Generative and Discriminative Learning

- **Generative learning**: find parameters that **explains all the data**.

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

  Examples: LDA, Naïve Bayes.
  - Makes use of all the data.
  - Flexible framework, can incorporate other tasks.
  - Stronger modelling assumptions.
- **Discriminative learning**: find parameters that help to **predict relevant data**.

  $$\theta^* = \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log p(y_i | x_i, \theta) \qquad \text{or} \qquad f^* = \operatorname*{argmin}_{f} \sum_{i=1}^{n} L(y_i, f(X_i))$$

  Examples: linear and logistic regression, rest of the course.
  - Learns to perform better on the given task.
  - Weaker modelling assumptions.
  - Can overfitting more easily.

Statistical Learning Theory

- We work with a joint distribution $p^*(X, Y)$ over data vectors and labels.
- A learning algorithm constructs a function $f(X)$ which predicts the label of X.
- Given a loss function L, the risk R of $f(X)$ is

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

  For classification, the best function $f^*(X)$ is the Bayes classifier, achieving the minimum risk (Bayes risk).
- Hypothesis space $\mathcal{H}$ is the space of functions under consideration.
- Find best function minimizing the risk:

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

  - **Empirical Risk Minimization**: minimize the empirical risk instead, since we typically do not know $p^*(X, Y)$.
  - **Regularization**: Large hypothesis spaces can lead to overfitting,

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

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, **which were used in learning** $f(x)$.
- **Test error** is the empirical risk on **new, previously unseen**, observations

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

  which were NOT used in learning.
- 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.

Logistic Regression

- 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:

$$a_{+1} + b_{+1}^\top x > a_{-1} + b_{-1}^\top x \quad \Leftrightarrow \quad (a_{+1} - a_{-1}) + (b_{+1} - b_{-1})^\top x > 0$$

- If we care about minimizing classification errors, we can try to find a, b to minimize directly the average misclassification error (empirical risk associated with 0-1 loss):

$$\begin{aligned} &\operatorname*{argmin}_{a,b} \frac{1}{n} \sum_{i=1}^{n} \begin{cases} 0 & \text{if } y_i = \operatorname{sign}(a + b^\top x) \\ 1 & \text{otherwise} \end{cases} \\ = &\operatorname*{argmin}_{a,b} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} - \frac{1}{2} \operatorname{sign}(y_i(a + b^\top x)) \end{aligned}$$

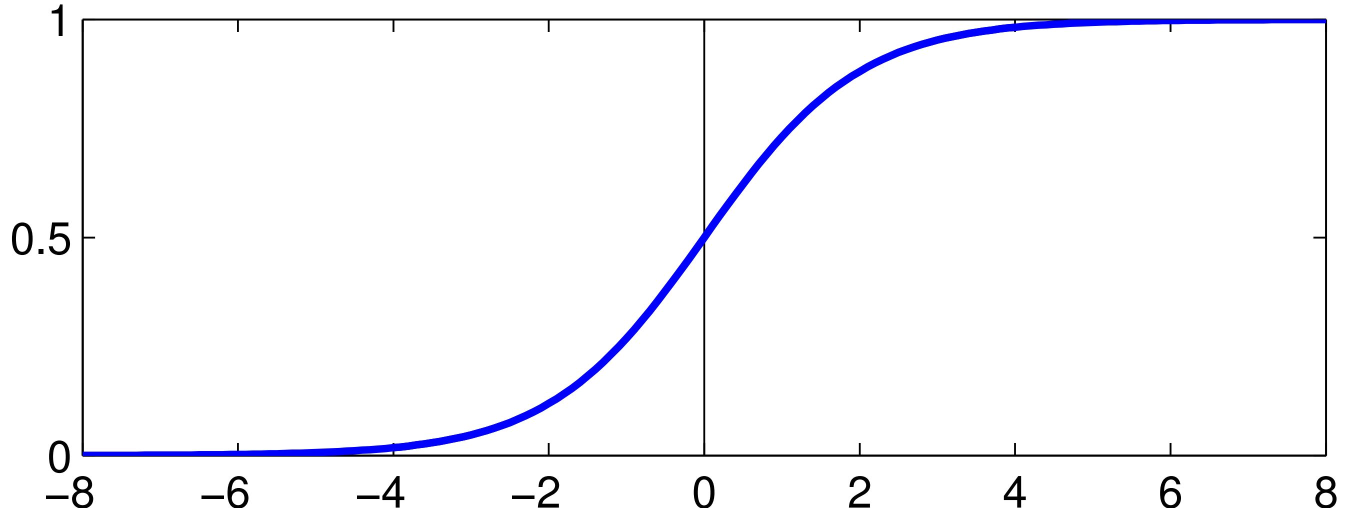
- An example of **Empirical Risk Minimization**. Unfortunately not typically possible to solve...

