Linear models for regression and classification

Devika Subramanian

Rice University

devika@rice.edu

August 15 and 16, 2019

Outline

Formulating linear regression

Given:

- training data $\mathcal{D} = \{(x^{(i)}, y^{(i)}) | 1 \leq i \leq m, x^{(i)} \in \Re^d, y^{(i)} \in \Re\}$ drawn i.i.d. from a fixed, but unknown distribution.
- parametric function class $h_\theta : \Re^{d+1} \rightarrow \Re$

$$h_\theta(x) = \theta^T [1; x] = \theta_0 + \theta_1 x_1 + \ldots + \theta_d x_d$$

  - model is linear in θ, $\theta \in \Re^{d+1}$
- Empirical loss $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right)^2$

Find: $\theta^* = argmin_\theta J(\theta)$ [minimize empirical loss]

What are the classes of functions?
What loss functions does your model support?

An example: predicting home prices

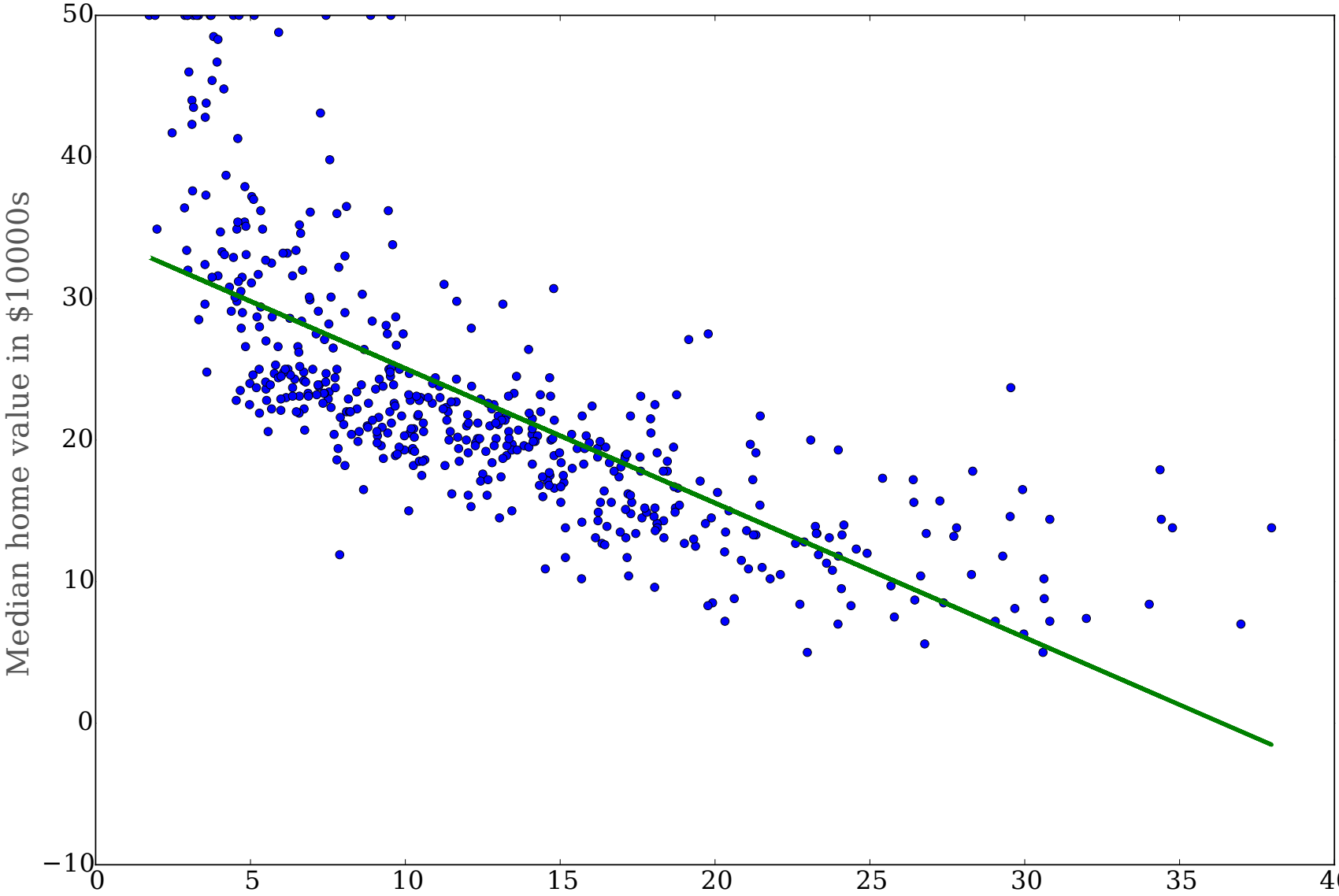
- The goal is to predict the median value of a home in the Boston suburbs based on thirteen characteristics of census tracts in the suburbs. There are 503 census tracts ($m = 503, d = 13$).

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	. . .	LSTAT
0.00632	18.00	2.31	0.00	0.538	6.575	65.20	4.09	. . .	4.98
0.02731	00.00	7.07	0.00	0.469	6.42	78.9	4.967	. . .	9.14
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

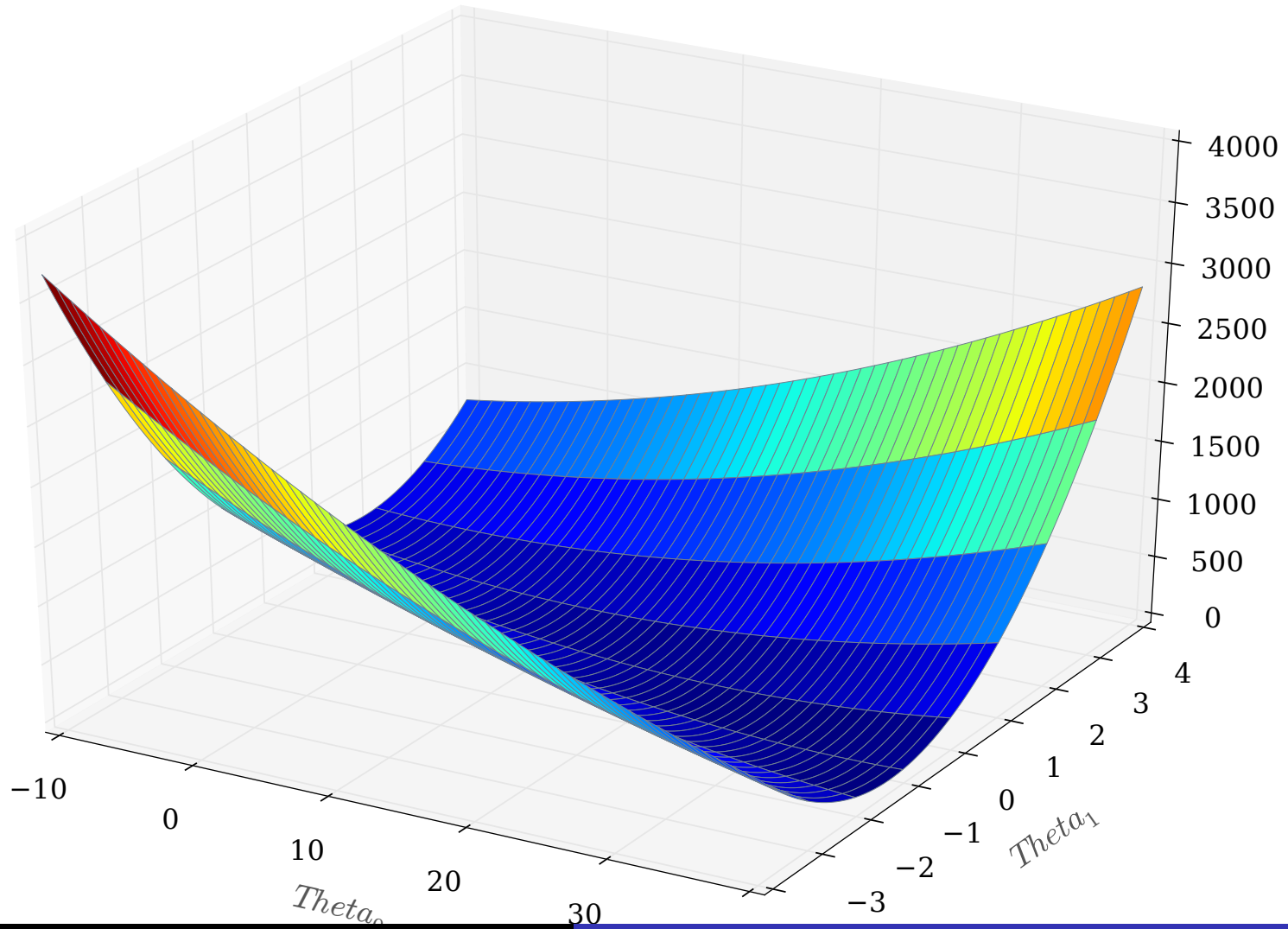
- We will first build a linear model that predicts median home value from the LSTAT (lower socioeconomic status percentage) feature alone.

