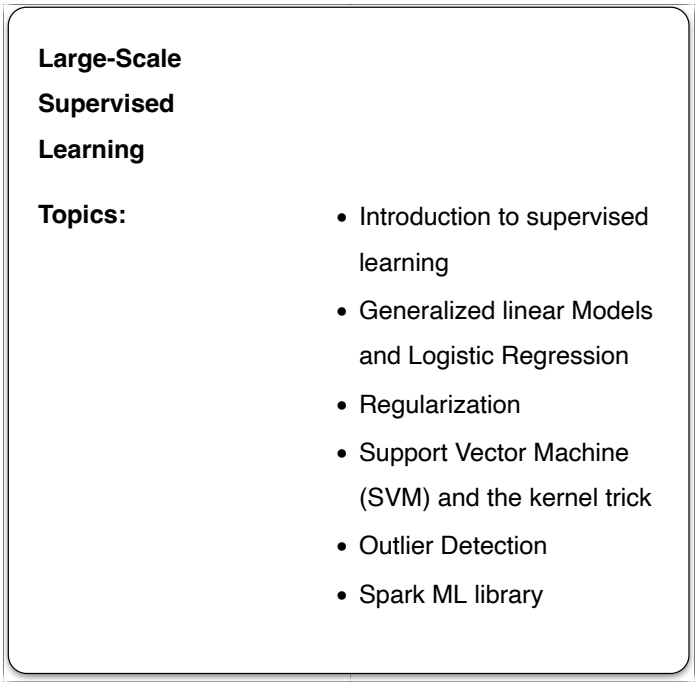
CS777 Module 4: Large-Scale Supervised Learning



4.1. Learning Objectives

After successfully completing the module, students will be able to do the following:

- Use logistic regression on a very large data set.
- Apply classification data model Support Vector Machine (SVM).
- Examine model learning problems like overfitting and underfitting and decide how to avoid the issues.

4.2. Module Overview

In this module, we learn about supervised learning. We will start from Generalized Linear Models and learn details of logistic regression and how we can use it on a very large data set. In addition, we learn also about Support Vector Machine (SVM) which is another important classification data model that we will learn it in detail. In addition, we learn model learning problems like overfitting and underfitting, and we will see solutions to avoid these issues.

4.3. Supervised Learning

In supervised learning we are working on a pair of data points and labels. For example, learning a data model to discriminate different plant types, like Roses and Cactus. Consider that our data points are sample from the population $\mathfrak{X}$ with a probability distribution of $P(\mathfrak{X})$ which is not known. We can formally represent these samples as:

$$X = x_1, x_2, \dots, x_n \qquad (4.1)$$

Note that $X \subset \mathfrak{X}$

For example Roses and Cactus plants can be characterized by their features or attributes, like, $\{Shape, Size, Color\}$, so we can work on 3 features or 3 dimensions.

Features can be discrete (a finite number of values) or continuous values, each feature $i \in [d]$ might be a scalar in $\mathbb{R}$ so that we have space of $\mathbb{R}^d$.

Each X_i data sample will have d dimensions.

$$X_i = x_{i_1}, x_{i_2}, \dots, x_{i_d} \qquad (4.2)$$

The input data and corresponding labels can be written in matrix form as:

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1d} \\ x_{11} & x_{12} & \dots & x_{1d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{11} & x_{12} & \dots & x_{1d} \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{bmatrix} \qquad (4.3)$$

We can also write the above matrix in tuple form as:

$$Data_{withLabels} = \{(X_1, y_1), (X_2, y_2), \dots, (X_n, y_n)\}$$

4.4. Generalized Linear Models (GLM)

To understand the generalized linear model, let us start with the ***Simple Linear Regression***.

$$Y = \beta_1 X + \beta_0 + \epsilon$$

- β_1 is the coefficient of the independent variable (slope).
- β_0 is the constant term or the y intercept.
- ϵ is the error – the distance between actual value and model value.
- Our goal is to minimize errors $\epsilon_i = \hat{y}_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i)$.

4.4.1. The Method of Least Squares

Given a set of observation data $\langle (x_1, y_1), (x_2, y_2), \dots, (x_n, y_n) \rangle$

- We find the optimal values of β_1 and β_0 by minimizing the cost function of

$$E(\beta_0, \beta_1) = \sum_{i=1}^{n} (y_i - (\beta_1 x_i + \beta_0)^2)$$

- Least squares linear regression assumes a stochastic process using normal error with variance 1.

Assuming that our observed values are $\langle x_1, x_2, x_3 \rangle$ and we can generate predictions values $\langle y_1, y_2, y_3 \rangle$ by using a normal distribution with variance 1.

