HT2015: SC4
Statistical Data Mining and Machine Learning

Dino Sejdinovic
Department of Statistics
Oxford

http://www.stats.ox.ac.uk/~sejdinov/sdmml.html

Generative vs Discriminative Learning

- **Generative learning**: find parameters which **explain all the data available**.

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log p(x_i, y_i|\theta)$$

  Examples: LDA, naïve Bayes.
  - Makes use of all the data available.
  - Flexible framework, can incorporate other tasks, incomplete data.
  - Stronger modelling assumptions.
- **Discriminative learning**: find parameters that aid in **prediction**.

$$\hat{\theta} = \operatorname*{argmin}_{\theta} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f_\theta(x_i)) \quad \text{or} \quad \hat{\theta} = \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log p(y_i|x_i, \theta)$$

  Examples: logistic regression, support vector machines.
  - Typically performs better on a given task.
  - Weaker modelling assumptions.
  - Can overfit more easily.

Generative Learning

- We work with a joint distribution $p_{X,Y}(x,y)$ over data vectors and labels.
- A learning algorithm: construct $f : \mathcal{X} \to \mathcal{Y}$ which predicts the label of X.
- Given a loss function L, the risk R of $f(X)$ is

$$R(f) = \mathbb{E}_{p_{X,Y}}[L(Y, f(X))]$$

- For 0/1 loss in classification, Bayes classifier

$$f_{\text{Bayes}}(x) = \underset{k=1,\ldots,K}{\operatorname{argmax}}\, p(Y=k|x) = \underset{k=1,\ldots,K}{\operatorname{argmax}}\, p_{X,Y}(x,k)$$

  has the minimum risk (Bayes risk), but is unknown since $p_{X,Y}$ is unknown.
- Assume a parameteric model for the joint: $p_{X,Y}(x,y) = p_{X,Y}(x,y|\theta)$
- Fit $\hat{\theta} = \operatorname{argmax}_\theta \sum_{i=1}^n \log p(x_i, y_i|\theta)$ and plug in back to Bayes classifier:

$$\hat{f}(x) = \underset{k=1,\ldots,K}{\operatorname{argmax}}\, p_{X,Y}(x,k|\hat{\theta}).$$

Hypothesis space and Empirical Risk Minimization

- Find best function in $\mathcal{H}$ minimizing the risk:

$$f_\star = \operatorname*{argmin}_{f \in \mathcal{H}} \mathbb{E}_{X,Y}[L(Y, f(X))]$$

- **Empirical Risk Minimization** (ERM): minimize the empirical risk instead, since we typically do not know $P_{X,Y}$.

$$\hat{f} = \operatorname*{argmin}_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i))$$

- Hypothesis space $\mathcal{H}$ is the space of functions f under consideration.
- How complex should we allow functions f to be? If hypothesis space $\mathcal{H}$ is "too large", ERM will overfit. Function

$$\hat{f}(x) = \begin{cases} y_i & \text{if } x = x_i, \\ 0 & \text{otherwise} \end{cases}$$

will have zero empirical risk, but is useless for generalization, since it has simply "memorized" the dataset.

Training and Test Performance

- **Training error** is the empirical risk

$$\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))$$

  For 0-1 loss in classification, this is the misclassification error on the training data $\{x_i, y_i\}_{i=1}^n$, **which were used in learning** f.
- **Test error** is the empirical risk on **new, previously unseen** observations $\{\tilde{x}_i, \tilde{y}_i\}_{i=1}^m$

$$\frac{1}{m}\sum_{i=1}^{m} L(\tilde{y}_i, f(\tilde{x}_i))$$

  which were NOT used in learning f.
- Test error is a much better gauge of how well learned function **generalizes** to new data.
- The test error is in general larger than the training error.