$$h_\theta(LSTAT) = \theta_0 + \theta_1 LSTAT$$

Visualization of the regression problem



Visualization of the empirical loss $J(\theta)$



Gradient descent to estimate θ

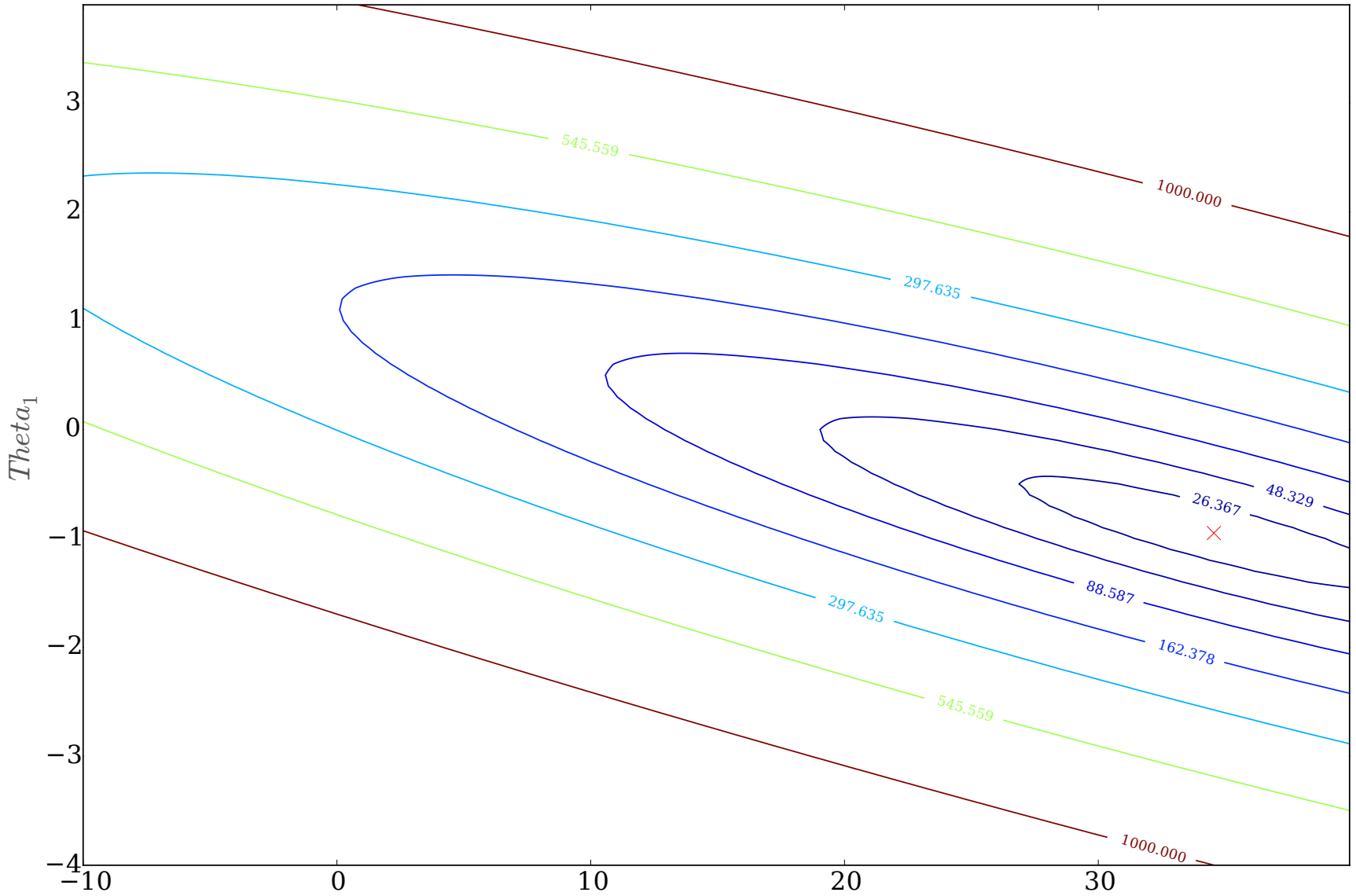
- Empirical loss $J(\theta)$ is convex and has a unique global minimum (quadratic in θ).
- A gradient descent algorithm can find θ^* which minimizes $J(\theta)$.
  - Start with a random guess for θ (e.g., the all zeros vector).
  - Repeatedly update θ by going in the direction of the steepest descent of $J(\theta)$.

Choice of Alpha
Small steps will avoid overshooting

$$\begin{aligned} \theta_j &\leftarrow \theta_j - \alpha \frac{\partial J}{\partial \theta_j}; \quad 0 \leq j \leq d \\ &\leftarrow \theta_j + \frac{\alpha}{m} \sum_{i=1}^{m} (y^{(i)} - \theta^T [1; x^{(i)}]) x_j^{(i)} \end{aligned}$$

until θ converges. $0 < \alpha \leq 1$ is the learning rate and is typically chosen to a small number, e.g., 0.05. α needs to be chosen carefully to ensure convergence.

Another visualization of the empirical loss $J(\theta)$



Stochastic gradient descent

Standard gradient descent computes the gradient of J with respect to all examples in $\mathcal{D}$. Stochastic gradient descent (SGD) updates θ on each example. SGD causes faster convergence, especially when m is large.

- Start with a random guess for θ (e.g., the all zeros vector).
- Repeatedly update θ by going in the direction of the steepest descent of $J(\theta)$.
  Loop
  for i $= 1$ to m do:

$$\theta_j \leftarrow \theta_j + \frac{\alpha}{m}(y^{(i)} - \theta^T[1; x^{(i)}])x_j^{(i)}$$

  until θ converges.

Most popular = batch / mini-batch to compute gradient

Making gradient descent work in practice

- Need to choose learning rate α very carefully.
- Need to scale the features in X so that coefficients θ_j are more or less of the same order. Example of scaling: zero mean- unit variance scaling of each column of X.

  Need to normalize / standardize values column by column; ensure gradient steps are even.

Closed-form solution for θ

Closed form not good for large data problems, hence use stochastic / batch

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \theta^T x^{(i)}\right)^2$$

$$= \frac{1}{2m}(X\theta - y)^T(X\theta - y)$$

where

$$X = \begin{bmatrix} 1 & \text{——} x^{(1)\,T} \text{——} \\ 1 & \text{——} x^{(2)\,T} \text{——} \\ \cdots & \\ 1 & \text{——} x^{(m)\,T} \text{——} \end{bmatrix}, y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \cdots \\ y^{(m)} \end{bmatrix} \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \cdots \\ \theta_d \end{bmatrix}$$

By solving for $\bigtriangledown_\theta J(\theta) = 0$, we get the normal equation

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

Vectorized form of gradient descent

- Start with a random guess for θ (e.g., the all zeros vector).
- Repeatedly update θ by going in the direction of the steepest descent of $J(\theta)$.

$$\begin{aligned} \theta &\leftarrow \theta - \alpha \bigtriangledown_\theta J(\theta) \\ &\leftarrow \theta - \frac{\alpha}{m} X^T (X\theta - y) \end{aligned}$$

until θ converges.

This algorithm works not just for $\theta \in \Re^2$, but for general $\theta \in \Re^{d+1}$.

Probabilistic interpretation of linear regression

We will assume a *generative model* for the data:

$$y^{(i)} = \theta^{(i)} x^{(i)} + \epsilon^{(i)}, \epsilon^{(i)} \sim N(0, \sigma^2)$$

For simplicity, assume $x^{(i)}, y^{(i)} \in \Re$. Then,

$$p(\epsilon^{(i)} \mid \theta, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{{\epsilon^{(i)}}^2}{2\sigma^2}\right)$$

$$p((x^{(i)}, y^{(i)}) \mid \theta, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

Maximum likelihood estimation of linear regression parameters

Given $\mathcal{D} = \{(x^{(i)}, y^{(i)}) \mid 1 \leq i \leq m, x^{(i)} \in \Re, y^{(i)} \in \Re\}$, and assuming that the data in $\mathcal{D}$ are drawn i.i.d.,

$$\begin{aligned} P(\mathcal{D} \mid \theta, \sigma^2) &= \prod_{i=1}^{m} p((x^{(i)}, y^{(i)}) \mid \theta, \sigma^2) \\ &= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \end{aligned}$$

Principle of maximum likelihood:

$$\theta^*_{MLE} = argmax_\theta P(\mathcal{D} \mid \theta)$$

Maximum likelihood estimation of linear regression parameters (contd.)

$$\begin{aligned} \theta^*_{MLE} &= argmax_\theta logP(\mathcal{D} \mid \theta) \\ &= argmin_\theta - logP(\mathcal{D} \mid \theta) \end{aligned}$$

The quantity $-logP(\mathcal{D} \mid \theta)$ is called the negative log likelihood function (NLL).

$$\begin{aligned} logP(\mathcal{D} \mid \theta) &= log\left(\prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)\right) \\ &= \sum_{i=1}^{m}\left(log\frac{1}{\sqrt{2\pi\sigma^2}} - \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \end{aligned}$$

Maximum likelihood estimation of linear regression parameters (contd.)

$$NLL(\theta) \quad = \quad -logP(\mathcal{D} \mid \theta) = \frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \theta^T x^{(i)})^2$$

Now we see that the squared error loss function arises out of maximum likelihood estimation of θ based on a linear generative model for the data.

Basis function expansion: improving expressiveness

- The goal is to predict the median value of a home in the Boston suburbs based on thirteen characteristics of census tracts in the suburbs.