The Probability Density Function (PDF) for my errors $\langle e_1, e_2, e_3 \rangle$ is:

$$PDF = \prod_{i=1}^{3} Normal(x_i | y_i, 1)$$

Log-likelihood is then:

$$\begin{aligned} LLH &= \sum_{i=1}^{3} Normal(x_i | y_i, 1) \\ &= \sum_{i=1}^{3} -\frac{1}{2}(x_i - y_i)^2 - \frac{1}{2}\log(2\pi) \\ &\propto \sum_{i=1}^{3} (x_i - y_i)^2 \end{aligned}$$

We have a set of training data like:

- Data Matrix of $X \in \mathbb{R}^{n \times d}$, with Labels $Y \in \mathbb{R}^{n \times 1}$
- Linear Regression Model:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d + e$$

  The same in matrix form: $Y_{n \times 1} = X_{n \times d} \Theta_{d \times 1} + e_{n \times 1}$

- We are looking for a vector of parameters (set of weights) $\Theta \in \mathbb{R}^{d \times 1}$ to minimize

$$MSE = L(\Theta) = \frac{1}{N} e^T e = \frac{1}{N} (y^T y - 2\Theta^T X^T y + \Theta^T X^T X \Theta)$$

Finding exact the solution by minimizing the cost function:

- This function is a convex function.
- Take the derivative and set it to zero $\frac{d}{d\Theta} L(\Theta) = -2X^T(Y - X\Theta) = 0$

We set the above to zero and will get:

$$\Theta = (X^T X)^{-1} X^T Y$$

4.5. Distributions in Exponential Family

- We can use **ANY stochastic process** for generating the error, not just normal distribution.
- GLM is an extension of linear regression that allows errors to be generated by a wide **variety of distributions**.
- In particular, any distributions in the “Exponential Family”
- What are distributions in “Exponential Family”?

Any kind of distributions is in ***Exponential Family*** if it can be written as:

$$p(y|\theta) = b(y)\exp(\eta(\theta)T(y) - f(\theta))$$

In Example of Bernoulli Distribution:

$$\begin{aligned} p(y|p) &= p^y * (1-p)^{(1-y)} \\ &= \exp(y\log(p) + (1-y)\log(1-p)) \\ &= \exp((\log(p) - \log(1-p))y + \log(1-p)) \end{aligned}$$

So we have:

$\eta(\theta)$ is $(\log(p) - \log(1-p))$ or $\log(p/(1-p))$

$f(\theta) = -\log(1-p) = \log(1+e^{\theta})$

$T(y)$ is y and $b(y)$ is 1.

Distributions in Exponential Family

$$p(y|\theta) = b(y)\exp(\eta(\theta)T(y) - f(\theta))$$

For Example – normal with variance of 1:

$$\begin{aligned} p(y|p) &= \frac{1}{\sqrt{(2\pi)}}\exp(-\frac{1}{2}(y-\mu)^2) \\ &= \frac{1}{\sqrt{(2\pi)}}\exp(-\frac{1}{2}y^2 + y*\mu - \frac{1}{2}\mu^2) \\ &= \frac{1}{\sqrt{(2\pi)}}\exp(-\frac{1}{2}y^2)\exp(\mu*y - \frac{1}{2}\mu^2) \end{aligned}$$

So we have:

$\eta(\theta)$ is μ

$T(y)$ is y

$b(y)$ is $\frac{1}{\sqrt{(2\pi)}}\exp(-1/2y^2)$

$f(\theta)$ is $\frac{1}{2}\mu^2$

Using GLM in Prediction Problems

Suppose that we have a prediction problem, where:

a. It is natural to assume that output y given the independent variable(s) X and model parameter θ is sampled from the exponential family.

b. The parameter θ is linearly related to X that is, assuming X is vector-valued

$$\theta = \sum^{j} x_j \beta_j$$

Where β_j is regression coefficients.

This is then an instance of a "generalized linear model" by definition of the exponential family, the log-likelihood can always be written as:

$$LLH = \sum^{i} \log b(y_i) + \theta_i T(y) - f(\theta_i)$$

where

$$\theta_i = \sum^{j} x_{ij} \beta_j$$

4.6. Example of Using GLM - Logistic Regression

- Suppose we have a classification problem that has yes/no (0/1) dependent variables (output Y)
- It makes sense to assume that the output Y is sampled from a **Bernoulli**.

$$\sum^{i} \log 1 + y_i \times \theta_i - \log(1 + e^{\theta_i})$$

Choosing $\beta_1, \beta_2, \ldots$ to maximize the above equation, is known as ***logistic regression***.

- Logistic regression for classification is one another example of a GLM!

Once we learn the model, when we want to make a prediction given an x, we choose y so as to **maximize the value of**:

$$y \times \theta - \log(1 + e^{\theta})$$

Example

- if θ is -4, then choosing $y = 1$ gives -4.008, but choosing $y = 0$ gives -0.008. So, we choose 0.
- if θ is 3, then choosing $y = 1$ gives 1.68, but choosing $y = 0$ gives -1.33. So we choose 1.