Hypothesis space for two-class LDA

- Assume we have two classes $\{+1, -1\}$.
- Recall that the discriminant functions in LDA are linear. Assuming that data vectors in class k is modelled as $\mathcal{N}(\mu_k, \Sigma)$, choosing class $+1$ over -1 involves:

$$a_{+1} + b_{+1}^\top x > a_{-1} + b_{-1}^\top x \qquad \Leftrightarrow \qquad a_\star + b_\star^\top x > 0,$$

  where $a_\star = a_{+1} - a_{-1}$, $b_\star = b_{+1} - b_{-1}$.
- Thus, hypothesis space of two-class LDA consists of functions $f(x) = \mathrm{sign}(a + b^\top x)$.
- We obtain coefficients $\hat{a}$ and $\hat{b}$, and thus the function $\hat{f}$ through fitting the parameters of the generative model.
- **Discriminative learning**: restrict $\mathcal{H}$ to a class of functions $f(x) = \mathrm{sign}(a + b^\top x)$ and select $\hat{a}$ and $\hat{b}$ which minimize empirical risk.

Space of linear decision functions

- Hypothesis space $\mathcal{H} = \{f : f(x) = \text{sign}(a + b^\top x), a \in \mathbb{R}, b \in \mathbb{R}^p\}$
- Find a, b that minimize the empirical risk under 0-1 loss:

$$\begin{aligned}
&\operatorname*{argmin}_{a,b} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f_{a,b}(x_i)) \\
&= \operatorname*{argmin}_{a,b} \frac{1}{n} \sum_{i=1}^{n} \begin{cases} 0 & \text{if } y_i = \text{sign}(a + b^\top x_i) \\ 1 & \text{otherwise} \end{cases} \\
&= \operatorname*{argmin}_{a,b} \frac{1}{2n} \sum_{i=1}^{n} \left[1 - \text{sign}(y_i(a + b^\top x_i))\right].
\end{aligned}$$

- Combinatorial problem - not typically possible to solve...
- Maybe easier with a different loss function? (Logistic regression)

Linearity of log-posterior odds

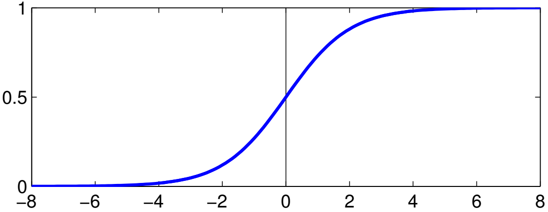
- Another way to express linear decision boundary of LDA:

$$\log \frac{p(Y=+1|X=x)}{p(Y=-1|X=x)} = a + b^\top x.$$

- Solve explicitly for conditional class probabilities:

$$p(Y=+1|X=x) = \frac{1}{1+\exp(-(a+b^\top x))} =: s(a+b^\top x)$$
$$p(Y=-1|X=x) = \frac{1}{1+\exp(+(a+b^\top x))} = s(-a-b^\top x)$$

where $s(\cdot)$ is the **logistic function**



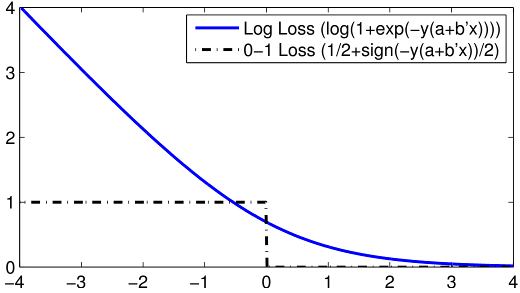
Logistic Regression

- Consider maximizing the **conditional log likelihood**:

$$\ell(a,b) = \sum_{i=1}^{n} \log p(Y = y_i | X = x_i) = \sum_{i=1}^{n} -\log(1 + \exp(-y_i(a + b^\top x_i)))$$

- Equivalent to minimizing the empirical risk associated with the **log loss**:

$$R_{\text{log}}^{\text{emp}} = \frac{1}{n}\sum_{i=1}^{n} \log(1 + \exp(-y_i(a + b^\top x_i))) = \frac{1}{n}\sum_{i=1}^{n}\left[-\log(\underbrace{s(y_i(a + b^\top x_i))}_{\hat{p}_{a,b}(y_i|x_i)})\right]$$