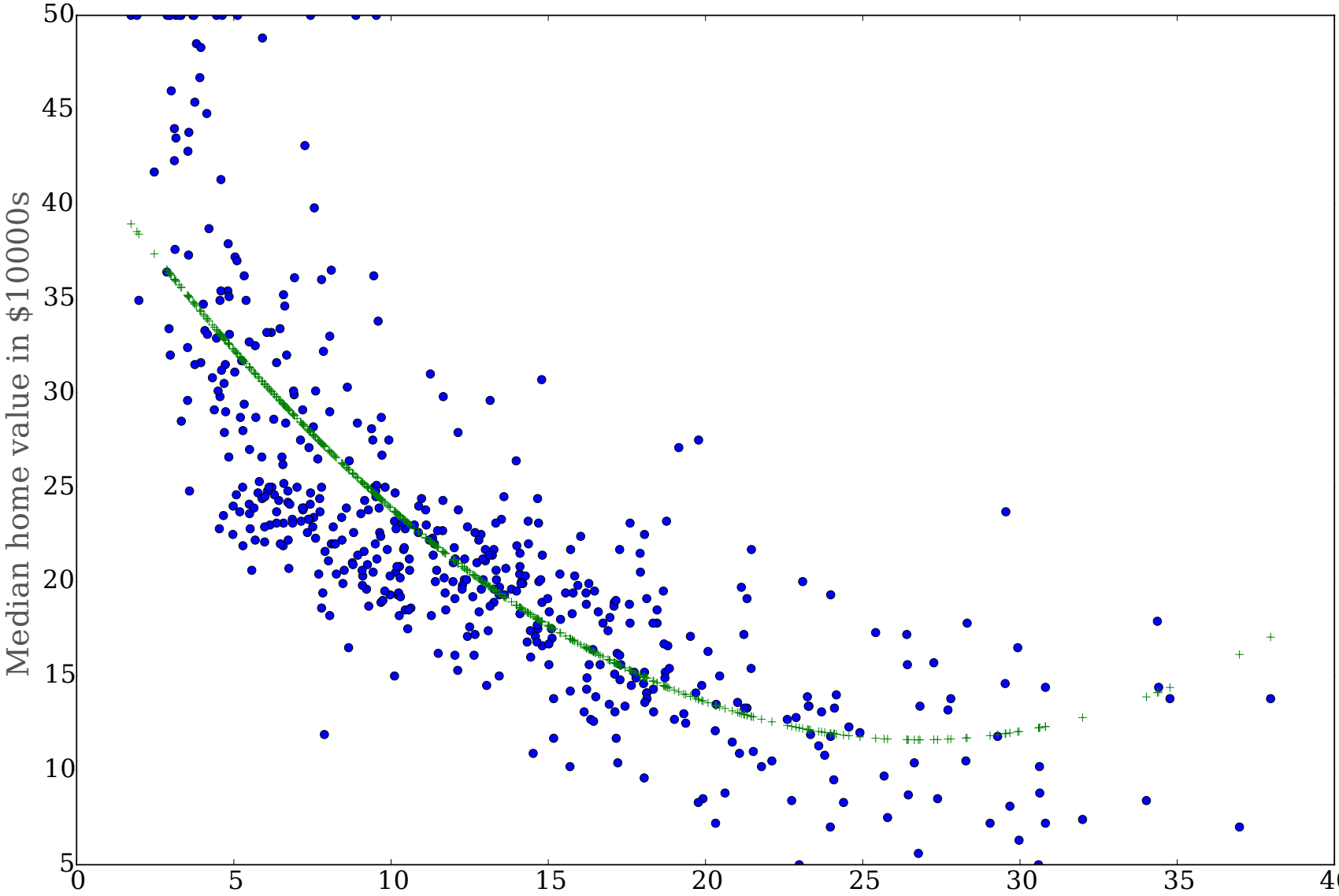
CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	. . .	LSTAT
0.00632	18.00	2.31	0.00	0.538	6.575	65.20	4.09	. . .	4.98
0.02731	00.00	7.07	0.00	0.469	6.42	78.9	4.967	. . .	9.14
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

- We will now build a model that predicts median home value from the LSTAT (lower socioeconomic status percentage) feature alone.

$$h_\theta(LSTAT) = \theta_0 + \theta_1 LSTAT + \theta_2 LSTAT^2$$

Note that while our model is non-linear in LSTAT, it is still linear in θ. Our algorithms for finding θ to minimize $J(\theta)$ can be used here! We need to scale the features $LSTAT$ and $LSTAT^2$ so that the θ components are more or less of the same order.

Quadratic basis function example



Basis function expansion

Suppose $x \in \Re$ and $y \in \Re$. We can build function classes h_θ that are all linear in θ.

$$\begin{aligned} h_\theta(x) &= \theta_0 + \theta_1 x \\ h_\theta(x) &= \theta_0 + \theta_1 x + \theta_p x^2 \\ &\dots \\ h_\theta(x) &= \theta_0 + \theta_1 x + \ldots + \theta_x x^p \end{aligned}$$

We can model basis function expansion with $\phi : \Re \rightarrow \Re^p$ which maps $x \in \Re \rightarrow \phi(x) = (x, x^2, \ldots, x^p) \in \Re^p$. In our algorithms, we replace X by $\phi(X)$.

$$X = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \dots & \dots \\ 1 & x^{(m)} \end{bmatrix}, \phi(X) = \begin{bmatrix} 1 & x^{(1)} & x^{(1)^2} & \dots & x^{(1)^p} \\ 1 & x^{(2)} & x^{(2)^2} & \dots & x^{(2)^p} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x^{(m)} & x^{(m)^2} & \dots & x^{(m)^p} \end{bmatrix}$$

Estimating θ with expanded basis functions

- Batch gradient descent

$$\theta \leftarrow \theta - \frac{\alpha}{m}\phi(X)^T(\phi(X)\theta - y)$$

- Stochastic gradient descent

$$\theta \leftarrow \theta - \frac{\alpha}{m}(\theta^T\phi(x^{(i)}) - y^{(i)})\phi(x^{(i)})$$

- Closed form solution

$$\theta^* = (\phi^T\phi)^{-1}\phi^T y$$

More on basis functions

ϕ is not limited to one dimensional x's. In fact, we can construct basis function expansions of the form $\phi : \Re^d \rightarrow \Re^p$ where $p > d$. For example

$$(x_1, x_2) \rightarrow (1, x_1, x_2, x_1 x_2, x_1^2, x_2^2)$$

Models will be of the form

$$h_\theta((x_1, x_2)) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2$$

We can use gradient descent of θ with x replaced by $\phi(x)$.

Controlling model complexity

- Lower values of p yield stabler estimates of θ, model could potentially underfit.
- Higher values of p yield higher variance in estimates of θ, model could potentially overfit.
- One way to automatically control model complexity is to penalize large θ_j. This idea is called regularization.

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(y^{(i)} - \theta^T \phi(x^{(i)})\right)^2 + \frac{\lambda}{2m} \sum_{j=1}^{d} \theta_j^2$$

penalize coefficients

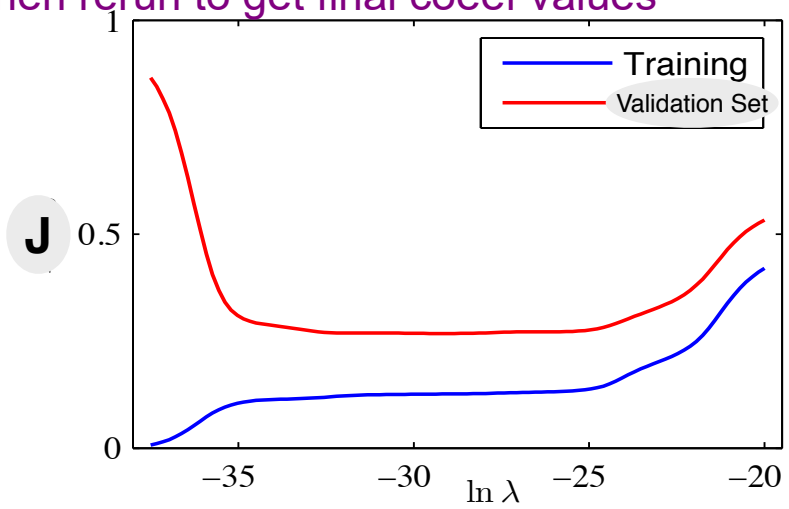
This cost function is an L2 regularizer of θ and estimates of θ which optimize this measure are called ridge regression estimates. Note that θ_0 is not penalized. λ is a regularization parameter whose value is estimated using a validation set.

Ridge regression or L2 penalties on θ

- We add a term to the cost function $\frac{\lambda}{m}\sum_{j=1}^{d}\theta_j^2$ which has the effect of driving the θ components to zero. The λ term determines the relative importance of the empirical loss term and the penalty term on θ.

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \theta^T\phi(x^{(i)})\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{d}\theta_j^2$$

original unregularized, find value of lambda,
then rerun to get final coeef values



Ridge regression estimators

- Batch gradient descent

$$\theta_j \leftarrow \theta_j - \alpha \left[\frac{1}{m} \phi(X)^T (\phi(X)\theta - y) + \frac{\lambda}{m} \theta_j \right] \quad 1 \leq j \leq d$$

$$\theta_j \leftarrow \theta_j - \alpha \left[\frac{1}{m} \phi(X)^T (\phi(X)\theta - y) \right] \quad j = 0$$

- Stochastic gradient descent

$$\theta_j \leftarrow \theta_j - \alpha \left[\frac{1}{m} (\theta^T \phi(x^{(i)}) - y^{(i)}) \phi(x^{(i)}) + \frac{\lambda}{m} \theta_j \right] \quad 1 \leq j \leq d0$$

$$\theta_j \leftarrow \theta_j - \alpha \left[\frac{1}{m} (\theta^T \phi(x^{(i)}) - y^{(i)}) \phi(x^{(i)}) \right] \quad j = 0$$