In fact with logistic regression, our prediction is:

$$\textbf{0 if } \theta < 0\textbf{, and 1 if } \theta > 0.$$

In the literature, you can find other forms of loss function for the logistic regression that depends on the data label assumptions and derivation method but will result in the same data model. Check out: Why there are two different logistic loss formulation/notations?

Example 1

Suppose we have a real-valued output. We assume that the output is sampled from a Normal. From the Normal example above, we have the following LLH:

$$\sum_i l - \frac{1}{2}y_i^2 + \theta_i y_i - \frac{1}{2}\mu^2.$$

Maximizing this just gives us least squares regression! So least squares regression is another case of a GLM.

4.6.1. Simple Linear Regression vs. GLM

- **Simple Linear Regression:** normal Errors, constant Variance
- **Generalized Linear Model:** non-normal Errors, non-constant Variance

4.6.2. Generalized Linear Model (GLM)

Any distribution is from “exponential family”.

Depending upon the particular use case, choose the model:

- Linear Regression – Normal Distribution
- Logistic Regression – Logistic Distribution
- Poisson Regression – Poisson Distribution
- Binomial Regression – Binomial Distribution
- Gamma Regression – Gamma Distribution
- Softmax Regression – Softmax Function

General vs. Generalized Linear Model

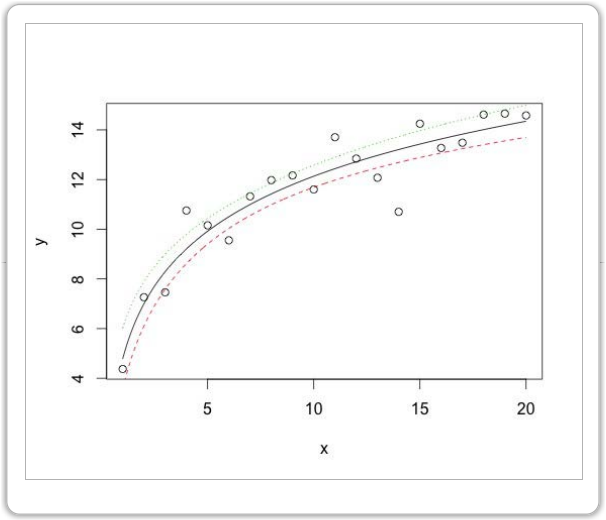
Note the naming confusion **“General” vs. “Generalized”**

- **Generalized** Linear Model. Examples: linear regression, logistic regression, Poisson regression, gamma regression, …
- **General** Linear Model. Examples: Analysis of Variance (ANOVA), Analysis of covariance

(ANCOVA), Multivariate analysis of variance (MANOVA), MANCOVA, …

4.6.3. Examples of Nonlinear Regressions

- Logarithmic Regression
  $y = a + b(ln(x))$
- Quadratic Regression
  $y = ax^2 + bx + c$
- Power Regression
  $y = ax^b$
- Exponential Regression
  $y = a^{bx}$



4.7. Performing Logistic Regression - Implementation Details

Let us implement the logistic regression from scratch:

For logistic regression, we have:

$$LLH(r_1, r_2, \dots, r_d | x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n) = \sum_i y_i \theta_i - \log(1 + e^{\theta_i})$$

where

- $\theta_i = r_j \times x_{i,j}$
- r is the regression coefficient.

Example 2

Suppose we have a bunch of (e1_score, e2_score) data pairs

(92, 12), (23, 67), (67, 92), (98, 78), (18, 45), (6, 100)

Final result in class: $\langle fail, fail, pass, pass, fail, fail \rangle$

- If coefs are (-1, -1), LLH is -698

- If coefs are (1, -1), LLH is -133
- If coefs are (-1, 1), LLH is 7.4
- If coefs are (1, 1), LLH is 394 (the maximum LLH that we get here)

4.7.1. Loss Function

We will learn this model using gradient descent.

Remember, GD tries to MINIMIZE, so our loss function is the NEGATIVE LLH

$$\sum_i -y_i\theta_i + \log(1 + e^{\theta_i})$$

4.7.2. Regularization

But regularize! Why?

- In previous example, if coefs are (1, 1), LLH is 394.
- If coefs are (3, 3), LLH is 1184.
- If coefs are (4, 4), LLH is 1579.

So our loss regularized loss function is

$$\sum_i -y_i\theta_i + \log(1 + e^{\theta_i}) + \lambda \sum_j r_j^2$$

With $\lambda = 10$, minimized at $r_1 = r_2 = 0.8$

4.7.3. Deriving Gradient Descent

Just have to take partial derivative with each r_k, use this in the gradient calculation.