Logistic Regression

- Not possible to find optimal a, b analytically.
- For simplicitiy, absorb a as an entry in b by appending '1' into x vector.
- Objective function:

$$R_{\text{log}}^{\text{emp}} = \frac{1}{n} \sum_{i=1}^{n} -\log s(y_i x_i^\top b)$$

Logistic Function

$$\begin{aligned} s(-z) &= 1 - s(z) \\ \nabla_z s(z) &= s(z)s(-z) \\ \nabla_z \log s(z) &= s(-z) \\ \nabla_z^2 \log s(z) &= -s(z)s(-z) \end{aligned}$$

- Differentiate wrt b:

$$\nabla_b R_{\text{log}}^{\text{emp}} = \frac{1}{n} \sum_{i=1}^{n} -s(-y_i x_i^\top b) y_i x_i$$

$$\nabla_b^2 R_{\text{log}}^{\text{emp}} = \frac{1}{n} \sum_{i=1}^{n} s(y_i x_i^\top b) s(-y_i x_i^\top b) x_i x_i^\top$$

Logistic Regression

- Second derivative is positive-definite: objective function is **convex** and there is **a single unique global minimum**.
- Many different algorithms can find optimal b, e.g.:
  - Gradient descent:
    $$b^{\text{new}} = b + \epsilon \frac{1}{n} \sum_{i=1}^{n} s(-y_i x_i^\top b) y_i x_i$$
  - Stochastic gradient descent:
    $$b^{\text{new}} = b + \epsilon_t \frac{1}{|I(t)|} \sum_{i \in I(t)} s(-y_i x_i^\top b) y_i x_i$$
    where $I(t)$ is a subset of the data at iteration t, and $\epsilon_t \to 0$ slowly ($\sum_t \epsilon_t = \infty, \sum_t \epsilon_t^2 < \infty$).
  - Newton-Raphson:
    $$b^{\text{new}} = b - (\nabla_b^2 R_{\text{log}}^{\text{emp}})^{-1} \nabla_b R_{\text{log}}^{\text{emp}}$$
    This is also called **iterative reweighted least squares**.
  - Conjugate gradient, LBFGS and other methods from numerical analysis.

Logistic Regression vs. LDA

- Both have linear decision boundaries and model log-posterior odds as

$$\log \frac{p(Y=+1|X=x)}{p(Y=-1|X=x)} = a + b^\top x$$

- LDA models the marginal density of x as a Gaussian mixture with shared covariance

$$g(x) = \pi_{-1}\mathcal{N}(x;\mu_{-1},\Sigma) + \pi_{+1}\mathcal{N}(x;\mu_{+1},\Sigma)$$

  and fits the parameters $\theta = (\mu_{-1},\mu_{+1},\pi_{-1},\pi_{+1},\Sigma)$ by maximizing joint likelihood $\sum_{i=1}^{n} p(x_i, y_i|\theta)$. a and b are then determined from θ.
- Logistic regression leaves the marginal density $g(x)$ as an **arbitrary density function**, and fits the parameters a,b by maximizing the conditional likelihood $\sum_{i=1}^{n} p(y_i|x_i;a,b)$.

Logistic Regression

Properties of logistic regression:

- Makes less modelling assumptions than generative classifiers.
- A simple example of a generalised linear model (GLM). Much statistical theory:
  - assessment of fit via deviance and plots,
  - interpretation of entries of b as **odds-ratios**,
  - fitting categorical data (sometimes called **multinomial logistic regression**),
  - well founded approaches to removing insignificant features (drop-in deviance test, Wald test).

