Southeast Asia Machine Learning School

Machine Learning Basics

Wee Sun Lee
School of Computing
National University of Singapore
leews@comp.nus.edu.sg

July 2019

Outline

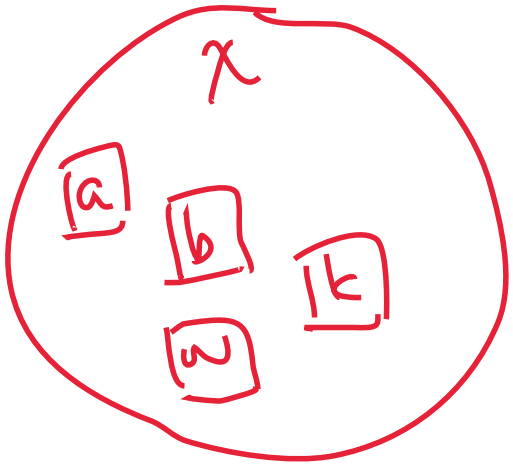
1. Loss Function
2. Maximum Likelihood
3. Model Selection
4. Features

Definitions

We will use the statistical learning framework for this talk. We first formulate supervised learning.

- **Domain set**: An arbitrary set $\mathcal{X}$. Usually represented using features, e.g. vector $\mathbf{x}$ of pixel values for an image, often also called attributes or covariates. Domain points are referred to as *instances* and $\mathcal{X}$ as *instance space*.

Example : handwritten character recognition



- **Label set**: Usually denoted $\mathcal{Y}$. A discrete set for classification, e.g. $\{0, 1\}$ or $\{-1, 1\}$. Real valued for regression. Also called response variable.

Handwritten character example: $\mathcal{Y} = \{a, b, c, \ldots\}$

- **Training data**: The training data or training set, $S = ((x_1, y_1), \ldots, (x_m, y_m))$ is a finite sequence of pairs in $\mathcal{X} \times \mathcal{Y}$. This is the input to the learning algorithm.

$S = ((\boxed{c}, c), (\boxed{d}, d), (\boxed{k}, k), \ldots\}$

- **Algorithm's output**: The output of the learning algorithm is a function $h : \mathcal{X} \to \mathcal{Y}$. This is often called the *predictor*, or *hypothesis*, or *classifier* in the case of classification problems.

h may be a linear predictor
or a neural networks

- **Data generation model**: We assume that S is sampled from a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$.
$\mathcal{D}$ is a joint distribution over the domain points and labels.

Example: S is sampled from $D(x,y)$

$x \in \{\boxed{a}, \boxed{b}, \dots\}$

$y \in \{a, b, c \dots\}$

Not: $D(x)$

$D(y)$

- Continued ...

  It is sometimes convenient to decompose the distribution into the *marginal distribution* $\mathcal{D}_x$ over $\mathcal{X}$

  Marginal $D_x(x=\boxed{a}) = \sum_{y=a}^{z} D(x=\boxed{a}, y)$

  and the *conditional distribution* $\mathcal{D}_{y|x}(y|x)$ of y given x.

  $$D_{y|x}(y=a \mid x=\boxed{a}) = \frac{D(x=\boxed{a}, y=a)}{D_x(x=\boxed{a})}$$

  In the case of classification/regression with no noise, $\mathcal{D}_{y|x}(y|x) = f(x)$ is a deterministic function of x.

- **Measures of success**: We will use a loss function to help measure our success. Given a set $\mathcal{H}$ of hypotheses of models, and a domain $\mathcal{Z}$, let ℓ be a function from $\mathcal{H} \times \mathcal{Z}$ to non-negative real numbers $\ell : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}_+$. We call such a function a **loss function**.

$Z = X \times Y$

Example : binary classification

$$\ell(h, (x,y)) = 0 \text{ if } h(x) = y$$
$$= 1 \text{ if } h(x) \neq y$$

The **risk function** is the expected loss of the hypothesis,

$$L_{\mathcal{D}}(h) = E_{z \sim \mathcal{D}}[\ell(h, z)].$$

We are interested in finding a hypothesis h that has small risk, or expected loss. Goal of learner: $\min_h$ over $L_D(h)$

Some commonly used loss functions:

- The **0-1 loss** measures the misclassification error in classification

$$\ell_{0-1}(h,(x,y)) = \begin{cases} 0 & \text{if } h(x) = y \\ 1 & \text{if } h(x) \neq y. \end{cases}$$

More generally

$$\ell(h,(x,y)) = a_{y,y'} \text{ if } h(x) = y'$$

- The **square loss** is commonly used for regression

$$\ell_{sq}(h,(x,y)) = (h(x) - y)^2.$$

Another example: absolute loss

$$\ell(h,(x,y)) = |h(x) - y|$$

Empirical Risk Minimization

- The learner does not know $\mathcal{D}$ and only have access to the training set S, a sample from $\mathcal{D}$.

$S = ((x_1, y_1), \ldots, (x_m, y_m))$

Don't have $D(x, y)$

- For a predictor $h : \mathcal{X} \to \mathcal{Y}$, we can approximate the expected error by using the training set error

$$L_S(h) = \frac{|\{i \in [m] : h(x_i) \neq y_i\}|}{m},$$

where $[m] = \{1, \ldots, m\}$.

$L_S(h)$ is an approximation of $L_D(h)$

- The training set error can be rewritten using the 0-1 loss

$$L_S(h) = \frac{\sum_{i=1}^{m} \ell_{0-1}(h, (x_i, y_i))}{m}.$$

Can also use other loss functions

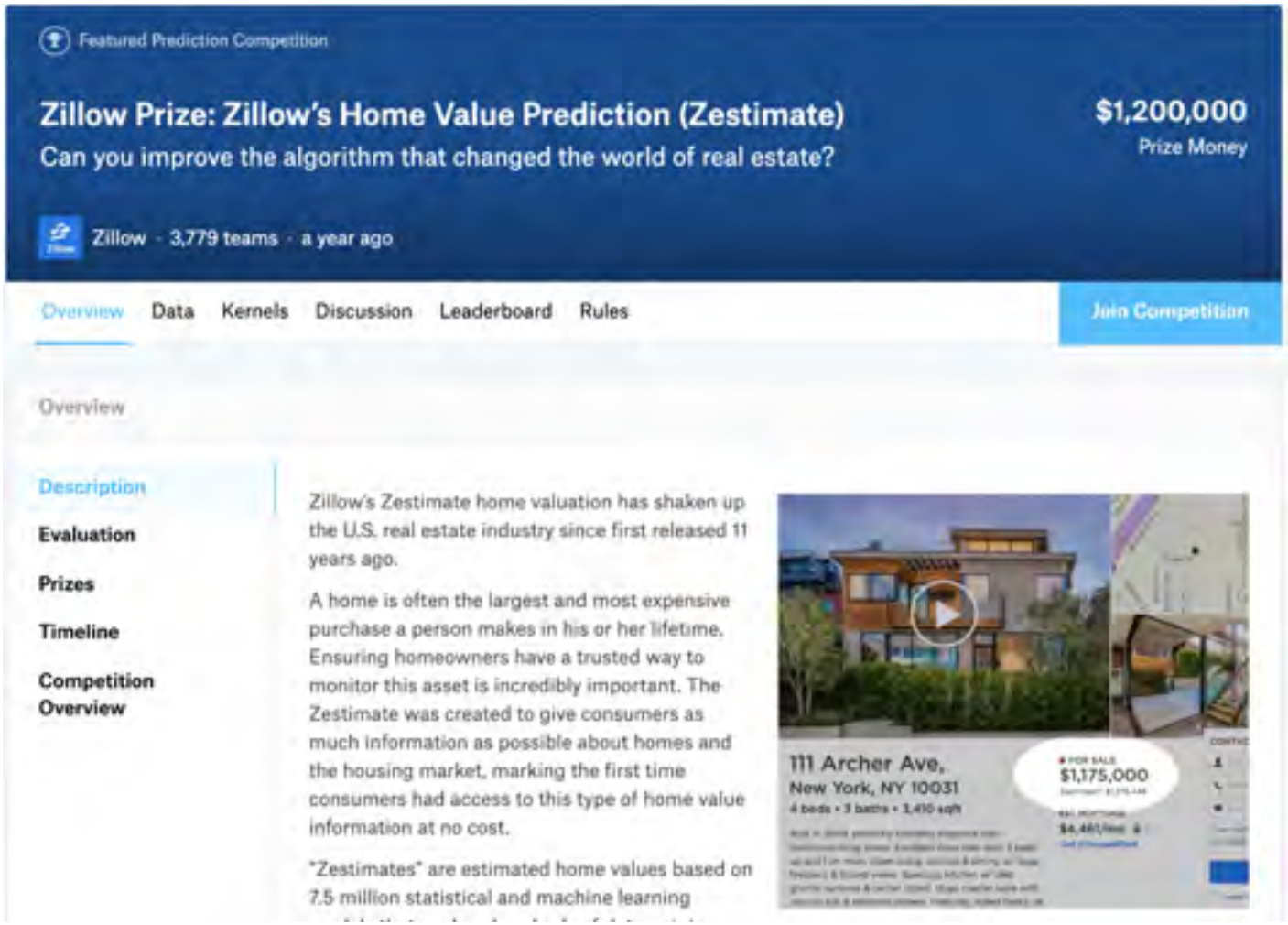
- The training set error is often called the *empirical error* or *empirical risk*. We will us this interchangeably with *training error* or *training loss*.
- Given a hypothesis class $\mathcal{H}$, finding the hypothesis $h \in \mathcal{H}$ that minimizes the empirical risk is a simple learning strategy.