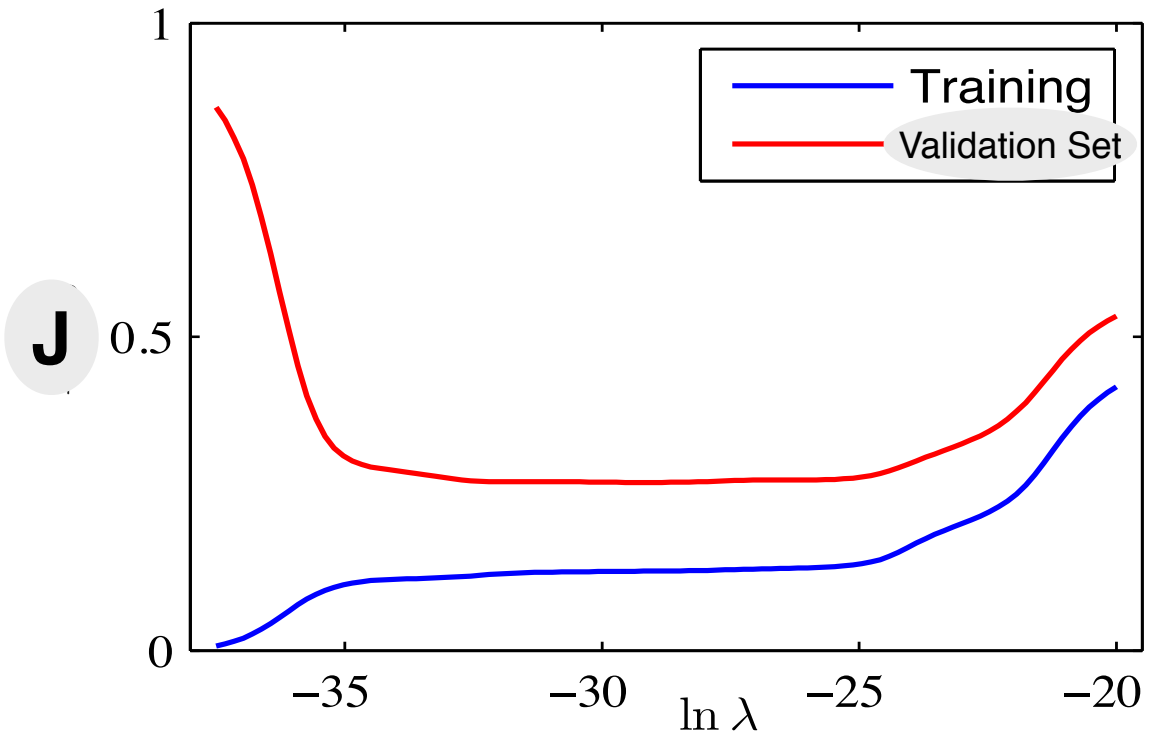
- Closed form solution

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

Regularization

Regularization allows complex models to be trained on data sets of limited size without severe overfitting, essentially by limiting effective model complexity,

Choosing regularization parameter λ



We sweep the λ parameter space on the log scale with a set-aside portion of the training data called the validation set. The training data minus the validation data is used for parameter estimation. The set-aside test data is used for final model evaluation.

Lasso regularization

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(y^{(i)} - \theta^T\phi(x^{(i)}))^2 + \frac{\lambda}{2m}\sum_{j=1}^{d}|\theta_j|$$

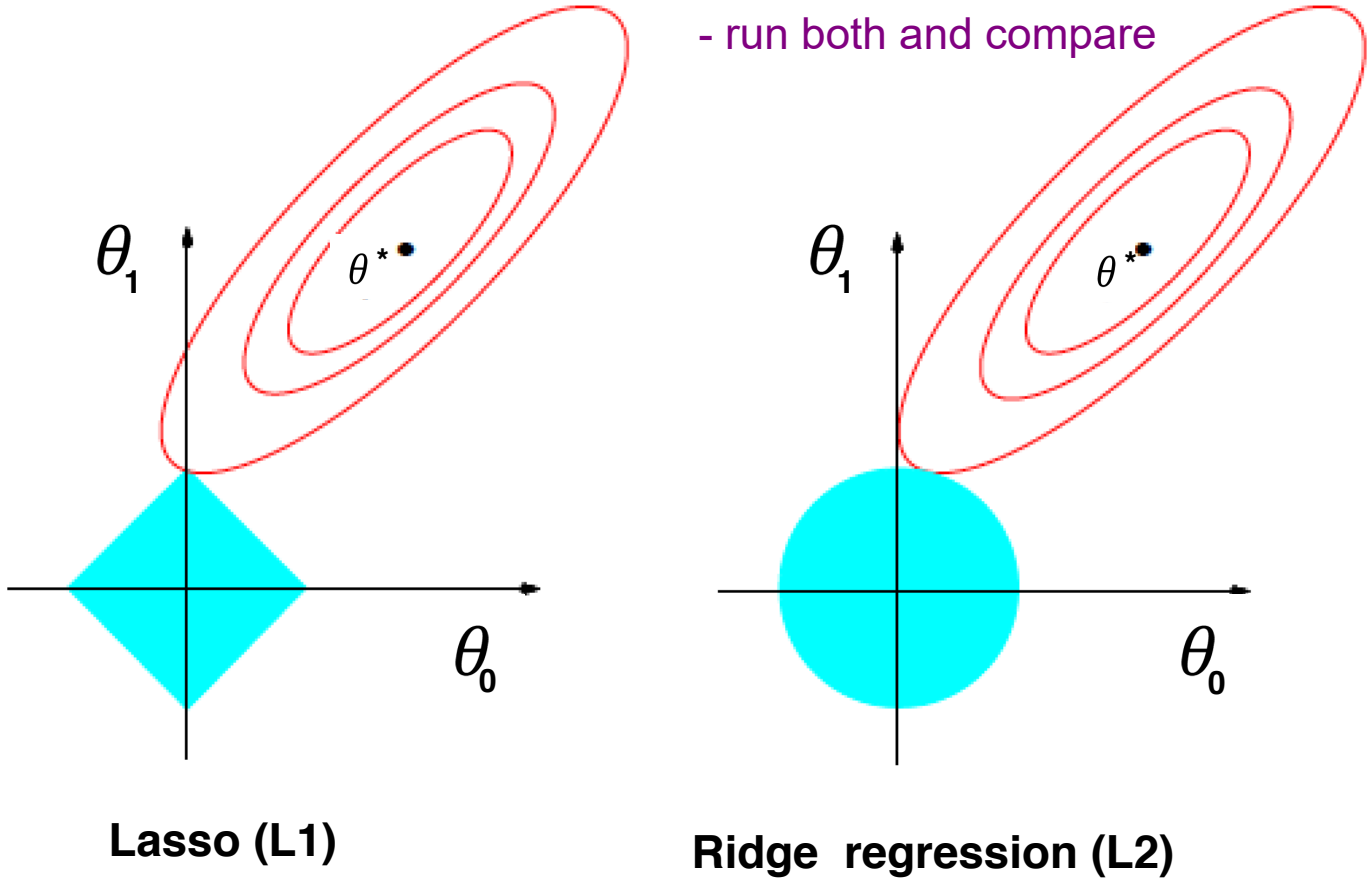
- The penalty term on θ is an L1 norm, unlike the L2 norm used in ridge regression.
- Lasso regularization also drives θ components to zero, just like ridge regression. However, the L1 penalty term drives many θ components to exactly zero, especially when the regularization parameter λ is large.
- Lasso yields sparse models that only involve a subset of the features of X. Thus, lasso performs automatic feature selection.

Ridge regression and lasso

- sparse

- performance

- run both and compare



Lasso (L1)

Ridge regression (L2)

It is hard to know *a priori* which regularization technique will yield better models – so best to run both and compare using cross-validation.

Maximum A Priori (MAP) estimation of θ

$$P(\theta|\mathcal{D}) \propto P(\mathcal{D}|\theta)P(\theta)$$

- $P(\mathcal{D}|\theta)$ is the likelihood of the data $\mathcal{D}$ given the parameter θ
- $P(\theta)$ is the prior distribution on the parameter θ
- MAP estimation of θ is a Bayesian update on the prior distribution of θ using the data $\mathcal{D}$.

MAP estimation of θ

To calculate $P(\mathcal{D}|\theta)$, we assume a generative model of the data

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}, \epsilon^{(i)} \sim N(0, \sigma^2)$$

Then,

$$P(\mathcal{D}|\theta; \sigma^2) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

The prior distribution on θ is a multivariate Gaussian – this is a conjugate prior. Gaussian bec. G * G = G

$$P(\theta) = N(\theta|\mathbf{0}, \alpha^2 I)$$

MAP estimation of θ (contd.)

$$
\begin{aligned}
P(\theta|\mathcal{D}) &\propto P(\mathcal{D}|\theta)P(\theta) \\
&= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&\quad \left(\frac{1}{2\pi\alpha^2}\right)^{\frac{\alpha+1}{2}} exp\left(-\frac{\theta^T\theta}{2\alpha^2}\right)
\end{aligned}
$$

$$
\begin{aligned}
\theta_{MAP} &= argmax_\theta P(\theta|\mathcal{D}) \\
&= argmax_\theta \, log P(\theta|\mathcal{D})
\end{aligned}
$$

maximize posterior prob.

Lab 1: Linear regression and regularization

We will use `sklearn`, `numpy`, `pandas`, `matplotlib` to understand the workflow in building and deploying learning models.