Just to refresh your memory:

If we restrict f to consider just one data point x_i:

$$\frac{\partial f}{\partial r'_j} = -x_{i,j'}y_i + x_{i,j'}(\frac{e^{\theta_i}}{1+e^{\theta_i}}) + 2\lambda r'_j$$

So for the whole data set we have:

$$\frac{\partial f}{\partial r'_j} = (\sum_i -x_{i,j'}y_i + x_{i,j'}(\frac{e^{\theta_i}}{1+e^{\theta_i}})) + 2\lambda r'_j$$

4.7.4. Note on Implementation

Depending upon learning rate, it will need a lot of iterations to converge. So core loop MUST be fast.

No joins, sorts, and anything other than a MapReduce.

Pseudo-code:

```
cur_answer = # some initialization while no convergence:

gradient = big_ass_list_of_vectors.map(

# use cur_answer to get grad for current point

    ).reduce(

        # add up all of the one-point gradients
    )

    gradient = gradient + regularization penalty

    curanswer -= learning_rate * gradient
```

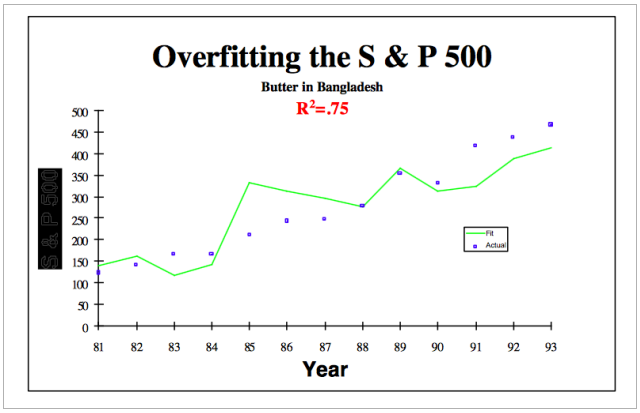
4.8. Over-Fitting and Regularization

One fundamental problem in data science is if we have enough hypotheses to check, then one of them is bound to be true.

4.8.1. Predicting the S&P 500

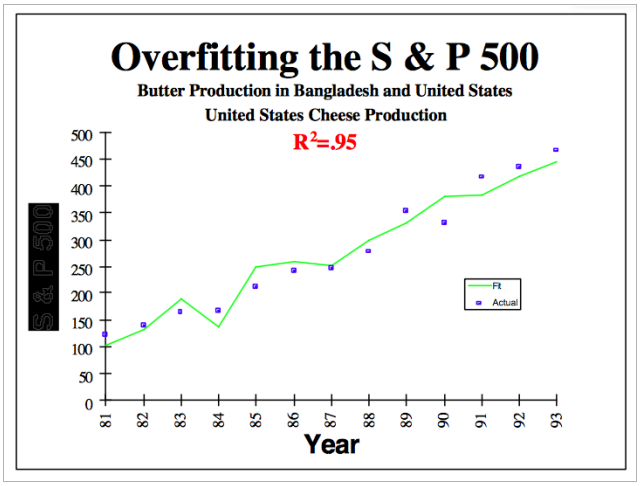
In the following you can see a set of overfitting graphs that builds a prediction model based on correlation between Butter production in Bangladesh and S&P 500 Stock market index.

Figure 4.1: Butter Production in Bangladesh and S&P 500 Rates with $R^2 = .75$



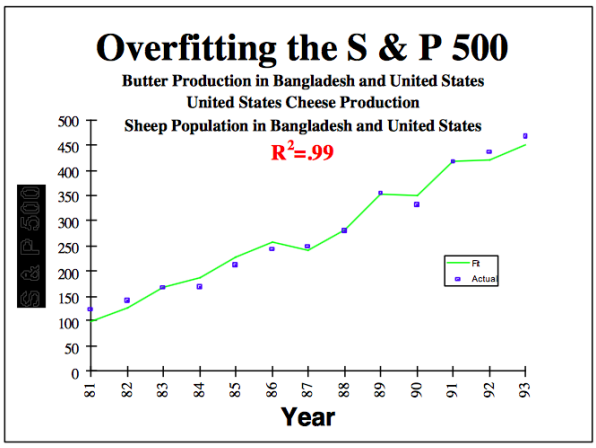
Source: Leinweber, D. J. (2017). Stupid Data Miner Tricks: Overfitting the S&P 500. The Journal of Investing.

Figure 4.2: Butter Production in Bangladesh and S&P 500 Rates with $R^2 = .95$



Source: Leinweber, D. J. (2017). Stupid Data Miner Tricks: Overfitting the S&P 500. The Journal of Investing.

Figure 4.3: Butter Production in Bangladesh and S&P 500 Rates with $R^2 = .99$



Source: Leinweber, D. J. (2017). Stupid Data Miner Tricks: Overfitting the S&P 500. The Journal of Investing.