Find $h \in H$ that min $L_S(h)$

- This is often called *empirical risk minimization* (ERM).

and hope it does well for $L_D(h)$

Example: Home Price Prediction



- Predict home price given location, size, number of rooms, etc.
- This is a regression problem.

offset or bias

weight

- Consider linear regression, with square loss.
  - The hypothesis is of the form $h(\mathbf{x}) = w_0 + \sum_{j=1}^{d} w_j x_j$ and the loss function is $\ell_{sq}(h, (\mathbf{x}, y)) = (h(\mathbf{x}) - y)^2$ and the training error is
$$L_S(h) = \frac{\sum_{i=1}^{m} (y_i - h(\mathbf{x}_i))^2}{m}.$$
  - Using matrix notation, we write
$$\mathbf{y} = X\mathbf{w}$$

where

label of example 1

features of example 1

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}, \quad X = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_m^\top \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{21} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & \cdots & x_{md} \end{pmatrix}, \quad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{pmatrix}.$$

← not averaged

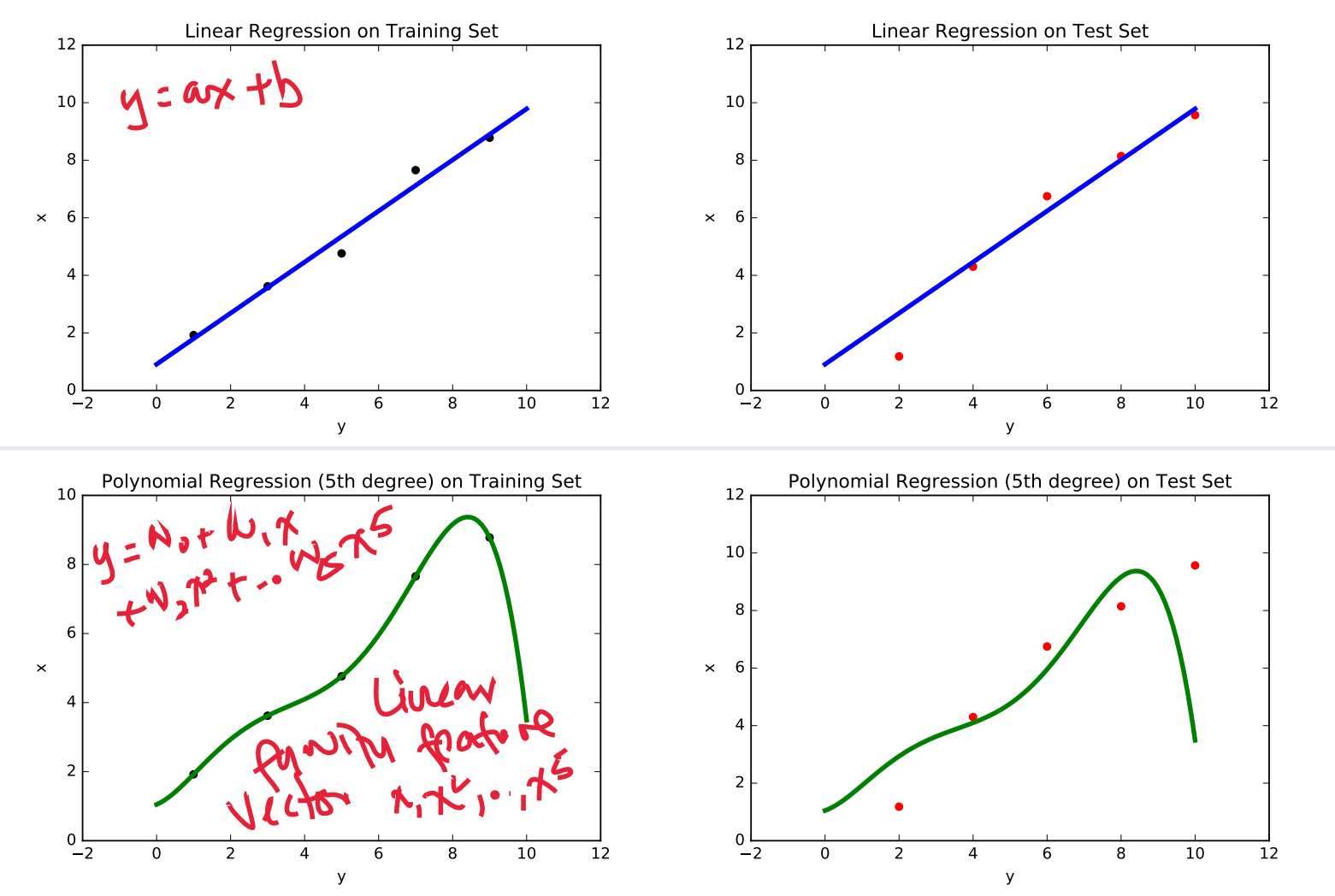
- The training error becomes $(\mathbf{y} - X\mathbf{w})^T(\mathbf{y} - X\mathbf{w})$.
- Differentiating and setting to 0, we get

$$\begin{aligned} & X^T(\mathbf{y} - X\mathbf{w}) = 0 \\ \Leftrightarrow\ & X^T\mathbf{y} - X^TX\mathbf{w} = 0 \\ \Leftrightarrow\ & X^TX\mathbf{w} = X^T\mathbf{y}. \end{aligned}$$

- This gives the solution

$$\mathbf{w} = (X^TX)^{-1}X^T\mathbf{y}.$$

Overfitting



Test set mean square error: 0.52 for linear regression, 8.34 for polynomial regression.

- Minimizing training error is sometimes not sufficient to ensure good performance on unseen data
  - Being able to reproduce performance on unseen data is often called **generalization**. Learning $L_S(h)$, do well on $L_D(h)$
- In the example, a 5th degree polynomial can fit the training data (on odd numbers) perfectly compared to using linear regression, but has much poorer generalization (on even numbers).
- Known as **overfitting**.
- For good generalization, need to control complexity of function class.
- Typically balance training error with complexity control: feature selection, regularization, etc.