Logistic Regression

- **Logistic regression** replaces the 0-1 loss with the log loss.
- A model parameterizing the conditional distribution of labels given data vectors:

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

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



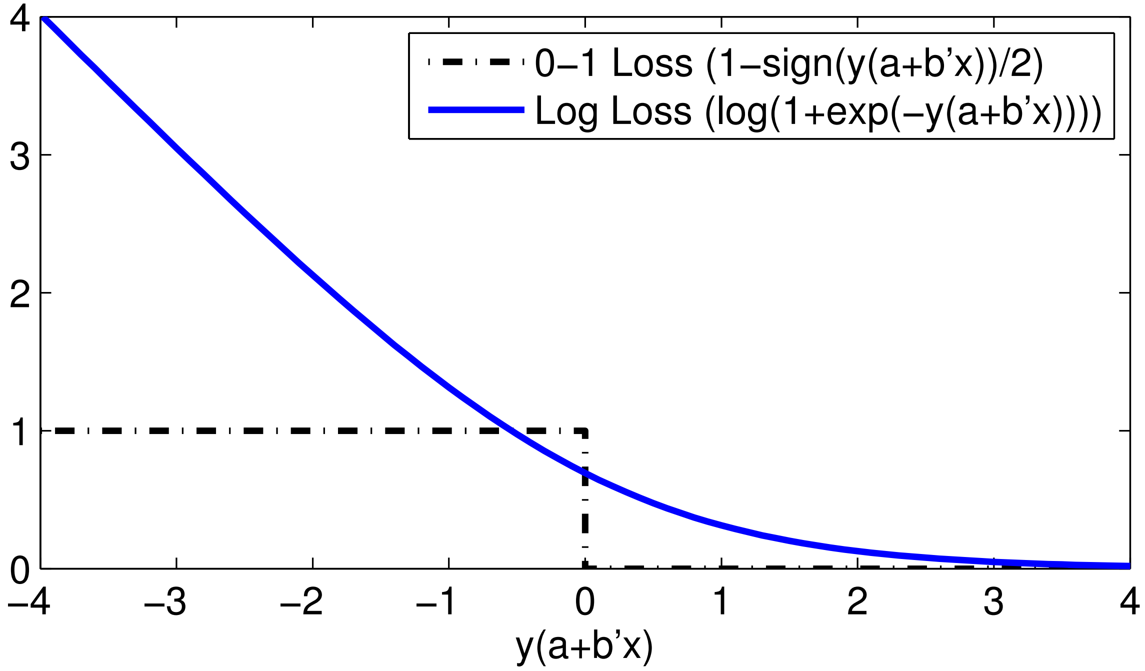
Logistic Regression

- 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**:

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



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:

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

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:

$$\nabla_b R^{\text{emp}}_{\text{log}} = \frac{1}{n}\sum_{i=1}^{n} -s(-y_i x_i^\top b) y_i x_i = \frac{1}{n}\sum_{i=1}^{n} -((.5 + .5y_i) - s(x_i^\top b))x_i$$