- Data collection and splits
  - read in and visualize simple (x, y) data
  - split data into training and test sets
- Estimate, evaluate, and visualize a linear regression model
  - build unregularized linear models on training data
  - evaluate model on test data
- Iterate model building: building more expressive linear models
  - expand basis functions and learn more expressive multivariate linear models
  - guarding against overfitting: ridge and lasso regression
- Understand and deploy model

Locally weighted linear regression

Consider a variation on $J(\theta)$ which weights training data points based on distance to a point x

$$J_{local}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} w^{(i)} (y^{(i)} - \theta^T \phi(x^{(i)}))^2$$

where

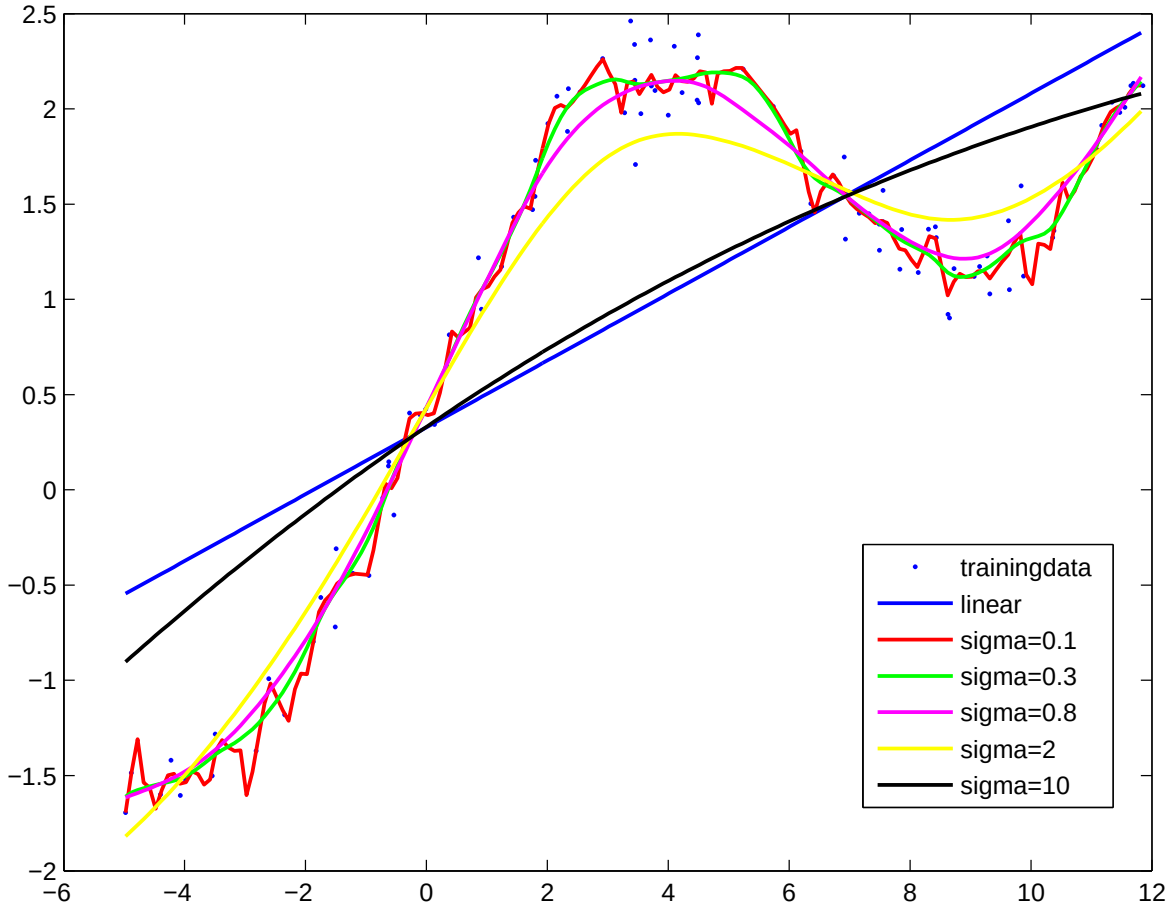
$$w^{(i)} = exp\left(- \frac{(x - x^{(i)})^T (x - x^{(i)})}{2\sigma^2} \right)$$

σ is a bandwidth parameter which controls the sphere of influence around $x^{(i)}$.

Consider the following procedure for predicting y for a new x.

- Calculate $w^{(i)}$ for $1 \leq i \leq m$.
- Solve for θ which minimizes $J_{local}(\theta)$.
- Predict $y = \theta^T x$.

Locally weighted regression



Questions

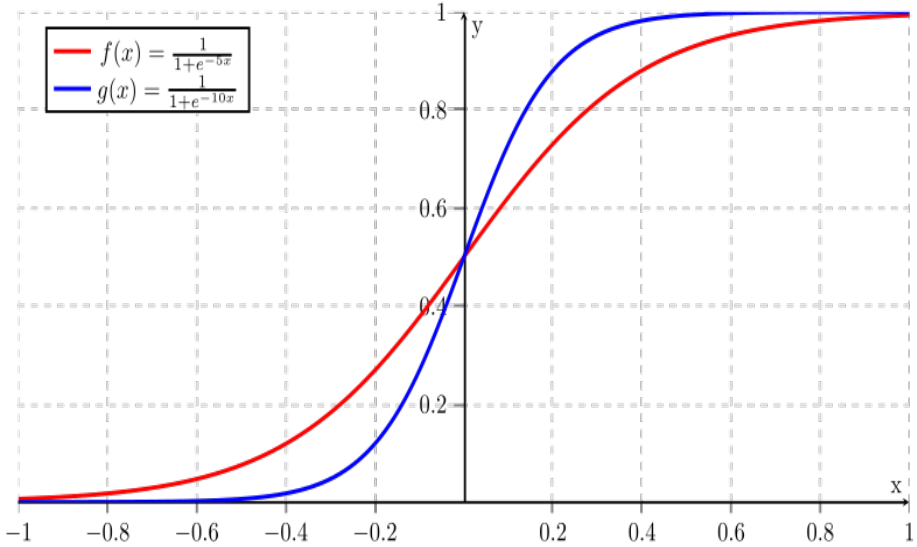
- Is locally weighted regression a parametric method or a non-parametric method?
- How can you control overfitting in locally weighted regression?
- When is it appropriate to use locally weighted regression?

Linear models for classification

Given:

- training data $\mathcal{D} = \{(x^{(i)}, y^{(i)}) | 1 \leq i \leq m, x^{(i)} \in \Re^d, y^{(i)} \in \{0, 1\}\}$ drawn i.i.d. from a fixed, but unknown distribution.
- parametric function class $h_\theta : \Re^{d+1} \rightarrow \Re$ to model posterior distribution $P(y = 1|x)$.

The sigmoid function g



$$h_\theta(x) = g(\theta^T x) = g(\theta_0 + \theta_1 x_1 + \ldots + \theta_d x_d)$$

$$P(y = 1|x) = h_\theta(x) = \frac{1}{1 + exp(-\theta^T x)}$$

Linear decision boundaries

Decision rule:

$$\text{if } g(\theta^T x) \geq 0.5 \text{ then } y = 1$$

$$\text{if } g(\theta^T x) < 0.5 \text{ then } y = 0$$

Since $g(\theta^T x) \geq 0.5$ if and only if $\theta^T x \geq 0$, the decision boundary between $y = 1$ and $y = 0$ is characterized by the hyperplane

$$\theta^T x = 0$$

Note further that the log odds ratio is simply

$$log \frac{P(y = 1|x)}{P(y = 0|x)} = log \frac{g(\theta^T x)}{1 - g(\theta^T x)} = \theta^T x$$

Fitting θ by the maximum likelihood principle

$$
\begin{aligned}
P(y|x) &= [P(y=1|x)]^{y}[P(y=0|x)]^{1-y} \\
P(\mathcal{D}|\theta) &= \prod_{i=1}^{m}[P(y^{(i)}=1|x^{(i)})]^{y^{(i)}}[P(y^{(i)}=0|x^{(i)}]^{1-y^{(i)}} \\
logP(\mathcal{D}|\theta) &= \sum_{i=1}^{m} y^{(i)} log[P(y^{(i)}=1|x^{(i)})] + (1-y^{(i)}) log[P(y^{(i)}=0|x^{(i)}] \\
&= \sum_{i=1}^{m} y^{(i)} log[h_\theta(x^{(i)}] + (1-y^{(i)}) log[1-h_\theta(x^{(i)})]
\end{aligned}
$$

Find θ to maximize log-likelihood of the data or minimize the negative log likelihood of the data.

"cross entropy function"

$$\theta_{MLE} = argmin_\theta -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} log[h_\theta(x^{(i)}] + (1-y^{(i)}) log[1-h_\theta(x^{(i)})]$$

The logistic regression model

out of order:
see next page then go back to this page

Given:

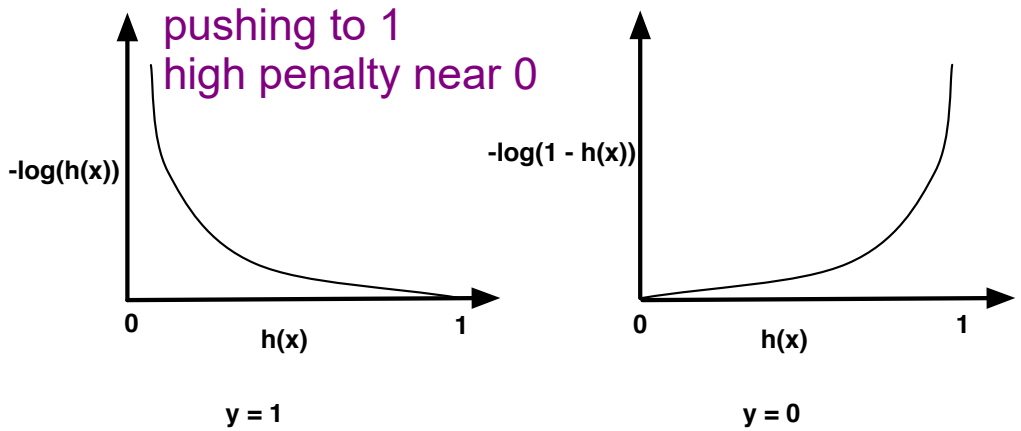
- training data
  $\mathcal{D} = \{(x^{(i)}, y^{(i)}) | 1 \leq i \leq m, x^{(i)} \in \Re^d, y^{(i)} \in \{0, 1\}\}$ drawn i.i.d. from a fixed, but unknown distribution.
- parametric function class $h_\theta : \Re^{d+1} \rightarrow \Re$ where $P(y = 1|x) = g(h_\theta(x))$.
- The cross-entropy function

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} y^{(i)} log[h_\theta(x^{(i)}] + (1 - y^{(i)}) log[1 - h_\theta(x^{(i)})]$$

Find $\theta_{MLE} = argmin_\theta J(\theta)$ which minimizes the negative log likelihood of $\mathcal{D}$ given θ.
The prediction for a new x is simply $g(\theta_{MLE}^T x)$ which is the probability of x belonging to class 1.