Regularization

- For linear regression, it is common to optimize

$$\frac{\sum_{i=1}^{m}(y_i - \mathbf{w}^T\mathbf{x}_i))^2}{m} + \lambda||\mathbf{w}||^2$$

regularizer

Balance training error with complexity

- This is often called *ridge regression* or *penalized least square*.
- The term $||\mathbf{w}||^2$ is called the regularizer.

Exercise 1:

In this experiment, the training and test examples are generated with the function $y = x$ with Gaussian noise added. We fit a linear function and a 10th degree polynomial.

For the 10th degree polynomial, we fit using polynomial regression and then with ridge regression. In scikit learn, ridge regression finds $\min_w ||Xw - y||_2^2 + \alpha||w||_2^2$.

- Run the experiment.
- Change the variable *data_size* to 10 and run it again.

I.I.D. Assumption

- We often make assumptions about the data generation process.
- One common assumption is that the data is **independently and identically distributed (i.i.d.)** according to the distribution $\mathcal{D}$. z_t independent of $z_{t-1}, \ldots, z_1$: $P(z_t | z_{t-1}, \ldots, z_1) = P(z_t)$

Example: Knowing prev characters does not help predict next char

Identically distributed: $P(z_{t+1}) = P(z_t)$

- This is denoted $S \sim \mathcal{D}^m$ where m is the training set size, and $\mathcal{D}^m$ denotes the probability over m-tuples induced by applying $\mathcal{D}$ to pick each element of S independently.
- Under assumptions such as i.i.d., can analyse generalization using statistical learning theory.

Exercise 2: Design and analysis of machine learning algorithms often assume the i.i.d. assumption. Consider the implications for following problem.

- You are given multiple pages of handwritten text.
- You segment the text into characters and label them, giving you a sequence $S = ((x_1, y_1), ...)$, where x_i is a scanned image of a character and y_i is its label.
- When deployed, you expect to also follow the same process, getting a sequence of unlabeled scanned images of characters that need to be labeled.

1. You train a classifier $h(x)$ that depends only on the current scanned image x. Given a large enough training set, do you expect the long term test error to be similar to the training error? If i.i.d. yes, unless overfitting
Real data in this case unlikely iid.
But likely uncorrelated after a while
Long enough training seq likely representative:
training data similar to test data

2. If you do not assume the data to be i.i.d., how can you exploit it? If exploit dependencies, eg learn current image conditional on prev images $p(z_t | z_{t-1}, \dots)$, may get lower training / test error

Outline

1. Loss Function
2. Maximum Likelihood
3. Model Selection
4. Features

Maximum Likelihood

Minimize expected loss $\Leftrightarrow$ maximize likelihood

- Maximum likelihood is a commonly used estimation method in statistics.
- Assume that the data distribution $\mathcal{P}$ is known up to some parameter $h \in \mathcal{H}$.

Example: Coin tossing Bernoulli distribution
Parameter θ, prob of head H
$p(x=H)=\theta$, $p(x=T)=1-\theta$

- The maximum likelihood principle suggests to select the h or θ that maximizes the probability of the data S being observed:

$S=(1,0,0,0,\ldots,1)$

$$h_{ML} = \arg\max_{h \in \mathcal{H}} \mathcal{P}(S|h).$$

- For i.i.d. data $S = (z_1, \ldots, z_m) \sim \mathcal{D}^m$, this becomes:

$$h_{ML} = \arg\max_{h \in \mathcal{H}} \prod_{i=1}^{m} \mathcal{D}(z_i|h).$$

$P(z_1, \ldots, z_m) = \prod_i P(z_i)$

Example: Assume 4 heads, 6 tails in 10 tosses

$\arg\max_\theta P(S|\theta) = \theta^4 (1-\theta)^6$

$\log P(S|\theta) = 4 \log \theta + 6 \log (1-\theta)$

$\frac{d}{d\theta} \log P(S|\theta) = \frac{4}{\theta} - \frac{6}{1-\theta}$

set to zero: $\frac{4}{\theta} = \frac{6}{1-\theta}$

$\theta = \frac{4}{10}$

Usually $\max \log P(S|\theta)$ instead. log is monotonic preserves arg max

- For supervised learning, $z_i = (x_i, y_i)$ and we have

$$h_{ML} = \arg\max_{h \in \mathcal{H}} \prod_{i=1}^{m} \mathcal{D}(y_i | x_i, h) \mathcal{D}(x_i).$$

no h

$\mathcal{D}(y_i, x_i | h) = \mathcal{D}(x_i | h)\, \mathcal{D}(y_i | x_i, h)$

from chain rule of probability $p(x_1, x_2, \ldots, x_m) = p(x_1)\, p(x_2 | x_1) \ldots p(x_m | x_{m-1}, \ldots, x_1)$

- As $\mathcal{D}(x_i)$ does not depend on h, we can drop $\mathcal{D}(x_i)$ from the criterion

$$h_{ML} = \arg\max_{h \in \mathcal{H}} \prod_{i=1}^{m} \mathcal{D}(y_i | h, x_i).$$

- Instead of maximizing the likelihood, it is often more convenient to maximize the log likelihood:

$$h_{ML} = \arg\max_{h \in \mathcal{H}} \sum_{i=1}^{m} \log \mathcal{D}(y_i | h, x_i).$$

Maximum Likelihood and Minimizing Empirical Risk

- For some distributions $\mathcal{D}$, maximizing the log likelihood is equivalent to empirical risk minimization with appropriate loss functions.
- Assume that the observations are generated by a true function plus some additive noise $y_i = h(x_i) + \xi_i$.

$$\mathcal{D}(y_i \mid h, x_i) = \mathcal{D}(\xi_i = y_i - h(x_i))$$

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} \log \mathcal{D}(y_i - h(x_i))$$

- If the density of ξ_i is zero mean Gaussian $\frac{1}{\sqrt{2\sigma^2\pi}}\exp(-\frac{\xi_i^2}{2\sigma^2})$, we get

$$\log \mathcal{D}(y_i|h, x_i) = -\frac{(y_i - h(x_i))^2}{2\sigma^2} + C$$

where C is a constant.

- Maximizing the log likelihood is equivalent to minimizing the empirical risk with the square loss

$$L_S(h) = \frac{\sum_{i=1}^m (y_i - h(x_i))^2}{m} = \frac{\sum_{i=1}^m \ell_{sq}(h, (x_i, y_i))}{m}.$$

- More generally, when an equivalent distribution can be found, empirical risk minimization is equivalent to maximum likelihood when for loss function

$$\ell(h,(x,y)) = -\ln p(y|x,h).$$

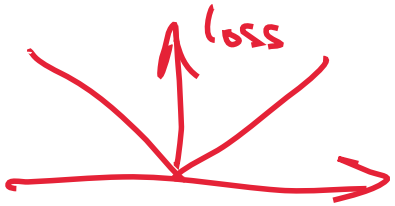
- For regression, assuming $y_i = h(x_i) + \xi_i$, we have

$$\ell(h,(x,y)) = -\ln p_\xi(y - h(x)).$$

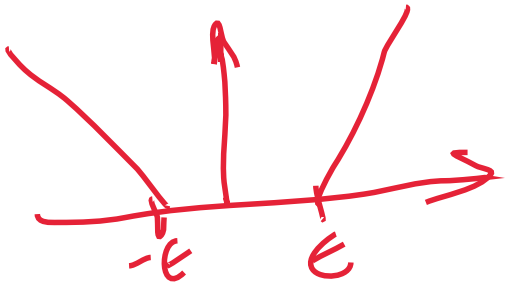
Some other commonly used loss function for regression and their corresponding densities:

	loss function	density model
ϵ-insensitive	$\max(\lvert\xi\rvert - \epsilon, 0) = \lvert\xi\rvert_\epsilon$	$\frac{1}{2(1-\epsilon)} \exp(-\lvert\xi\rvert_\epsilon)$
Laplacian	$\lvert\xi\rvert$	$\frac{1}{2} \exp(-\lvert\xi\rvert)$
Huber's robust loss	$\begin{cases} \frac{1}{2\sigma}(\xi)^2 & \text{if } \lvert\xi\rvert \leq \sigma \\ \lvert\xi\rvert - \frac{\sigma}{2} & \text{otherwise} \end{cases}$	$\propto \begin{cases} \exp\left(\frac{1}{2\sigma}(\xi)^2\right) & \text{if } \lvert\xi\rvert \leq \sigma \\ \exp\left(\lvert\xi\rvert - \frac{\sigma}{2}\right) & \text{otherwise} \end{cases}$