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

Logistic Regression

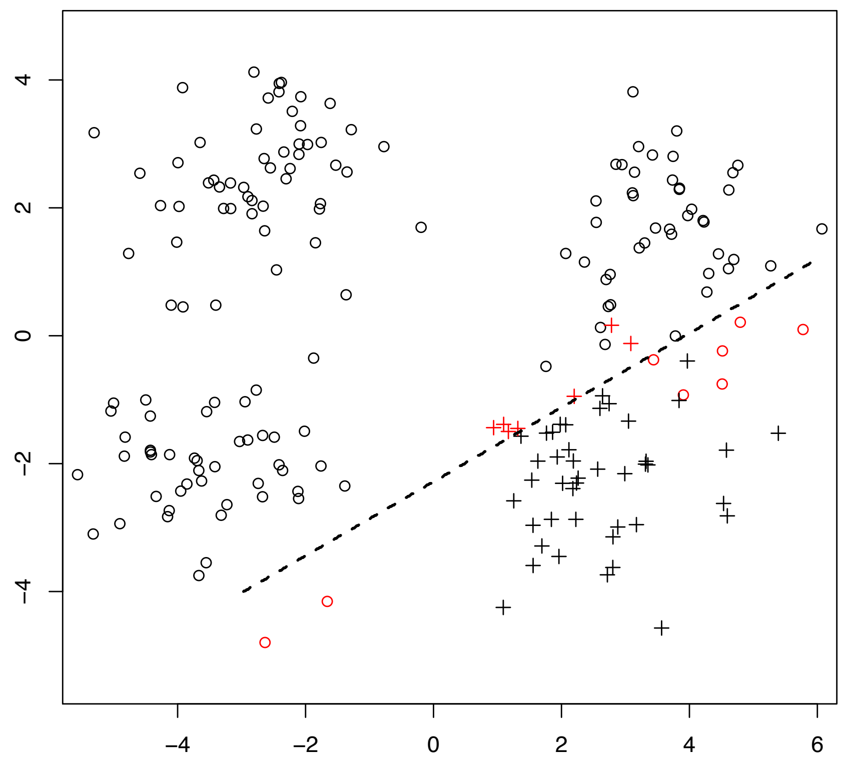
- Second derivative is positive-definite: objective function is **convex** and there is **a single unique global minimum**.
- Many different algorithms can find optimal b, e.g.:
  - Gradient descent:
    $$b^{\text{new}} = b + \epsilon \frac{1}{n} \sum_{i=1}^{n} s(-y_i x_i^\top b) y_i x_i$$
  - Stochastic 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.

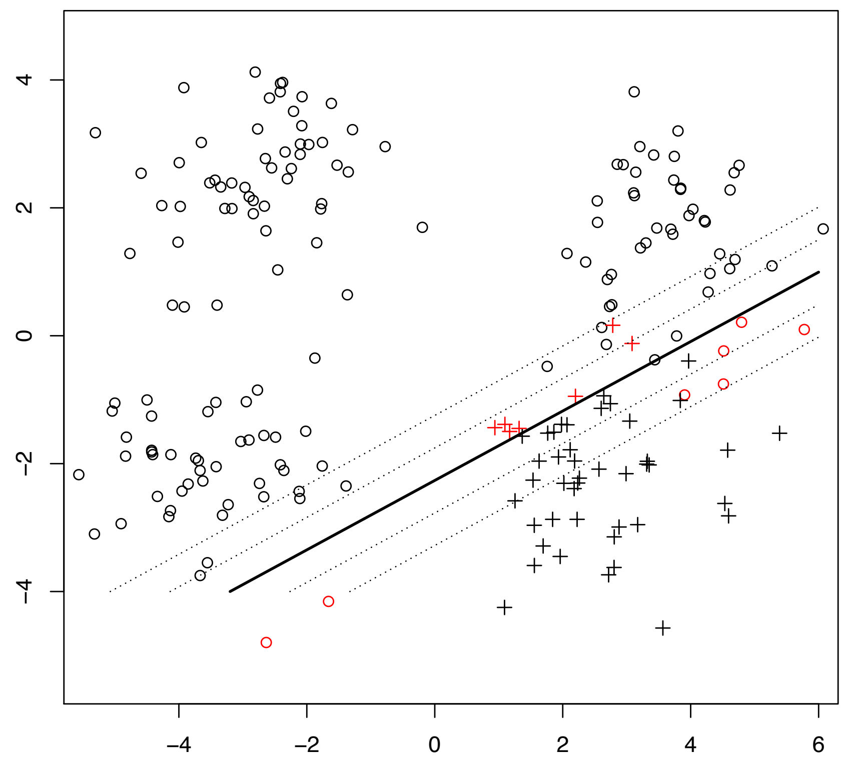
Logistic Regression

Properties of logistic regression:

- Makes less modelling assumptions than LDA and naïve Bayes.
- Models only the conditional distribution of labels, not the marginal distribution of X.
- A linear method: decision boundary is a separating hyperplane.
- Logistic regression can be made **non-linear** by applying a non-linear transformation $X \mapsto \phi(X)$.
- Logistic regression is 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),