Example: Spam Dataset

A data set collected at Hewlett-Packard Labs, that classifies 4601 e-mails as spam or non-spam. 57 variables indicate the frequency of certain words and characters.

```
> library(kernlab)
> data(spam)
> dim(spam)
[1] 4601   58
> spam[1:2,]
  make address  all num3d  our over remove internet order mail receive will
1 0.00    0.64 0.64     0 0.32 0.00   0.00     0.00     0 0.00    0.00 0.64
2 0.21    0.28 0.50     0 0.14 0.28   0.21     0.07     0 0.94    0.21 0.79
  people report addresses free business email  you credit your font num000
1   0.00   0.00      0.00 0.32     0.00  1.29 1.93      0 0.96    0   0.00
2   0.65   0.21      0.14 0.14     0.07  0.28 3.47      0 1.59    0   0.43
  money hp hpl george num650 lab labs telnet num857 data num415 num85
1  0.00  0   0      0      0   0    0      0      0    0      0     0
2  0.43  0   0      0      0   0    0      0      0    0      0     0
  technology num1999 parts pm direct cs meeting original project re edu table
1          0    0.00     0  0      0  0       0        0       0  0   0 0
2          0    0.07     0  0      0  0       0        0       0  0   0 0
  conference charSemicolon charRoundbracket charSquarebracket charExclamation
1          0             0            0.000                 0           0.778
2          0             0            0.132                 0           0.372
  charDollar charHash capitalAve capitalLong capitalTotal type
1       0.00    0.000      3.756          61          278 spam
2       0.18    0.048      5.114         101         1028 spam
> str(spam$type)
 Factor w/ 2 levels "nonspam","spam": 2 2 2 2 2 2 2 2 2 2 ...
```

Spam Dataset

Use logistic regression to predict spam/not spam.

```
## let Y=0 be non-spam and Y=1 be spam.
Y <- as.numeric(spam$type)-1
X <- spam[ ,-ncol(spam)]

gl <- glm(Y ~ ., data=X,family=binomial)
```

Which predictor variables seem to be important? Can for example check which ones are significant in the GLM.

```
> summary(gl)
Call:
glm(formula = Y ~ ., family = binomial, data = X)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-4.127e+00  -2.030e-01  -1.967e-06   1.140e-01   5.364e+00
```

Spam Dataset

```
Coefficients:
                   Estimate Std. Error z value Pr(>|z|)    
(Intercept)      -1.569e+00  1.420e-01 -11.044  < 2e-16 ***
make             -3.895e-01  2.315e-01  -1.683 0.092388 .  
address          -1.458e-01  6.928e-02  -2.104 0.035362 *  
all               1.141e-01  1.103e-01   1.035 0.300759    
num3d             2.252e+00  1.507e+00   1.494 0.135168    
our               5.624e-01  1.018e-01   5.524 3.31e-08 ***
over              8.830e-01  2.498e-01   3.534 0.000409 ***
remove            2.279e+00  3.328e-01   6.846 7.57e-12 ***
internet          5.696e-01  1.682e-01   3.387 0.000707 ***
order             7.343e-01  2.849e-01   2.577 0.009958 ** 
mail              1.275e-01  7.262e-02   1.755 0.079230 .  
receive          -2.557e-01  2.979e-01  -0.858 0.390655    
will             -1.383e-01  7.405e-02  -1.868 0.061773 .  
people           -7.961e-02  2.303e-01  -0.346 0.729557    
report            1.447e-01  1.364e-01   1.061 0.288855    
addresses         1.236e+00  7.254e-01   1.704 0.088370 .  
business          9.599e-01  2.251e-01   4.264 2.01e-05 ***
email             1.203e-01  1.172e-01   1.027 0.304533    
you               8.131e-02  3.505e-02   2.320 0.020334 *  
credit            1.047e+00  5.383e-01   1.946 0.051675 .  
```