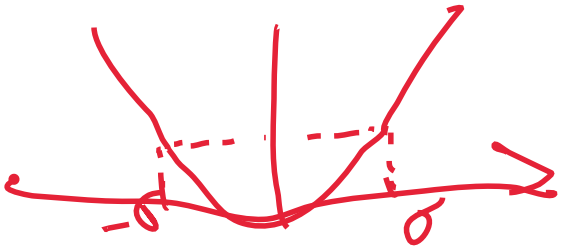
Laplacian



ϵ-insensitive



Huber



Classification Loss Functions

- For classification, we have a finite set of labels. $\mathcal{Y} = \{1, \ldots, K\}$
- Assume that the predictor outputs a probability distribution $q = p(y|x, h)$ over the possible classes.
- We can maximize the likelihood, or correspondingly do ERM using the log loss

$$\ell(h, (x, y)) = -\ln p(y|x, h).$$

- Sometimes called the cross entropy loss.

$y = \{-1, 1\}$

- For binary classification using $h(x) \in (-\infty, \infty)$, we often compose the output of our function $h(x)$ with the logistic (also called sigmoid) function (shown below) to get a probability model $P(y = 1|x, h) = \frac{\exp(h(x))}{1+\exp(h(x))} = \frac{1}{1+\exp(-h(x))}$.

Example: linear fn

$h(x) = \sum_{i=1}^{d} w_i x_i$

$h(x) \in (-\infty, \infty)$

$f(x) = \frac{1}{1+\exp(-h(x))}$

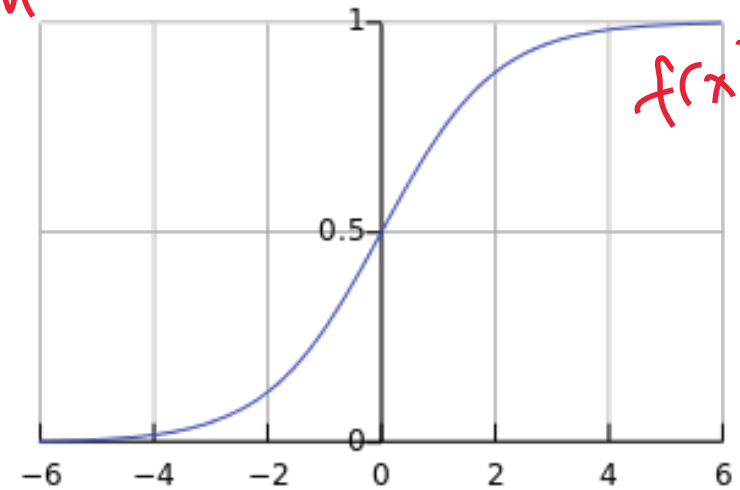


Figure from [6].

- In this model, $P(y = -1|x, h) = 1 - \frac{\exp(h(x))}{1+\exp(h(x))} = \frac{1}{1+\exp(h(x))}$.

$$p(y|x,h) = \begin{cases} \frac{1}{1+e^{-h(x)}} & \text{if } y=1 \\ \frac{1}{1+e^{h(x)}} & \text{if } y=-1 \end{cases}$$

- For $y \in \{1, -1\}$, can write $P(y|x, h) = \frac{1}{1+\exp(-yh(x))}$.
- So log loss can be written as

$$\ell_{log}(h, (x, y)) = -\log P(y|h(x)) = \log(1 + \exp(-yh(x))).$$

logistic loss

- When $h(x)$ is a linear function, this is often called *logistic regression*.
- Neural networks are also often used as $h(x)$.

- This model can be generalized to multi-class classification by using the model $P(y = k|x, h) = \frac{\exp(h_k(x))}{\sum_{i=1}^{K} \exp(h_i(x))}$. $y = \{1, \ldots, k\}$
- When $h_i(x)$ is linear, this is often called *multiclass logistic regression*, *softmax regression* or *maximum entropy classifier*.

Also used with neural nets.

MAP Estimation and Regularization

- Empirical risk minimization or maximum likelihood may overfit the data if the hypothesis class used is too powerful.
- The estimate depends entirely on the data.
- One way to reduce overfitting is to balance the dependence on the data with prior knowledge.
- Instead of maximizing the likelihood

$$h_{ML} = \arg\max_{h \in \mathcal{H}} \prod_{i=1}^{m} \mathcal{D}(z_i|h),$$

in *maximum a posteriori (MAP) estimation*, find the parameter that maximizes the posterior probability $P(h|D)$.

- In MAP estimation, we want to find

$$
\begin{aligned}
h_{MAP} &= \arg\max_{h \in \mathcal{H}} P(h|D) \\
&= \arg\max_{h \in \mathcal{H}} P(D|h)P(h)/Z \\
&= \arg\max_{h \in \mathcal{H}} \prod_{i=1}^{m} \mathcal{D}(z_i|h)P(h)/Z,
\end{aligned}
$$

Independence

In contrast to $\arg\max_h P(D|h)$ in max likelihood

where Z is a constant normalization factor.

Bayes rule: $P(y|x) = \frac{P(x|y)P(y)}{Z}$

Likelihood, Prior, evidence

$Z = \sum_{y'} P(x|y')P(y')$

- Taking log, for the supervised learning case, we get

$$h_{MAP} = \arg\max_{h \in \mathcal{H}} \left(\sum_{i=1}^{m} \log \mathcal{D}(y_i | h, x_i) + \log P(h). \right)$$

prior

- Balance between fitting the data (likelihood) well and fitting the prior well.
- For example, for linear regression, we often put a zero mean Gaussian prior on the weight vector. Assuming additive Gaussian noise, the MAP estimator minimizes a combination of the empirical risk with the square loss and the size of the linear function weights

$$\frac{\sum_{i=1}^{m} (y_i - \mathbf{w}^T \mathbf{x}_i))^2}{m} + \lambda ||\mathbf{w}||^2,$$

$P(w) = \frac{1}{2} e^{-\lambda ||w||^2}$

where $\mathbf{w}$ is the weight vector of the linear regressor.

- This is the *ridge regression* or *penalized least square* described earlier.
- Minimizing a combination of empirical risk and a regularizer is also called regularized loss minimization.
- Under the MAP interpretation, we are maximizing a combination of prior and likelihood functions.
- The regularizer can also be viewed as a measure of complexity, so we are balancing risk minimization with complexity.

Bayesian Estimation

- In Bayesian estimation, instead of selecting a single h, we maintain the posterior distribution over the parameters $P(h|z_1, \ldots, z_m)$.
- This can be used to make optimal prediction (assuming the Bayesian model is correct) for the variable of interest. For example,

Marginalization

$\int_h p(y,h) = p(y)$

$$P(y|x, z_1, \ldots, z_m) = \int_h P(y, h|x, z_1, \ldots, z_m)$$

$$= \int_h P(y|h, x, z_1, \ldots, z_m) \underbrace{P(h|z_1, \ldots, z_m)}_{\text{Posterior}}.$$

Chain rule

Often intractable

Approx. e.g. using variational methods or sampling

- For classification, we would predict with the value y that maximizes $P(y|x, z_1, \ldots, z_m)$ to get optimal prediction.
- Bayesian estimation is often computationally more expensive unless there is special structure that can be exploited (e.g. use of conjugate prior).