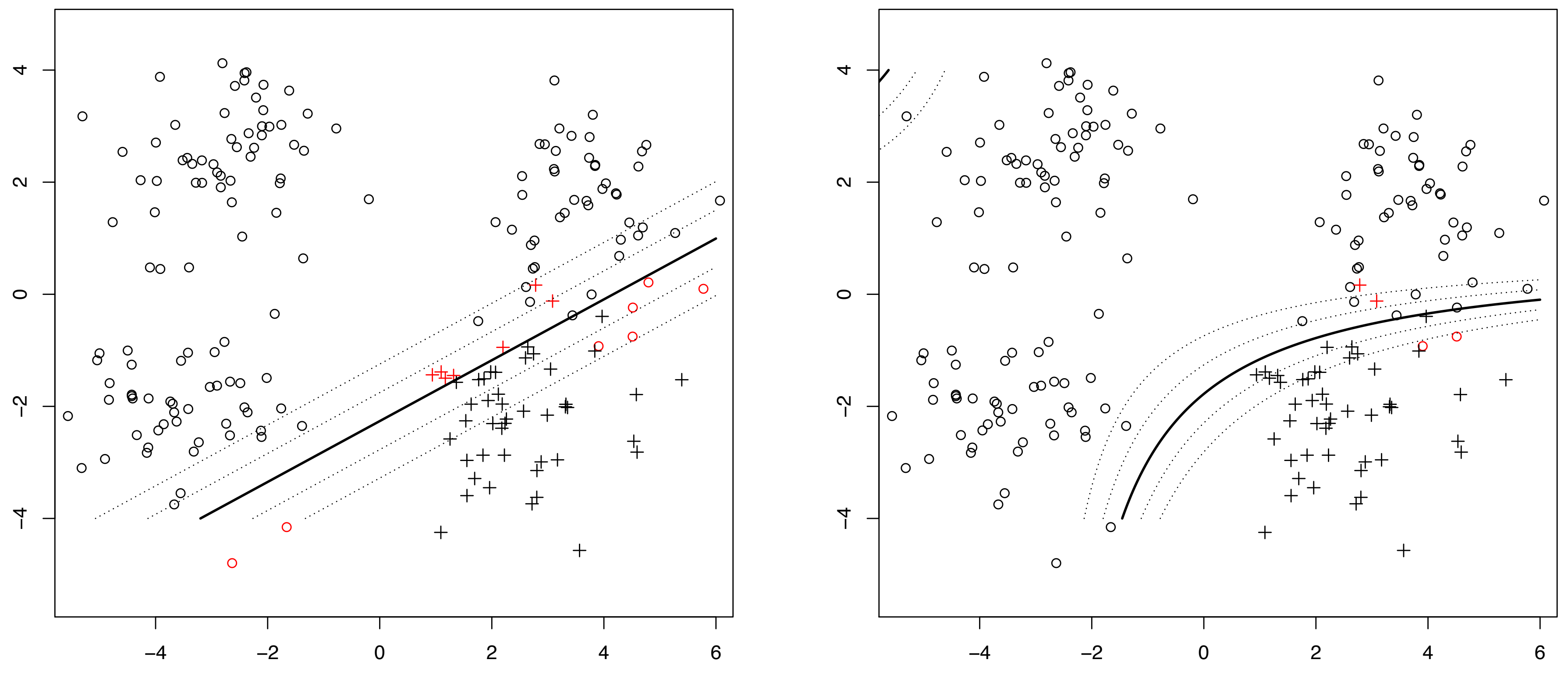
Crab Dataset





Comparing LDA and logistic regression.

Crab Dataset



Comparing logistic regression with and without quadratic interactions.

Crab Dataset

```
library(MASS)
## load crabs data
data(crabs)
ct <- as.numeric(crabs[,1])-1+2*(as.numeric(crabs[,2])-1)
## project into first two LD
cb.lda <- lda(log(crabs[,4:8]),ct)
cb.ldp <- predict(cb.lda)
x <- cb.ldp$x[,1:2]
y <- as.numeric(ct==0)
eqscplot(x,pch=2*y+1,col=y+1)

## visualize decision boundary
gx1 <- seq(-6,6,.02)
gx2 <- seq(-4,4,.02)
gx <- as.matrix(expand.grid(gx1,gx2))
gm <- length(gx1)
gn <- length(gx2)
gdf <- data.frame(LD1=gx[,1],LD2=gx[,2])

lda <- lda(x,y)
y.lda <- predict(lda,x)$class
eqscplot(x,pch=2*y+1,col=2-as.numeric(y==y.lda))
y.lda.grid <- predict(lda,gdf)$class
contour(gx1,gx2,matrix(y.lda.grid,gm,gn),
   levels=c(0.5), add=TRUE,d=FALSE,lty=2,lwd=2)
```