Spam Dataset

your	2.419e-01	5.243e-02	4.615	3.94e-06	***
font	2.013e-01	1.627e-01	1.238	0.215838	
num000	2.245e+00	4.714e-01	4.762	1.91e-06	***
money	4.264e-01	1.621e-01	2.630	0.008535	**
hp	-1.920e+00	3.128e-01	-6.139	8.31e-10	***
hpl	-1.040e+00	4.396e-01	-2.366	0.017966	*
george	-1.177e+01	2.113e+00	-5.569	2.57e-08	***
num650	4.454e-01	1.991e-01	2.237	0.025255	*
lab	-2.486e+00	1.502e+00	-1.656	0.097744	.
labs	-3.299e-01	3.137e-01	-1.052	0.292972	
telnet	-1.702e-01	4.815e-01	-0.353	0.723742	
num857	2.549e+00	3.283e+00	0.776	0.437566	
data	-7.383e-01	3.117e-01	-2.369	0.017842	*
num415	6.679e-01	1.601e+00	0.417	0.676490	
num85	-2.055e+00	7.883e-01	-2.607	0.009124	**
technology	9.237e-01	3.091e-01	2.989	0.002803	**
num1999	4.651e-02	1.754e-01	0.265	0.790819	
parts	-5.968e-01	4.232e-01	-1.410	0.158473	
pm	-8.650e-01	3.828e-01	-2.260	0.023844	*
direct	-3.046e-01	3.636e-01	-0.838	0.402215	
cs	-4.505e+01	2.660e+01	-1.694	0.090333	.
meeting	-2.689e+00	8.384e-01	-3.207	0.001342	**
original	-1.247e+00	8.064e-01	-1.547	0.121978	
project	-1.573e+00	5.292e-01	-2.973	0.002953	**
re	-7.923e-01	1.556e-01	-5.091	3.56e-07	***

Spam Dataset

```
edu              -1.459e+00  2.686e-01  -5.434 5.52e-08 ***
table            -2.326e+00  1.659e+00  -1.402 0.160958
conference       -4.016e+00  1.611e+00  -2.493 0.012672 *
charSemicolon    -1.291e+00  4.422e-01  -2.920 0.003503 **
charRoundbracket -1.881e-01  2.494e-01  -0.754 0.450663
charSquarebracket -6.574e-01  8.383e-01  -0.784 0.432914
charExclamation   3.472e-01  8.926e-02   3.890 0.000100 ***
charDollar        5.336e+00  7.064e-01   7.553 4.24e-14 ***
charHash          2.403e+00  1.113e+00   2.159 0.030883 *
capitalAve        1.199e-02  1.884e-02   0.636 0.524509
capitalLong       9.118e-03  2.521e-03   3.618 0.000297 ***
capitalTotal      8.437e-04  2.251e-04   3.747 0.000179 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6170.2  on 4600  degrees of freedom
Residual deviance: 1815.8  on 4543  degrees of freedom
AIC: 1931.8

Number of Fisher Scoring iterations: 13
```

Spam Dataset

How good is the classification?

```
> proba <- predict(gl,type="response")
> predicted_spam <- as.numeric( proba>0.5)
> table(predicted_spam,Y)
              Y
predicted_spam    0    1
             0 2666  194
             1  122 1619

> predicted_spam <- as.numeric( proba>0.99)
> table(predicted_spam,Y)
              Y
predicted_spam    0    1
             0 2776 1095
             1   12  718
```

Advantage of a probabilistic approach: probabilities give interpretable confidence to predictions.

Spam Dataset

Success rate is calculated on the same data that the GLM is trained on!
Separate in training and test set.

```
n <- length(Y)
train <- sample( n, round(n/2) )
test<-(1:n)[-train]
```

Fit only on training set and predict on both training and test set.

```
gl <- glm(Y[train] ~ ., data=X[train,],family=binomial)

proba_train <- predict(gl,newdata=X[train,],type="response")
proba_test  <- predict(gl,newdata=X[test,],type="response")

predicted_spam_lr_train <- as.numeric(proba_train > 0.95)
predicted_spam_lr_test  <- as.numeric(proba_test > 0.95)
```

Spam Dataset

Results for training and test set:

```
> table(predicted_spam_lr_train, Y[train])
predicted_spam_lr_train    0    1
                      0 1398  363
                      1    9  530
```

```
> table(predicted_spam_lr_test, Y[test])
predicted_spam_lr_test    0    1
                     0 1357  357
                     1   24  563
```

Note: testing performance is worse than training performance.

Spam Dataset

Compare with LDA.

```
library(MASS)
lda_res <- lda(x=X[train,],grouping=Y[train])

proba_lda_test <- predict(lda_res,newdata=X[test,])$posterior[,2]
predicted_spam_lda_test <- as.numeric(proba_lda_test > 0.95)

> table(predicted_spam_lr_test, Y[test])
predicted_spam_lr_test    0    1
                     0 1357  357
                     1   24  563

> table(predicted_spam_lda_test, Y[test])
predicted_spam_lda_test    0    1
                      0 1365  534
                      1   16  386
```

It looks like LDA might be beating logistic regression here, but this is across a single threshold 0.95. Does this persist across multiple thresholds?
Answer: **ROC curves**.