Unsupervised Learning

In unsupervised learning, $S = (x_1, \ldots, x_m)$ no labels y_i

- Many (but not all) unsupervised learning problems can be posed as density estimation problems, i.e. learning the distribution of the data. $P_\theta(x)$
- We can often pose the problem of learning the data distribution as a maximum likelihood (or maximum a posteriori) estimation problem.
- For example, we may model data using a mixture of Gaussians and learn using maximum likelihood for clustering. After learning, the means (centers) of the Gaussians can be treated as our cluster centers.

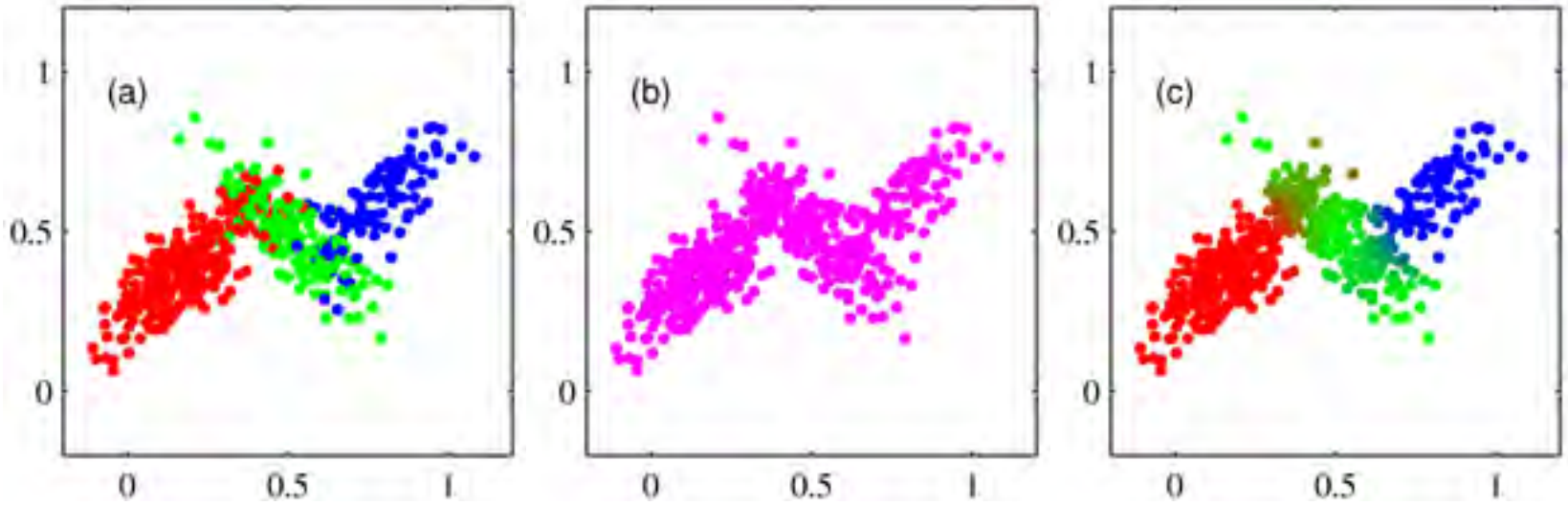


Figure: From [1]. Mixture of 3 Gaussians. Left figure shown in red, green, blue corresponding to the three mixtures. Middle is sample from marginal $p(\mathbf{x})$. Right shows the estimated component for each point using proportions of red, blue and green.

$P(x) = \sum_{y=1}^{k} P(x, y)$

chain rule

In this example, the marginal distribution is

$$P(X = \mathbf{x}) = \sum_{y=1}^{k} P(Y = y) P(X = \mathbf{x} | Y = y)$$

$$= \sum_{y=1}^{k} c_y \frac{1}{(2\pi)^{d/2} |\Sigma_y|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_y)^T \Sigma_y^{-1} (\mathbf{x} - \boldsymbol{\mu}_y)\right).$$

One way to learn c_y, $\boldsymbol{\mu}_y$ and Σ_y is to use maximum likelihood.

Find param values that

$\max \prod_i P(X_i = x_i)$

Another example

- A common type of model is an autoregressive model: From chain rule of probability

Chain rule

$$P(x_1, \ldots, x_m) = P(x_1)P(x_2|x_1)P(x_3|x_2, x_1) \cdots P(x_m|x_{n-1}, \ldots, x_1).$$

For example, language model:

- **the** cat sat on the mat $P(x_1)$
- the **cat** sat on the mat $P(x_2|x_1)$
- the cat **sat** on the mat $P(x_3|x_2, x_1)$
- the cat sat **on** the mat $P(x_4|x_3, x_2, x_1)$
- the cat sat on **the** mat $P(x_5|x_4, x_3, x_2, x_1)$
- the cat sat on the **mat** $P(x_6|x_5, x_4, x_3, x_2, x_1)$

The log likelihood decomposes nicely into a sum.

$$\log P(x_1, \ldots, x_m) = \sum_{i=1}^{n} \log P(x_i | x_{i-1}, \ldots, x_i).$$

Learning is often done by maximum likelihood: select h that maximizes

$$\log P(x_1, \ldots, x_m | h) = \sum_{i=1}^{n} \log P(x_i | x_{i-1}, \ldots, x_i | h)$$

where h comes from a class of functions (e.g. recurrent neural networks) representing conditional distribution.

Relationship with Data Compression/Information Theory

- As described earlier, maximizing the log likelihood $\sum_{i=1}^{m} \log P(x_i|h)$ of the data $x_1, \ldots, x_m$ is equivalent to minimizing the empirical risk using the log loss $-\sum_{i=1}^{m} \log P(x_i|h)$.
- For **lossless compression**, using $P(x_1, \ldots, x_m)$ to compress $x_1, \ldots, x_m$ using Huffman coding or arithmetic coding, gives the shortest code length of $-\sum_{i=1}^{m} \log P(x_i|h)$ (ignoring rounding required for discrete lengths).
  - Maximizing $P(x_i|h)$ minimizes number of bits required.

Other Optimization Problems

- Density estimation by maximizing likelihood is one common formulation for unsupervised learning.
- Also, often formulated as (approximately) optimizing other objectives.
- For example, K-means is a clustering algorithm for partitioning a set of points in $\mathbb{R}^d$ into k clusters, minimizing

$$\mathcal{L}(\mathbf{C}, \boldsymbol{\mu}) = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

where $\boldsymbol{\mu}_i \in \mathbb{R}^d$ is a vector of length d acting as a cluster center and $\mathbf{C} = C_1, \ldots, C_k$ partitions the dataset with C_i being a subset of points whose closest cluster center is $\boldsymbol{\mu}_i$.

- Generative adversarial networks (GAN) is a popular recent model that solves the following two player minimax game:

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))],$$

  - G is a function that generates samples: takes in a random number z and outputs data of the target distribution
  - the first component of the objective tries to maximizes the likelihood of the observed data using D, and
  - the second component uses D to minimize the likelihood of data generated by G.

GAN generated images from [4].

Summary

1. Formulate as loss minimization
2. loss min $\xleftrightarrow[\text{equiv}]{\text{often}}$ Max likelihood

Reduce overfitting

Reg loss minimization $\xleftrightarrow[\text{equiv}]{\text{often}}$ MAP estimation

Outline

1. Loss Function
2. Maximum Likelihood
3. Model Selection
4. Features

Model Selection

- Most algorithms have parameters to tune in order to get good generalization performance.
- Many algorithms optimize a regularized loss function consisting of the empirical risk plus a regularization term that penalizes for complex functions

Example : Ridge regression

$$\frac{1}{m}\sum_{i=1}^{m}(y_i - w^T x_i)^2 + \lambda \|w\|^2$$

$$L_S(\mathbf{w}) + \lambda R(\mathbf{w})$$

for some function R, e.g. when doing MAP estimation.