Understanding the cross entropy function

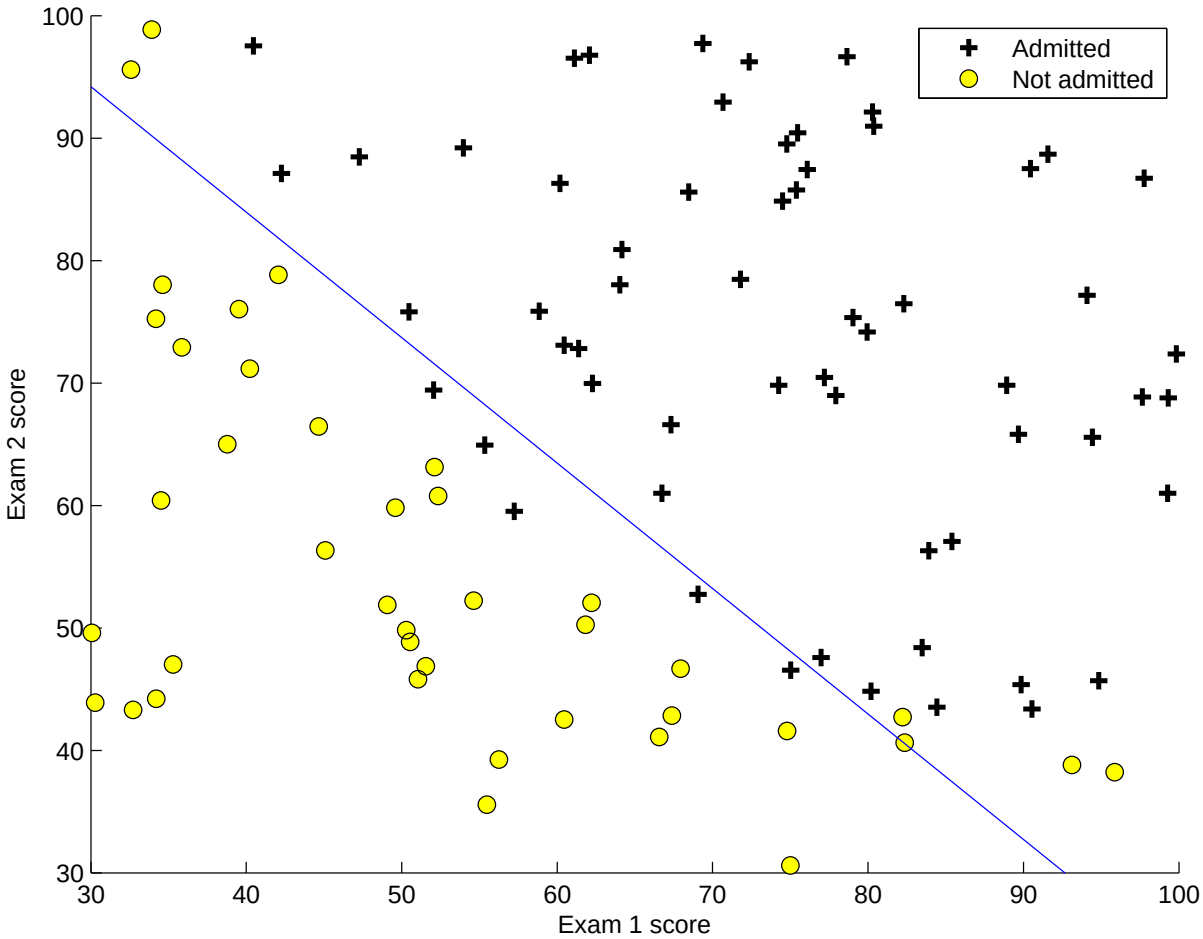
$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} log[h_\theta(x^{(i)}] + (1 - y^{(i)}) log[1 - h_\theta(x^{(i)})]$$



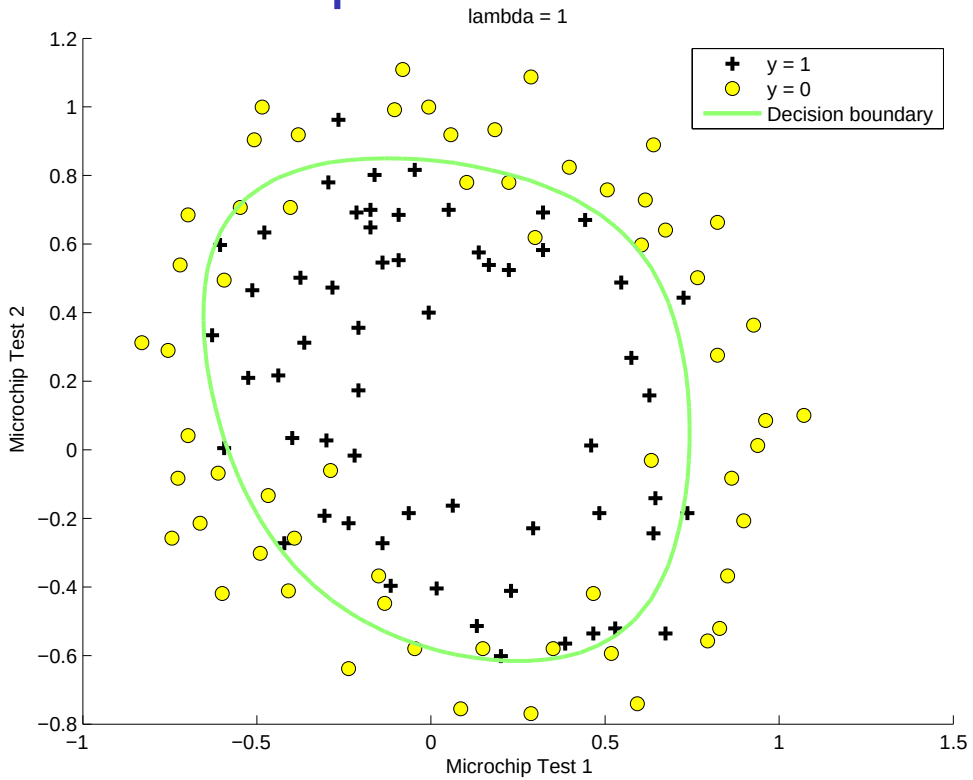
- For $y = 1$, the further $h(x)$ is from 1, the first term of J increases.
- For $y = 0$, the further $h(x)$ is from 0, the second term of J increases.
- $J(\theta)$ is a convex function and has a global minimum. Standard optimization algorithms (such as conjugate gradient) can compute the optimal θ_{MLE}.

Logistic regression example

notebook in lab2



Logistic regression example: basis function expansion



Each (x_1, x_2) is mapped into a 15-dimensional space of all sixth order polynomial combinations of the original features. The projection of the decision boundary in the original two-dimensional space is shown.

Regularizing logistic regression

Logistic regression is popular because it can be explained, even if not the best performing model.

- L2 regularizer

$$J_{reg}(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} log[h_\theta(x^{(i)}] + (1-y^{(i)}) log[1-h_\theta(x^{(i)})] + \frac{\lambda}{m}\sum_{j=1}^{d}\theta_j^2$$

- L1 regularizer

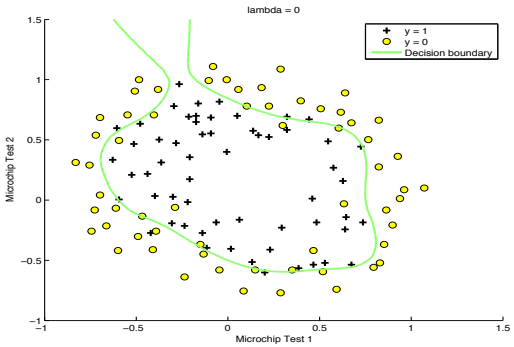
$$J_{reg}(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} log[h_\theta(x^{(i)}] + (1-y^{(i)}) log[1-h_\theta(x^{(i)})] + \frac{\lambda}{m}\sum_{j=1}^{d}|\theta_j|$$

Use crossvalidation as before to select the best value of the regularization parameter λ.

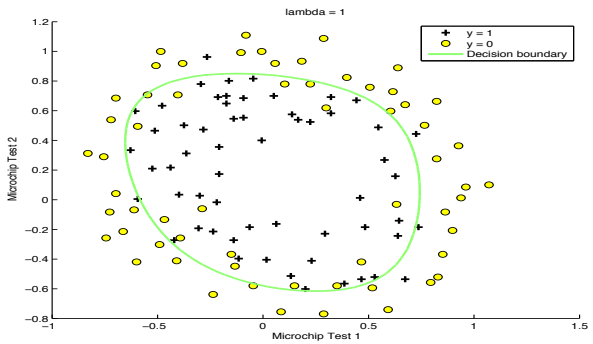
Varying regularization parameter λ

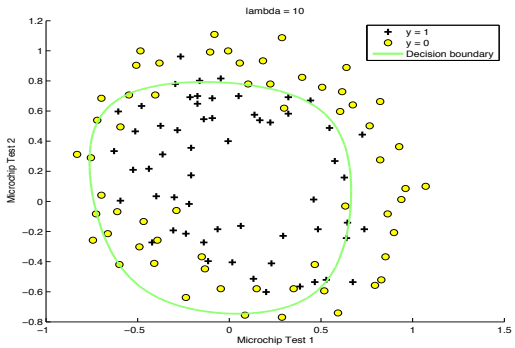
lambda 1 does a good job, an increase in lambda pushed towards axes

lambda = 0

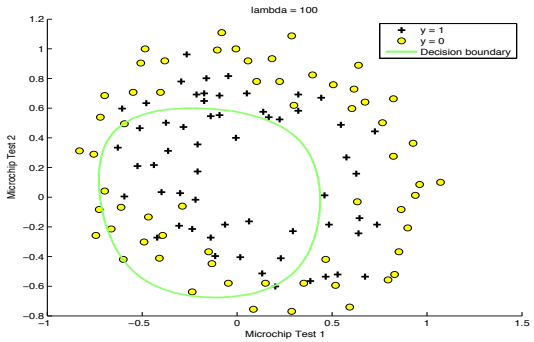


lambda =1





lambda = 10



lambda = 100

Generative models for classification

- Given $\mathcal{D} = \{(x^{(i)}, y^{(i)}) | 1 \leq i \leq m, x^{(i)} \in \Re^d, y^{(i)} \in \{0, 1\}\}$, we will learn the full joint distribution $P(xy)$.
- Since $P(xy) = P(x|y)P(y)$, we can learn the joint distribution by estimating the component distributions
  - Prior probabilities: $P(y = 1)$, $P(y = 0)$
  - Class conditional densities: $P(x|y = 1)$, $P(x|y = 0)$
- Parametric generative models make assumptions about the form of the prior probability and class conditional densities and estimate their parameters from $\mathcal{D}$.
- To classify a new example x, we use Bayes Rule with the learned distributions.