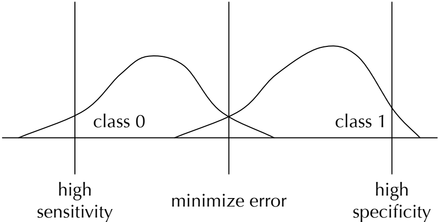
Performance Measures

- **Confusion matrix**:

True	state	0	1
Prediction	0	**# true negative**	**# false negative**
	1	**# false positive**	**# true positive**

  - **Accuracy**: $(TP + TN)/(TP + TN + FP + FN)$.
  - **Error rate**: $(FP + FN)/(TP + TN + FP + FN)$.
  - **Sensitivity (true positive rate)**: $TP/(TP + FN)$.
  - **Specificity (true negative rate)**: $TN/(TN + FP)$.
  - **Precision**: $TP/(TP + FP)$.
  - **Recall** (same as Sensitivity): $TP/(TP + FN)$.
  - **F1**: harmonic mean of precision and recall.

- As we vary the prediction threshold c from 0 to 1:
  - Specificity varies from 0 to 1.
  - Sensitivity goes from 1 to 0.



ROC Curves

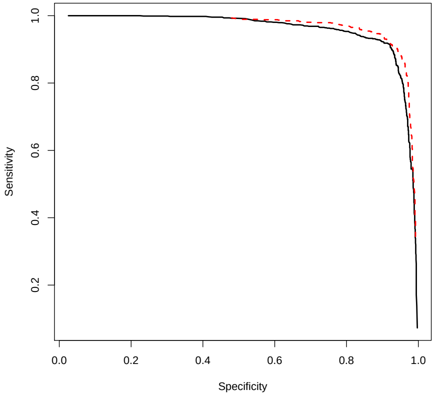
ROC curve plots sensitivity versus specificity as threshold varies.

```
cvec <- seq(0.001,0.999,length=1000)
specif <- numeric(length(cvec))
sensit <- numeric(length(cvec))
speciflr <- numeric(length(cvec))
sensitlr <- numeric(length(cvec))

for (cc in 1:length(cvec)){
  sensit[cc] <- sum( proba_lda_test> cvec[cc] & Y[test]==1)/sum(Y[test]==1)
  specif[cc] <- sum( proba_lda_test<=cvec[cc] & Y[test]==0)/sum(Y[test]==0)
  sensitlr[cc] <- sum( proba_test> cvec[cc] & Y[test]==1)/sum(Y[test]==1)
  speciflr[cc] <- sum( proba_test<=cvec[cc] & Y[test]==0)/sum(Y[test]==0)
}
plot(specif,sensit,xlab="Specificity",ylab="Sensitivity",type="l",lwd=2)
lines(speciflr,sensitlr,col='red',lwd=2)
```

ROC Curves

ROC curve: LDA = black/full; LR = red/dashed.



LR beats LDA on this dataset in terms of **area under ROC**:
Wilcoxon-Mann-Whitney statistic: probability that the classifier will score a randomly drawn positive example higher than a randomly drawn negative example.

ROC Curves

R library ROCR contains various performance measures

```
library(ROCR)
> pred_lda <- prediction(proba_lda_test,Y[test])
> perf_lda <- performance(pred_lda, "tpr", "tnr")
> pred_lr <- prediction(proba_test,Y[test])
> perf_lr <- performance(pred_lr, "tpr", "tnr")
> plot(perf_lda,lwd=2)
> plot(perf_lr,add=TRUE,col='red',lwd=2,lty="dashed")
> auc_lda <- as.numeric(performance(pred_lda,"auc")@y.values)
> auc_lda
[1] 0.9580931
> auc_lr <- as.numeric(performance(pred_lr,"auc")@y.values)
> auc_lr
[1] 0.9668392
```