  - One parameter to tune is the value of λ.

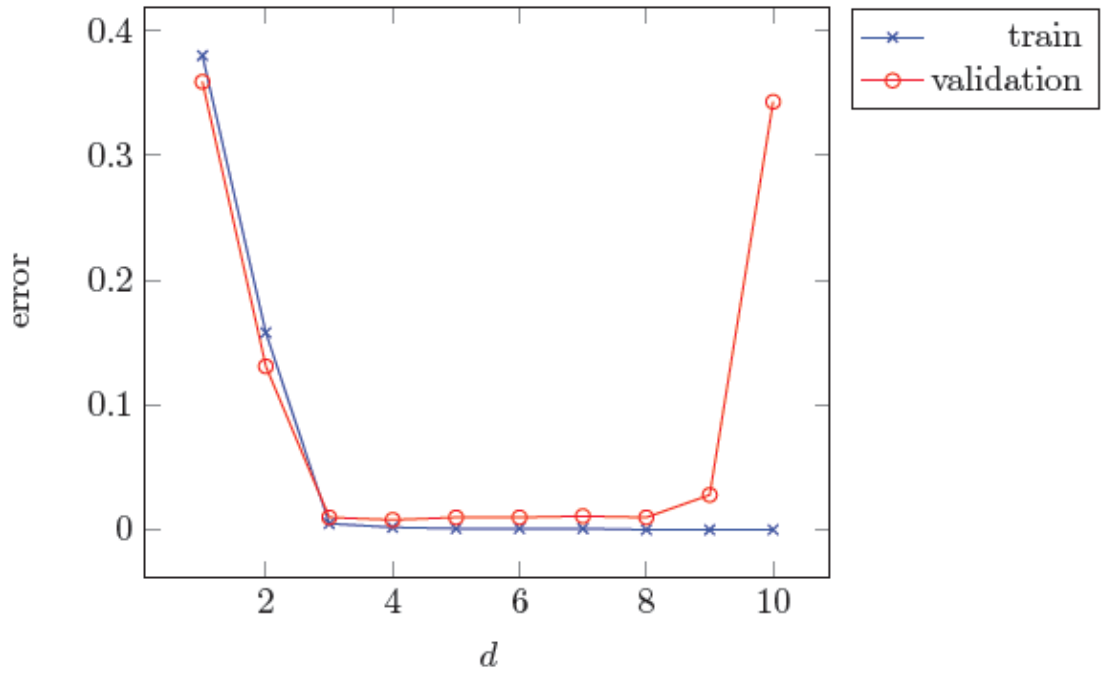


Figure from SSBD. Using validation set to select degree of polynomial in polynomial regression.

- In practice, it is common to leave some (randomly selected) data out of the training set to use to select the model.
- This is called the **validation set**. Also called development set

- The validation set is used to evaluate and select the different models.
  - For continuous parameters, a few discrete values may be selected for evaluation.
  - Grid search is sometimes done, with a coarse grid first, then zooming into the correct region to do a finer grid.
- The validation error is a reasonable estimate for the true error if
  - the validation set is large enough
  - the number of models being evaluated is not too large
  - the training data is representative of the test distribution
- After the model is selected, the validation set is often combined back into the training set and use to retrain the selected model with all the data.
- In solving a problem, typically we split the data into three parts: a training set (e.g. 60%), a validation set (e.g. 20%), and a test set (e.g. 20%).
- The test set is used to do the final evaluation of performance.

- If there is not enough data to train a good model after setting aside a validation set, k-**fold cross validation** is often used.
  - The training data is partitioned into k sets of size m/k.
  - The following is done for each partition
    - Use the partition to evaluate while the rest of the data for training
  - The validation errors are averaged and used to select the best model.
- In the extreme case, only one example is left out each time, with the rest of the training set used for training.
  - This is called **leave-one-out cross validation**.

What to do when learning fails

Do error analysis, What do cases with error have in common

- If training error is large.
  - Possibly underfitting. Look to reduce approximation error. Look for more useful features, use more powerful approximators.
  - Check that your are not regularizing too much.
  - It is also possible that the approximating function class is powerful enough, but the optimization algorithm is failing. Usually less likely with convex problems. For non-convex problems
    - Tuning your optimization parameters may help, e.g. learning rate in neural networks
    - Better initialization may help, particularly if you can use prior information to get a good starting point.

use validation not test set

- If validation error is high but training error is low
  - Possible overfitting. Look to reduce the estimation error. Do feature selection, or more regularization, etc. If possible, get more data.

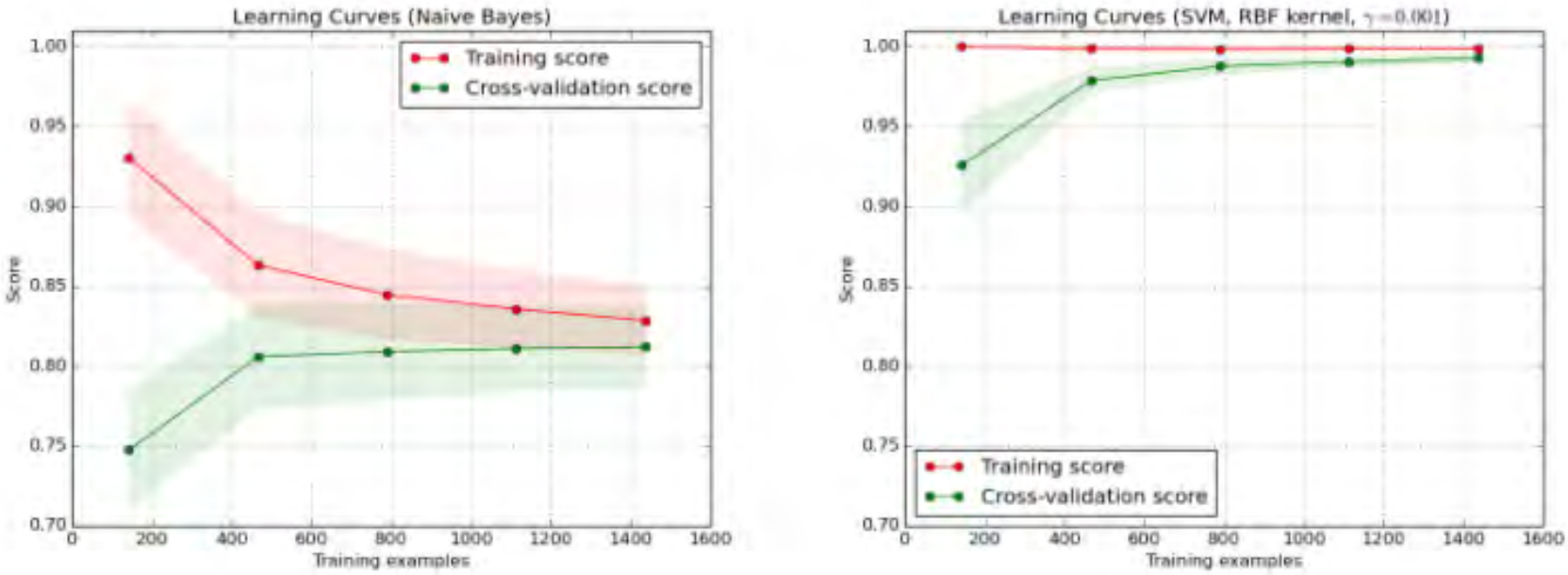


Figure from [5]

- The learning curve shows the validation and training score for varying number of training samples.
  - Plotting the learning curve can be helpful in telling us whether getting more data will likely be helpful for an algorithm.