$$P(y = 1|x) = \frac{P(x|y = 1)P(y = 1)}{\sum_{y' \in \{0,1\}} P(x|y')P(y')}$$

Gaussian Discriminant Analysis

- This parametric generative classification method makes the following assumptions.

$$\begin{aligned} y &\sim Bernoulli(\phi) \\ x|y=0 &\sim \mathcal{N}(\mu^0, \Sigma) \\ x|y=1 &\sim \mathcal{N}(\mu^1, \Sigma) \end{aligned}$$

- The parameters ϕ, μ^0, μ^1, Σ are estimated by maximizing the log likelihood of the data $\mathcal{D}$ given these parameters.

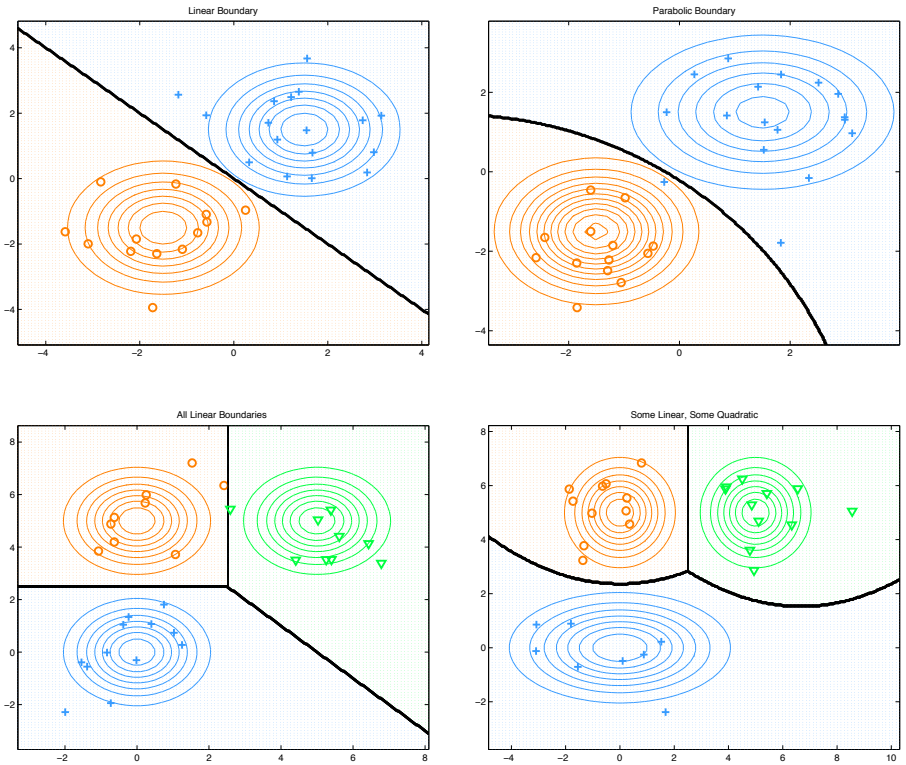
$$\begin{aligned} \mathcal{L}(\mathcal{D}) &= \prod_{i=1}^{m} P(x^{(i)}y^{(i)}; \phi, \mu^0, \mu^1, \Sigma) \\ &= \prod_{i=1}^{m} P(y^{(i)}; \phi) P(x^{(i)}|y^{(i)}; \mu^0, \mu^1, \Sigma) \\ &= \prod_{i=1}^{m} \phi^{y^{(i)}} (1-\phi)^{(1-y^{(i)})} [\mathcal{N}(x^{(i)}|\mu^1, \Sigma)]^{y^{(i)}} [\mathcal{N}(x^{(i)}|\mu^0, \Sigma)]^{(1-y^{(i)})} \end{aligned}$$

Estimation equations for Gaussian Discriminant Analysis

- We differentiate $log(\mathcal{L}(\mathcal{D}))$ with respect to each of the parameters and set the resulting derivative to zero. These equations can be solved in closed form to yield the following:

$$\begin{aligned}
\phi &= \frac{1}{m}\sum_{i=1}^{m} y^{(i)} \\
\mu^1 &= \frac{\sum_{i=1}^{m} y^{(i)} x^{(i)}}{\sum_{i=1}^{m} y^{(i)}} \\
\mu^0 &= \frac{\sum_{i=1}^{m} (1 - y^{(i)}) x^{(i)}}{\sum_{i=1}^{m} (1 - y^{(i)})} \\
\Sigma &= \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu^{y^{(i)}})(x^{(i)} - \mu^{y^{(i)}})^T
\end{aligned}$$

Decision boundaries for Gaussian Discriminant Analysis



Linear decision boundaries when pooled covariance matrix Σ is used, quadratic boundaries when each class has its own covariance matrix.

Gaussian Discriminant Analysis and Logistic Regression

- GDA is a generative model which estimates the joint distribution $P(x, y)$, while logistic regression is a discriminative method which learns a parametric form of the posterior probability $P(y = 1|x)$. They make different assumptions about the origin of $\mathcal{D}$.
- Both yield linear boundaries (generally different for the same data sets).
- When assumptions made about the data hold, GDA is asymptotically efficient – it learns the best classifier with the least amount of data.
- Logistic regression is less sensitive to assumptions about $\mathcal{D}$, but it needs more data than GDA to build accurate models.

Naive Bayes models

Multivariate distributions of the form $P(x|y)$ are difficult to handle, both analytically and computationally. Naive Bayes models are factored generative models that make very strong conditional independence assumptions on the different components x_j of the input.

$$P(x|y) = \prod_{j=1}^{d} P(x_j|y)$$

Instead of dealing with a d dimensional distribution, we will compute with d one-dimensional distributions. The model works well, particularly for discrete x, even when the conditional independence assumption does not hold. It is a baseline method to implement for every data analytics problem.

Generative Naive Bayes Models

Gaussian Naive Bayes model

$$\begin{aligned} y &\sim \textit{Bernoulli}(\phi) \\ x|y=0 &\sim \mathcal{N}(\mu^0, \Sigma) \\ x|y=1 &\sim \mathcal{N}(\mu^1, \Sigma) \end{aligned}$$

Bernoulli Naive Bayes

$$\begin{aligned} y &\sim \textit{Bernoulli}(\phi) \\ x|y=0 &\sim \textit{Bernoulli}(\phi_j^0) \\ x|y=1 &\sim \textit{Bernoulli}(\phi_j^1) \end{aligned}$$

2 step process for estimating parameters of a Naive Bayes Model

- Construct the log likelihood function $l(\mathcal{D})$ conditioned on the parameters.
- Solve for the parameters in closed form by setting first order partial derivatives of the log likelihood function to zero.

calc mean and std dev for each class
naive bayes imagenet 1000 classes 224 x 224 x 3 rbg x 2 param

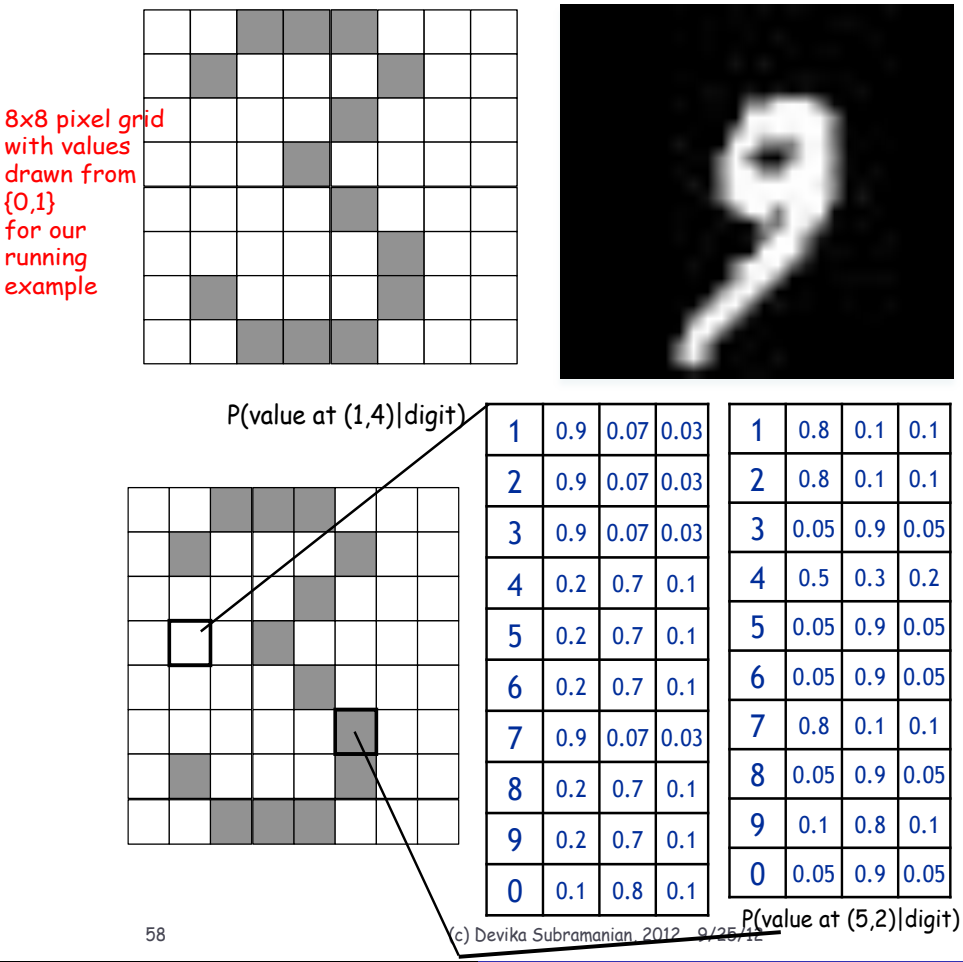
Estimation equations for Bernoulli Naive Bayes

