{\boldsymbol{w},b}(x)] + \lambda R(\boldsymbol{w}, b)\\ &= \arg\min_{\boldsymbol{w},b} \frac{1}{N}\sum_{n} \ell(y_n, h_{\boldsymbol{w},b}(\boldsymbol{x}_n)) + \lambda R(\boldsymbol{w}, b)\end{aligned} \tag{1}$$

Loss functions ℓ, $\hat{y} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$

- Zero/One: $\ell(y, h(x)) = \mathbf{1}\{y \neq h(x)\} = \mathbf{1}\{y\hat{y} \leq 0\}$

Regularizer

Instead of $\mathcal{R}^{\text{emp}}$, use

$$\begin{aligned} &\arg\min_{\boldsymbol{w},b} \mathcal{R}^{\text{EMP}}[h_{\boldsymbol{w},b}(x)] + \lambda R(\boldsymbol{w}, b) \\ &= \arg\min_{\boldsymbol{w},b} \frac{1}{N} \sum_n \ell(y_n, h_{\boldsymbol{w},b}(\boldsymbol{x}_n)) + \lambda R(\boldsymbol{w}, b) \end{aligned} \tag{1}$$

Regularizer

- Squared 2-norm: $R(\boldsymbol{w}, b) = \|\boldsymbol{w}\|_2^2 = \sum_{d=1}^{D} {w_d}^2$
- 1-norm: $R(\boldsymbol{w}, b) = \|\boldsymbol{w}\|_1 = \sum_{d=1}^{D} |w_d|$.
- 0-norm: $R(\boldsymbol{w}, b) = \sum_{d=1}^{D} \mathbf{1}\{w_d \neq 0\}$.
- p-norm: $R(\boldsymbol{w}, b) = \|\boldsymbol{w}\|_p = (\sum_{d=1}^{D} |w_d|^p)^{\frac{1}{p}}$

Framework for supervised learning

- Pick:
  - Model/hypotheses
  - Loss function
  - Regularizer
  - Algorithm to solve optimization problem

These choices lead to different learning algorithms

Framework for machine learning

- Application
  - Labeled vs unlabeled data
  - Labeled: supervised learning. Type of label: categorical (classification), quantitative (regression)
  - Unlabeled: unsupervised learning.
- Model/hypotheses
- Optimization problem
- Algorithm to solve optimization problem

- Good luck on the mid-term!
- Extra office hours on Friday.