None of the above Models is likely to generalize well. This means:

- They have learned the input data
- Not any underlying truth
- When deployed in the field, likely to fail

The Term "Data Mining"

Data Mining was originally a derogatory term used by stats, meant that you could always find something if you look hard enough.

4.8.2. Detecting Over-Fitting

1. **Detection method 1**: Sniff Test

   For example, do some visualization, scatterplot, residual plots, etc.

   For example, check if the learned regression coefficients would make sense—e.g., there is no very small number close to zero.

2. **Detection method 2:** Independent Validation and Test Sets
   - Avoid temptation!

4.8.3. The Bias-Variance Trade-Off

In a nutshell, we have two main sources of error in learning:

- **"Bias:"** error from incorrect model assumptions
- **"Variance:"** sensitivity of model to bad data

Bias

Difference between what our model predicts and the truth

Imperfect models:

- Simplifying assumptions to make the model easier to learn
- What are common simplifying assumptions? e.g., sometimes we assume that data is normally distributed

Variance

Variability of a model's prediction for a data point

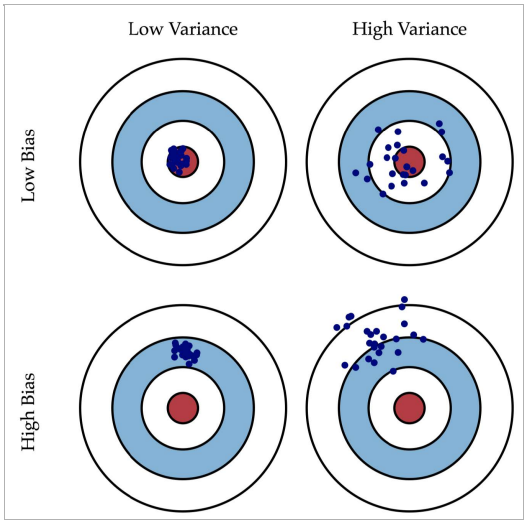
Imperfect models:

- If we build multiple models, how consistent are the predictions across these models?
- Sensitivity of the output to small fluctuations in the training set
- The model is highly dependent on our selection of training data.
- A good distribution of training data will enable us to predict new data well.
- Variance is sensible to the existence of outliers in training data.

Bias Variance Errors

- Error due to Bias
- Error due to Variance

Figure 4.4: Bias Variance Error



Source: Understanding the Bias-Variance Tradeoff

4.8.4. Understanding the Trade-Off

Expected squared error or any prediction is:

$$E[(Y - \hat{f}(X))^2]$$

Here:

- Y is the output we are trying to predict.
- $f(.)$ is the model we are learning (is a random variable!)
- X is the observed data (Example: set of regressors).
- Y is the value we are trying to predict from X.

4.8.5. Expected Value and Variance

Expected values can be used to compute the variance. We use the computational formula for the variance:

$$Var(Y) = E[Y^2] - E^2[Y]$$

We define that

$$Bias^2(\hat{f}(x)) = E^2[\hat{f}(x) - Y]$$

Understanding the Trade-Off

$Var(Y) = E[Y^2] - E^2[Y]$,

$Bias^2(\hat{f}(x)) = E^2[\hat{f}(x) - Y]$

Expected squared error or any prediction is:

$$\begin{aligned} E[(Y - \hat{f}(X))^2] &= E[Y^2 + \hat{f}^2(X) - 2Y\hat{f}(X)] \\ &= E[Y^2] + E[\hat{f}^2(X)] - E[2Y\hat{f}(X)] \\ &= Var(Y) + E^2[Y] + E[\hat{f}^2(X)] - E[2Y\hat{f}(X)] \\ &= Var(Y) + E^2[Y] + Var(\hat{f}(X)) + E^2[\hat{f}(X)] - E[2Y\hat{f}(X)] \\ &= Var(Y) + Var(\hat{f}(X)) + (E^2[Y] - E[2Y\hat{f}(X)] + E^2[\hat{f}(X)]) \\ &= Var(Y) + Var(\hat{f}(X)) + Bias^2(\hat{f}(X)) \end{aligned}$$

Means error is a sum of:

- “Looseness” of relationship between X and Y
- Sensitivity of the learner to variability of the training data (variance)
- Inability of the learner $\hat{f}$ to learn the relationship between X and Y (bias)

It is the second one (variance) that leads to over-fitting.

Ideally, Reduce Both Bias and Variance!

Unfortunately, it's not possible "**in real life**".

- We don't know the real model – bias is guaranteed.
- Since we ARE wrong, choose a general model and lots of features.
- Hence variance (over-fitting) is also guaranteed.
- Best we can do: choose sweet spot where error is minimized.