Crab Dataset

```
## logistic regression
xdf <- data.frame(x)
logreg <- glm(y ~ LD1 + LD2, data=xdf, family=binomial)
y.lr <- predict(logreg,type="response")
eqscplot(x,pch=2*y+1,col=2-as.numeric(y==(y.lr>.5)))
y.lr.grid <- predict(logreg,newdata=gdf,type="response")
contour(gx1,gx2,matrix(y.lr.grid,gm,gn),
   levels=c(.1,.25,.75,.9), add=TRUE,d=FALSE,lty=3,lwd=1)
contour(gx1,gx2,matrix(y.lr.grid,gm,gn),
   levels=c(.5), add=TRUE,d=FALSE,lty=1,lwd=2)
```

```
## logistic regression with quadratic interactions
logreg <- glm(y ~ (LD1 + LD2)^2, data=xdf, family=binomial)
y.lr <- predict(logreg,type="response")
eqscplot(x,pch=2*y+1,col=2-as.numeric(y==(y.lr>.5)))
y.lr.grid <- predict(logreg,newdata=gdf,type="response")
contour(gx1,gx2,matrix(y.lr.grid,gm,gn),
   levels=c(.1,.25,.75,.9), add=TRUE,d=FALSE,lty=3,lwd=1)
contour(gx1,gx2,matrix(y.lr.grid,gm,gn),
   levels=c(.5), add=TRUE,d=FALSE,lty=1,lwd=2)
```

Spam Dataset

```
> library(kernlab)
> data(spam)
> dim(spam)
[1] 4601   58

> spam[1:2,]
  make address  all num3d  our over remove internet order mail receive wil
1 0.00    0.64 0.64     0 0.32 0.00   0.00     0.00     0 0.00    0.00 0.6
2 0.21    0.28 0.50     0 0.14 0.28   0.21     0.07     0 0.94    0.21 0.7
  people report addresses free business email  you credit your font num00C
1   0.00   0.00      0.00 0.32     0.00  1.29 1.93      0 0.96    0   0.0C
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 ta
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 charExclamat
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
```

Spam Dataset

Use logistic regression to predict spam/not spam.

```
library(kernlab)
data(spam)

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

So out of 730 emails marked as spam, 12 were actually not spam.
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)
i <- sample( rep(c(TRUE,FALSE),each=n/2),round(n) ,replace=FALSE )
train <- (1:n)[i]
test  <- (1:n)[!i]
```

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_train <- as.numeric(proba_train > 0.95)
predicted_spam_test  <- as.numeric(proba_test > 0.95)
```

Spam Dataset

Results for training and test set:

```
> table(predicted_spam_train, Y[train])
predicted_spam_train    0    1
                   0 1403  354
                   1   11  567
```

```
> table(predicted_spam_test, Y[test])
predicted_spam_test    0    1
                  0 1346  351
                  1   28  541
```

It is no coincidence that test performance is worse than training performance.

Spam Dataset

Compare with LDA.

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

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

> table(predicted_spam_test, Y[test])
predicted_spam_test    0    1
                  0 1346  351
                  1   28  541



> table(predicted_spam_lda, Y[test])
predicted_spam_lda    0    1
                 0 1364  533
                 1   10  359
```

It seems as if LDA beats logistic regression here, but would need to adjust decision threshold to get proper comparison. Use **ROC curves**.

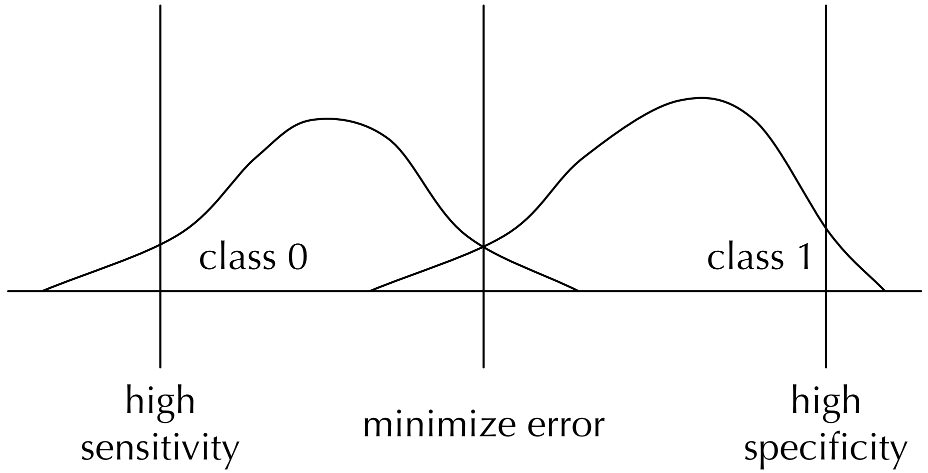
Performance Measures

- **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**: $TP/(TP + FN)$.
  - **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