Outline

1. Loss Function
2. Maximum Likelihood
3. Model Selection
4. Features

How to Win on Kaggle [2]

For most Kaggle competitions the most important part is feature engineering, which is pretty easy to learn how to do. (Tim Salimans)

The features you use influence more than everything else the result. No algorithm alone, to my knowledge, can supplement the information gain given by correct feature engineering. (Luca Massaron)

- Feature engineering involves using domain knowledge to define features that may be useful. For example, in text/NLP applications, may use
  - unigrams, bigrams, trigrams, constructed from the word sequences.
  - parts of speech, perhaps components of the parse trees.
  - specially selected words from dictionary.
  - other NLP components such as named entity detector, sentiment analysis, etc.

Reduce overfitting

- Then can do feature selection.

e.g. word embedding

- May also do feature generation by unsupervised learning.
- In certain domains, e.g. vision applications, learning the features e.g. using neural networks from raw data using supervised learning may work better.

Feature Selection and Generation

- We assume that instance space $\mathcal{X}$ that we are given is a subset of $\mathbb{R}^n$.
- This may be the raw input that we get, e.g. pixels in an image. Or it may be the output of a set of variables or features of the problem domain.
- For simplicity (and to be consistent with the term feature selection), we call each input component a feature.
- We may want to select only a subset of features
  - Sparse solutions may be more interpretable, e.g. we want to know which genes are likely responsible for a disease.
  - Using fewer features would speed up prediction.
  - Feature selection can improve generalization for some learning algorithms.
- We may also want to transform the features by weighting, normalization, and other transformations.
- We may want to construct other features out of the current features.

Filter

- Assess each feature independently of other features.
- Select k features that achieves the highest score.
- For example, Pearson correlation coefficient is defined as:

$$\frac{cov(X_j, Y)}{\sqrt{var(X_j)var(Y)}},$$

where *cov* is the covariance and *var* is the variance, and we want to maximize the correlation coefficient.

  - Ranks the features according to how well the single variable X_j can minimize the mean square error when used in a linear predictor:

$$\min_{a,b\in\mathbb{R}} \frac{1}{m}\sum_{i=1}^{m}(ax_{ij} + b - y_i)^2.$$

- For classification, *mutual information* is often used:

$$
\begin{aligned}
I(X_i; Y) &= \sum_{x_i} \sum_{y} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)} \\
&= H(Y) - H(Y|X).
\end{aligned}
$$

- $H(Y) - H(Y|X)$ is the difference between the entropy of the label Y and the conditional entropy of Y given input X, and is also called the *information gain*.
- For feature selection, we rank features according to information gain, and select the k features with the largest information gain.
- Another commonly used feature selection method is the χ^2 (chi square) feature selection method which tests whether the feature is independent of the label.

Wrapper

- Filter approaches do not take into account dependence among the features.
- Wrapper approaches search for subset of variables that will perform well. Often, the learning method is treated as a black box that is optimized by searching for a subset of variables.
- Common greedy methods include:
  - *Forward selection:* Features are progressively added as learning is done.
  - *Backward elimination:* In backward elimination, we start with all variables and progressively eliminate the least promising ones.
- Combinations of forward selecton and backward elimination and other search methods are also used in practice.

Sparsity-Inducing Norms

- We can formulate the feature selection problem as:

$$\min_{\mathbf{w}} L_S(\mathbf{w}) \text{ s.t. } ||\mathbf{w}||_0 \leq k,$$

Optimization problem

where $||\mathbf{w}||_0 = |\{i : w_i \neq 0\}|$.

  - $||\mathbf{w}||_0$ is often called the ℓ_0 norm, even though mathematically, it is not a norm.

- This is often computationally hard, so we often relax the ℓ_0 norm into a ℓ_1 norm to get

$$||w||_1 = \sum_{i=1}^{d} |w_i|$$

$$\min_{\mathbf{w}} L_S(\mathbf{w}) \text{ s.t. } ||\mathbf{w}||_1 \leq k,$$

  - If L_S is convex, then this is a convex optimization problem and can be solved efficiently.

- A related and more common way to encourage a sparse solution is to use the ℓ_1 norm of the weight as a regularizer,

$$\min_{\mathbf{w}}(L_S(\mathbf{w}) + \lambda||\mathbf{w}||_1).$$

  - Again, this is convex if L_S is convex, e.g. in logistic regression.
- Adding ℓ_1 regularization to linear regression with the squared loss gives us the LASSO algorithm,

$$\min_{\mathbf{w}}(\frac{1}{2m}||X\mathbf{w} - \mathbf{y}||^2 + \lambda||\mathbf{w}||_1).$$

- Can derive Lasso as MAP with each w_i independently distributed with Laplacian prior $\Pr(w_i) = \frac{\lambda}{2}\exp(-\lambda|w_i|)$.

Closely related problem:

Find params in shaded area that minimize loss

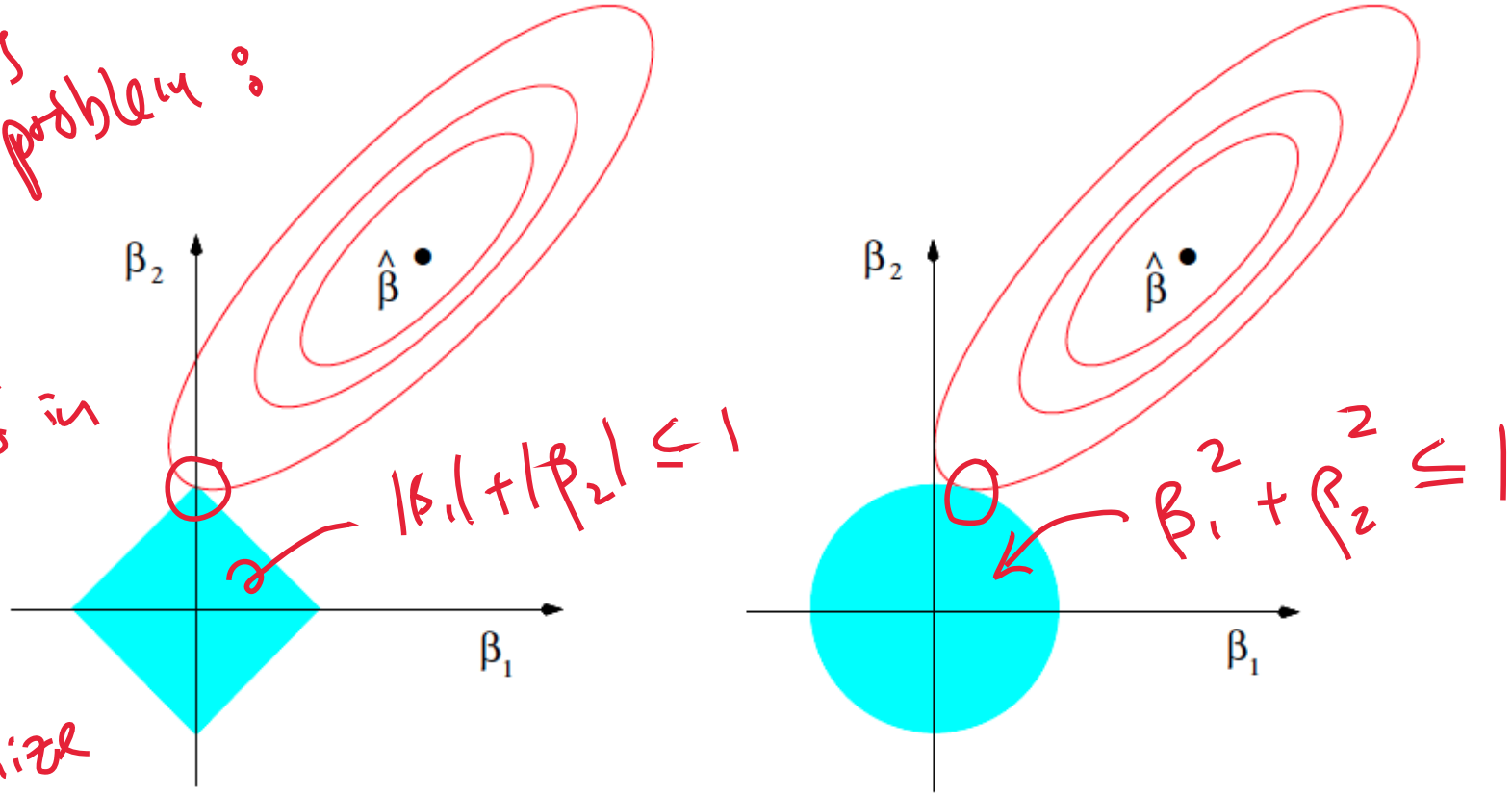


Figure from [3]. Lasso vs ridge regression.