4.8.6. Regularization

Regularization is a massively important idea in data science.

- In a nutshell: Give learning algorithm ability to choose complexity of model.
- Automatically choose the correct trade-off between bias and variance.

Done by adding a penalty term to objective function: Penalizes model for complexity.

Typically, penalty is a norm computed over the model.

Recall, l_p norm of a vector $\langle x_1, x_2, \ldots \rangle$ computed as:

$$\sum_{i=1}^{n} |x_i|^p)^{1/p}$$

Common penalties are l_1, l_2.

Example 3: Logistic Regression

Standard objective function is:

$$\sum_i y_i \theta_i - \log(1 + e^{\theta_i})$$

where $\theta_i = r_j \times x_{i,j}$

Change objective function to:

$$\sum_i y_i \theta_i - \log(1 + e^{\theta_i}) + \lambda ||r||_p$$

Here $||r||_p$ is the l_p norm of the regression coefficients,

- if $p = 1$ have **"the lasso regression"**.
- if $p = 1$ have **"ridge regression"**.
- λ controls the magnitude of the penalty.
- typically, try different values of λ during validation.

Some Remarks

When regularizing, it is important to normalize data.

Bayesians (Bayesian statistician) argue they are protected from over-fitting.

- A good prior protects against complex models.
- In fact, “the lasso” is closely related to Bayesian Linear Regression (BLR) with Laplace prior on r.
- Ridge regression is closely related to BLR with Normal prior.

4.9. Support Vector Machines

Support vector machine or SVM for short is an alternative approach to Logistic Regression for classification.

The main “Big Three” methods for classification are:

- Logistic Regression
- Support Vector Machines (SVM)
- k-Nearest Neighbors (kNN)

4.9.1. Problem With Un-Regularized Logistic Regression

When the data set is “linearly separable”

- It is possible to choose model that gets infinite LLH
- Just choose ANY cutting plane between classes
- Then pump up the magnitude of regression coefs arbitrarily

Two important disadvantages:

- Bad: Not clear which plane is preferred.
- Bad: Model fails in easiest case!

Figure 4.5: Positive and Negative Data Elements

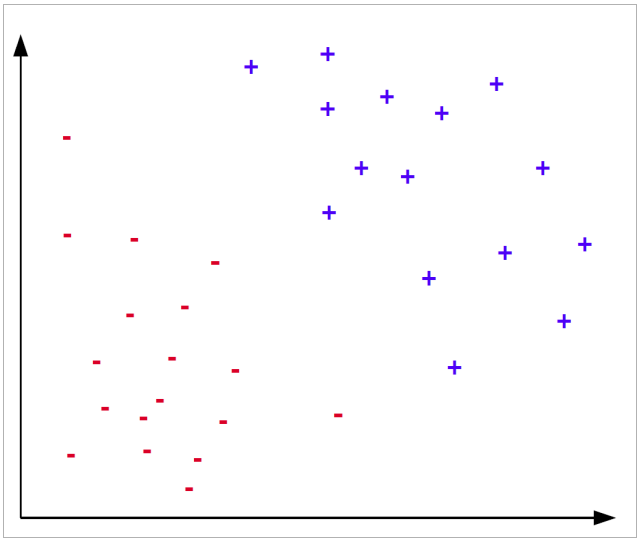
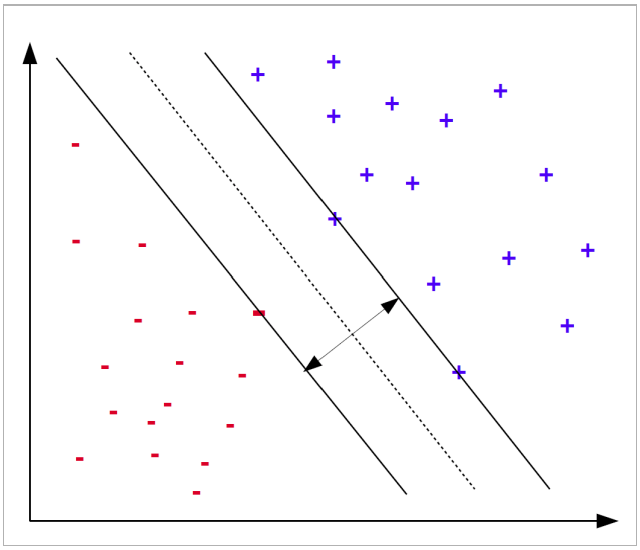


Figure 4.6: Support Vectors Separating Positive Data Elements from Negatives



4.9.2. SVMs: Geometric, Not Probabilistic

Starts with question:

- What should classifier do in this super-easy case?
- Just put the widest strip possible between two classes.
- Future points above center of strip are **"yes"**.
- Below center of strip are **"no"**.