- We differentiate $log(\mathcal{L}(\mathcal{D}))$ with respect to each of the parameters and set the resulting derivative to zero. These equations can be solved in closed form to yield the following:

$$\begin{aligned}
\phi &= \frac{1}{m}\sum_{i=1}^{m} y^{(i)} \\
\phi_j^1 &= \frac{\sum_{i=1}^{m} y^{(i)} x_j^{(i)}}{\sum_{i=1}^{m} y^{(i)}} \\
\phi_j^0 &= \frac{\sum_{i=1}^{m} (1-y^{(i)}) x_j^{(i)}}{\sum_{i=1}^{m} (1-y^{(i)})}
\end{aligned}$$

Digit recognition using the Bernoulli Naive Bayes model

- 10 classes, $y \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- inputs are images of size 28x28 pixels, $x \in \{0, 1, 2\}^{28 \times 28}$



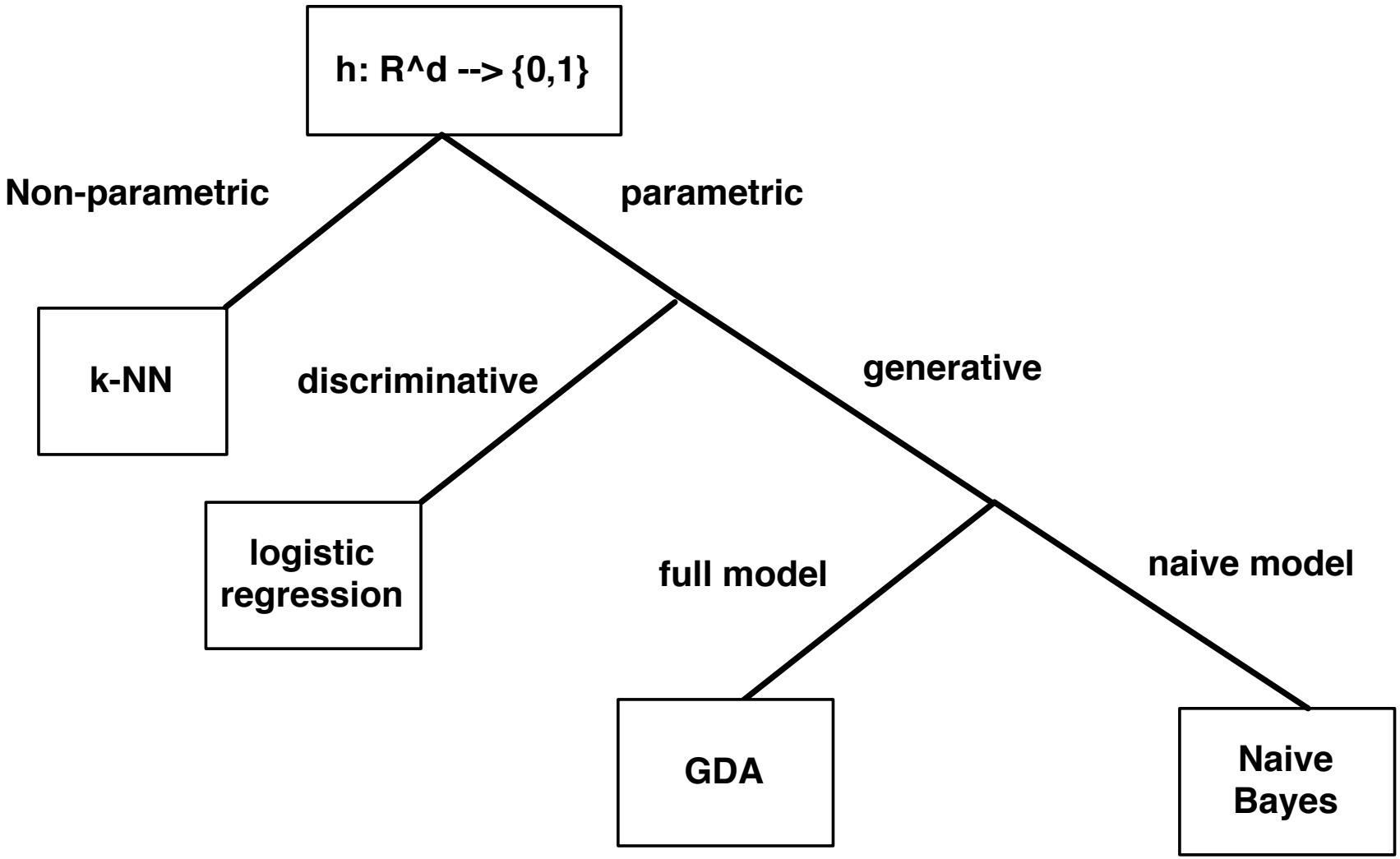
build model by counting

Regularizing Naive Bayes models

- Bernoulli Naive Bayes models are estimated using counts.
  - For example: $P(y = k) =$ the fraction of the training set with label k.
  - But what if there are no examples of digit k in the training set? Then, $P(y = k) = 0$.
  - Laplace smoothing: add a constant c to all counts. So if the count of digit k is zero and there are m examples, then $P(y = k) = \frac{c}{m+10c}$.

    LaPlace wanted to calculate prob that the sun would rise tomorrow
    d / (d + 1)

Recap of binary classification



Evaluating binary classifiers: confusion matrix

		Predicted Class: Yes	Predicted Class: No
Actual Class	Yes	TP	FN
	No	FP	TN

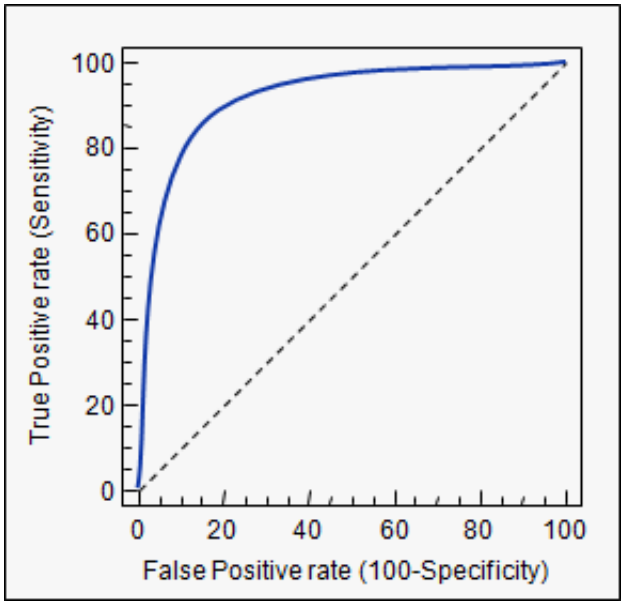
- TP = true positives
- TN = true negatives
- FP = fale positives
- FN = false negatives

Evaluating binary classifiers: metrics

		Predicted Class: Yes	Predicted Class: No
Actual Class	Yes	TP	FN
	No	FP	TN

- accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$
- specificity $= P(y_{pred} = 0 | y = 0) = \frac{TN}{FP+TN}$
- sensitivity $= P(y_{pred} = 1 | y = 1) = \frac{TP}{FN+TP}$

Receiver Operator Characteristic Curve



Each point on the Receiver Operator Characteristic (ROC) curve represents specificity and sensitivity as a function of classification threshold. The area under the ROC curve is a measure of classifier quality. Higher numbers indicate a better classifier (range is 0.5-1).

Softmax for multiclass classificaton

An extension of logistic regression for handling more than two classes. The parameter vector θ is a $d \times K$ matrix instead of a vector of length d.

$$\theta = \begin{bmatrix} \uparrow & \cdots & \uparrow \\ \theta^{(1)} & \cdots & \theta^{(K)} \\ \downarrow & \cdots & \downarrow \end{bmatrix}$$

$$P(y = k|x;\theta) = \frac{e^{\theta^{(k)^T} x}}{\sum_{k=1}^{K} e^{\theta^{(k)^T} x}}$$

Maximum likelihood method for estimating θ using this specific $P(y|x;\theta)$ for a given data set.

Lab 2: Linear models for classification

- Data collection and splits
  - read in and visualize two-class data
  - split data into training and test sets
- Estimate, evaluate, and visualize a logistic regression model
  - build unregularized model on training data
  - evaluate model on test data
- Iterate model building: building more expressive logistic models
  - expand basis functions and learn more expressive multivariate logistic models
  - guarding against overfitting: using lasso
- Explore model space/model selection: GDA and softmax regression
  - estimate and visualize gaussian discriminant analysis models
  - estimate and visualize softmax models
- Understand and deploy model