Unlike the ℓ_1 norm, the ℓ_2 norm does not induce sparse solutions.

Exercise 3:

Regularizing using the ℓ_1 norm rather than the ℓ_2 norm induces sparsity and can serve as a method for feature selection. This is often called Lasso. The optimization objective for Lasso is $\frac{1}{2m}||y - Xw||_2^2 + \alpha||w||_1$. We will compare the two methods using polynomial regression with a 10th degree polynomial using noisy training data from a linear function to see if we get a sparse solution.

Comment on the outcome of the experiment.

- Lasso zeroed out most coeffs
- Ridge regression does not have sparse solution

Feature Transformation and Normalization

Example: Ridge regression. Assume x_1, x_2 measured in meters
True function is $w_1 x_1 + w_2 x_2$, $w_1 = w_2 = 1$
If instead, receive x_2 in kilometers
true function have $w_1 = 1$, $w_2 = 1000$

- Some simple transformations on the features can often affect performance.
- For example, if the scales of the features are very different, a feature that has very small range may need very large weights to have an effect.
  - With regularization, this feature may be overpenalized as it would cost a lot to use a large weight.
  - In this case, normalizing the features may be helpful.

poor generalization

Also cause poor optimization performance, ill-conditioning

Let $\mathbf{f} = (f_1, \ldots, f_m) \in \mathbb{R}^m$ be the value of feature f over the m training examples and $\bar{f} = \frac{1}{m}\sum_{i=1}^{m} f_i$ be the empirical mean of the feature.

Some common transformations:

- **Centering:** Transform the feature to have zero mean: $f_i \leftarrow f_i - \bar{f}$.
- **Unit Range:** Set the range of the feature to $[0, 1]$. Let $f_{\max} = \max_i f_i$ and $f_{\min} = \min_i f_i$. Then the transformation is: $f_i \leftarrow \frac{f_i - f_{min}}{f_{\max} - f_{\min}}$.
- **Standardization:** Transform the feature to have zero mean and unit variance. Let $\nu = \frac{1}{m}\sum_{i=1}^{m}(f_i - \bar{f})^2$. Then the transformation is: $f_i \leftarrow \frac{f_i - \bar{f}}{\sqrt{\nu}}$.

variance

std dev

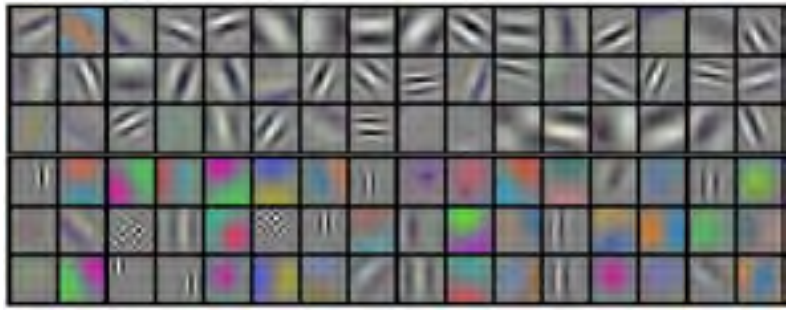
- **Clipping:** Clips the high low values of a feature: $f_i \to \mathrm{sign}(f_i)\max(b, |f_i|)$ where b is a user specified parameter.
- **Sigmoidal transformation:** Apply sigmoid function to the feature: $f_i \leftarrow \frac{1}{1+\exp(bf_i)}$ where b is a user specified parameter.
- **Logarithmic Transformation:** Used when the difference for small values, e.g. between 0 and 1, is more important than the difference for large values, e.g. 1000 and 1001: $f_i \leftarrow \log(b + f_i)$, where b is a user specified parameter.

Transformation may be domain specific. For example, in text processing, a word that appears many times in a document may be important to the document, but if the word appears in all documents, it is unlikely to be useful in telling documents apart. This motivates the *term frequency-inverse document frequency* (TF-IDF) representation.

- Let $f(w, d)$ denote the number of times word w appears in document d.
- Let $g(w)$ denote the number of documents where w appears in within the document collection with m documents.
- Then $\text{tf-idf}(w) = f(w, d) \log \frac{m}{g(w)}$.

Feature Learning

- Instead of selecting a subset of predefined features, we can also learn a feature mapping $\psi : \mathcal{X} \mapsto \mathbb{R}^d$ which maps instances in $\mathcal{X}$ to d dimensional feature space.
- Such learned features can be more informative that the original features for some domains, e.g. pixels in images are not very informative but a feature that can tell if a set of pixels comes from a part of a face would be more useful.
- One way would be to learn the features directly as part of supervised learning, e.g. by using hidden units in neural networks.



Features learned by AlexNet.

Image source: `http://cs231n.github.io/convolutional-networks/`.

- But supervised learning requires labeled data, which can often be expensive.
- If we have a lot of unlabeled data, may be useful to do unsupervised feature learning, sometimes called *dictionary learning*.

Dimension Reduction

- One approach for dictionary learning is to construct an *autoencoder*.
  - An autoencoder constructs a function pair: an encoder $\psi : \mathbb{R}^d \mapsto \mathbb{R}^k$ and a decoder $\phi : \mathbb{R}^k \mapsto \mathbb{R}^d$.

    $d \gg k$ $\psi(x) = v$ encoder (compression)

    $\phi(v) = \hat{x}$ decoder (reconstruct)

  - The goal of learning is to minimize the reconstruction error: $\sum_i ||\mathbf{x}_i - \phi(\psi(\mathbf{x}_i))||^2$ is often used as the reconstruction error.

- Continued ...
  - Principal Component Analysis (PCA) is an example where $k < d$. In PCA, we map the input into a smaller number of dimensions (dimensionality reduction) in a where that the reconstruction error using the smaller number of dimensions is minimized.
  - In PCA, $\psi(\mathbf{x}) = V_k^T \mathbf{x}$ is a linear transformation. The decoder $\phi(\mathbf{u}) = V\mathbf{u}$ is also a linear transformation.
  - In general, k need not be less than d. It is also possible to use non-linear transformations such as deep neural networks for ψ and ϕ.

Reading

Some of the presented material are taken from:
Understanding Machine Learning: From Theory to Algorithms
Shai Shalev-Shwartz and Shai Ben-David
Cambridge University Press.

Online copy `http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/copy.html`

1. Section 9.3 on logistic regression
2. Chapter 24.1 on maximum likelihood
3. Chapter 11 on model selection
4. Chapter 25 on feature selection and generation.

References I

[1] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] *David Wind, Learning from the Best*. [Online: http://blog.kaggle.com/2014/08/01/learning-from-the-best/, accessed July 2017].

[3] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York, 2001.

[4] Tero Karras et al. "Progressive growing of gans for improved quality, stability, and variation". In: *arXiv preprint arXiv:1710.10196* (2017).

References II

[5] *Learning curve*. [Online: http://scikit-learn.org/0.15/auto_examples/plot_learning_curve.html, accessed July 2017].

[6] *Logistic Regression*. [Online: https://en.wikipedia.org/wiki/Logistic_regression, accessed July 2017].