- Points that keep strip from expanding are “support vectors”.

4.9.3. Basic Formulation

Any line/plane/hyperplane can be described by a normal vector w, distance b

- The line/plane/hyperplane is all points x where $\vec{w} \cdot \vec{x} + b = 0$

We need to consider them as vectors.

$$\vec{w} \cdot \vec{x} + b = 0 \qquad (4.4)$$
$$\forall \vec{x}_i : \vec{w} \cdot \vec{x}_i = -b \qquad (4.5)$$

Assume that we have an arbitrary point $\vec{x}$ and we want to calculate the distance between an arbitrary point to the hyperplane.

Distance is the dot product of two vectors $\vec{w} \cdot (\vec{x} - \vec{x}_0)$, assume that $\vec{x}_0$ is a point on the hyperplane and $\vec{x}$ is any point.

$$\begin{aligned}\gamma &= \vec{w} \cdot (\vec{x} - \vec{x}_0) \\ &= \vec{w} \cdot \vec{x} - \vec{w} \cdot \vec{x}_0 \\ &= \vec{w} \cdot \vec{x} + b\end{aligned}$$

We know from above that $\vec{w} \cdot \vec{x}_0 = -b$

Given a training set of $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, maximize the distance γ by varying the $\vec{w}$ and b subject to constraint that for all $i = 1, 2, \ldots n$

$$y_i(\vec{w} \cdot \vec{x} + b) \geq \gamma$$

Notice that y_i must be +1 or -1 determines which side of the hyperplane the point $\vec{x}$ is.

SVM chooses two planes:

$$\vec{w} \cdot \vec{x} + b = 1 \qquad (4.6)$$
$$\vec{w} \cdot \vec{x} + b = -1 \qquad (4.7)$$

- Where $\vec{w} \cdot \vec{x}_i + b \geq 1$ when $\vec{x}_i$ is **“yes”** (or a point of positive samples)
- Where $\vec{w} \cdot \vec{x}_i + b \leq -1$ when $\vec{x}_i$ is **“no”** (or a point of negative samples)

We can introduce a new variable y, so that y is 1 for positive samples and -1 for negative samples.

We multiply the above equations by y so that we get for both of them $y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1$.

- Distance between them is $\frac{2}{||w||}$.
- Use $||w||$ to denote the l_2 norm of the vector w. It is the Frobenius/Euclidean norm, or the square root of the sum of the squares of the components of w.

> So in the end we have…

Choose w to **maximize** the distance $\frac{1}{||w||}$ (Or alternative **minimize** $||w||$)

Or more mathematically convenient **minimize** $\frac{1}{2}||w||^2$

Subject to $y_i(w \cdot x_i + b) \geq 1$

This is our objective function.

Example 4: A 2D SVM Simple Example

Assume that we have the following four data points.

$([1, 2], +1), ([2, 1], -1)$
$([3, 4], +1), ([4, 3], -1)$

Let $w = [u, v]$

Our goal is to minimize $\sqrt{u^2 + v^2}$ subject to the constrains that we derive from the given four training examples.

- For the first one $x_1 = [1, 2]$ and $y_1 = +1$, then $(+1)(u + 2v + b) = u + 2v + b \geq 1$
- For the second point $x_2 = [2, 1]$ and $y_2 = -1$, then $(-1)(2u + 1v + b) = 2u + v + b \geq 1$ or $2u + v + b \leq -1$

So for all 4 data points we have then:

$u + 2v + b \geq 1$ and $2u + v + b \leq -1$
$3u + 4v + b \geq 1$ and $4u + 3v + b \leq -1$
$b = 0$ and $w = [u, v] = [-1, +1]$

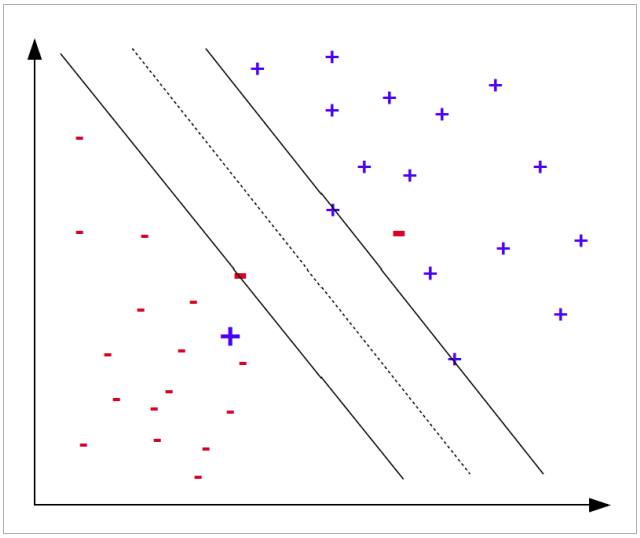
So this is an unusual easy-to-solve case.

4.9.4. One Issue: What If Not Linearly Separable?

Then no solution to above problem!

- Solution: don't require a “hard margin”
- Allow some error
- Training points on wrong side of cutting plane gets a penalty

Figure 4.7: Support Vectors Separating Positive Data Elements from Negatives



4.9.5. "Soft Margin" Formulation

Choose w **to minimize** $||w||^2 + c \sum_i \epsilon_i$

Subject to $y_i(w \cdot x_i + b) \geq 1 - \epsilon_i$

ϵ is error. We allow to have some errors.

c is some constant.

The above is called ***soft-margin SVM***.

4.9.6. How To Solve?

This is a constrained optimization problem.

Everything we've done (GD, Newton's) has been un-constrained.

It turns out, we can re-write it as unconstrained.

We need to simply minimize the following function:

$$\frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_i max(0, 1 - y_i(w.x_i)) \qquad (4.8)$$

where $\lambda = \frac{1}{n \times c}$

Constant c is the regularization parameter.

Amenable to gradient descent (one issue: non-smooth max function)

This quadratic optimization problem is known as the ***primal problem***.

4.9.7. Implementing SVM

Our objective function is the following function, and we need to minimize this Loss with Gradient Descent.

$$\frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_i max(0, 1 - y_i(w.x_i)) \qquad (4.9)$$

where $\lambda = \frac{1}{n \times c}$

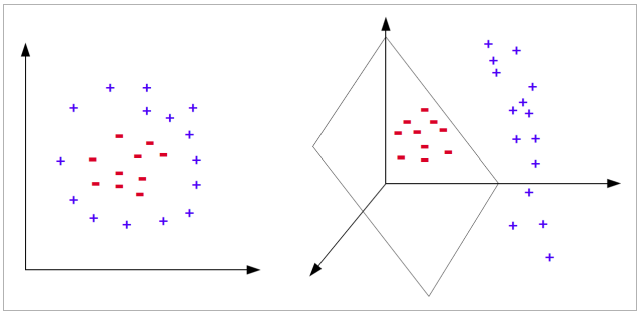
Constant c is the regularization parameter.

4.10. The Kernel Trick

Motivation

- SVMs (and logistic regression) are linear.
- But many classification problems are not …
- Kernel trick: map into higher dimension, use linear classifier there.
- Not just for SVMs, but closely linked with them.

Figure 4.8: SVM Kernel Trick Example – Mapping 2D data into 3D



The "***Kernel***" Math tells us: ANY mapping from pairs to matrix entries is associated with SOME high dimensional space, as long as we get a positive semi-definite matrix. This mapping is called a "kernel".

The "***Trick***": It's possible to learn an SVM without explicitly performing mapping. To do this, start with the "**dual**" formulation of the problem:

- Select αs to Maximize the following:

$$\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j)$$

Subject to: $\forall i \;\; 0 \geq \alpha_i \leq \frac{1}{\lambda} \quad and \quad \sum_i \alpha_i y_i = 0$

We want to select αs.

- Mathematically equivalent to the last one.
- Key observation: each x_i only used as input to dot product.
- So, no need to explicitly map to high dimension, as long as we have n by n matrix, where each entry is pairwise dot product in high-D, we are fine!

4.10.1. Standard Kernels

- ***Polynomial kernel***: Replace $(x_i \cdot x_j)$ with $(1 + (x_i \cdot x_j))^d$ for $d > 0$
- ***Gaussian kernel***: Replace $(x_i \cdot x_j)$ with $\exp(-||x_i - x_j||^2/(2\sigma^2))$

4.10.2. Kernel Trick Advantages/Disadvantages

Advantages:

- Can give better classification accuracy.
- Often used in practice for this reason.

Disadvantages:

- Often sensitive to parameter (σ in Gaussian kernel, for example)
- Computationally more complex…
- Dual formulation not easily amenable to Gradient Descent
- Means kernels useful mostly for smaller problems

4.11. Logistic Regression vs. SVM

Logistic Regression and SVM often give the similar results because training data is linearly separable, which happens very often and there is no need to project data to higher dimensions.

- SVM costs longer to train than logistic regression.
- SVM naturally regularizes.
- SVM gives better results when training sample size is relatively small, and we have to deal with lots of features.

- But not much difference between “linear SVM” and “regularized Logistic Regression”.
- Regularized LR seems more common in “Big Data”.
- If the dataset is not linearly separable Soft-margin SVM may lead to better performance.
- Logistic regression is the most frequently used algorithm in industry.

But kernel trick makes SVM standard choice for small data.

4.12. Further Reading References and Links

- Book: Leskovec et al. “Mining of Massive Datasets”. Chapter 12. Section 12.3
- Bias and Variance
- Video about Vladimir Vapnik