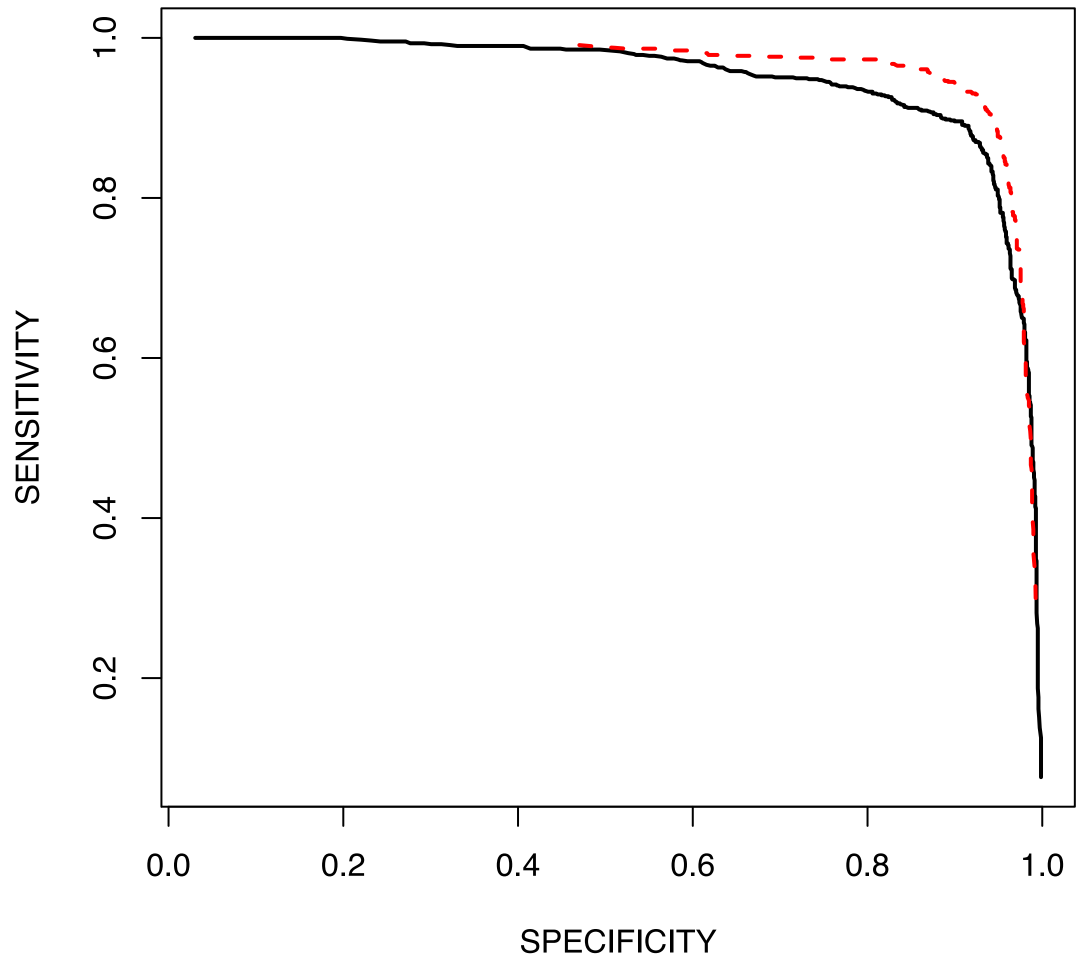
ROC curve plots sensitivity versus specificity as threshold varies.

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

for (cc in 1:length(cvec)){
  sensit[cc] <- sum( proba_lda> cvec[cc] & Y[test]==1)/sum(Y[test]==1)
  specif[cc] <- sum( proba_lda<=cvec[cc] & Y[test]==0)/sum(Y[test]==0)
}
plot(specif,sensit,xlab="SPECIFICITY",ylab="SENSITIVITY",type="l",lwd=2)
```

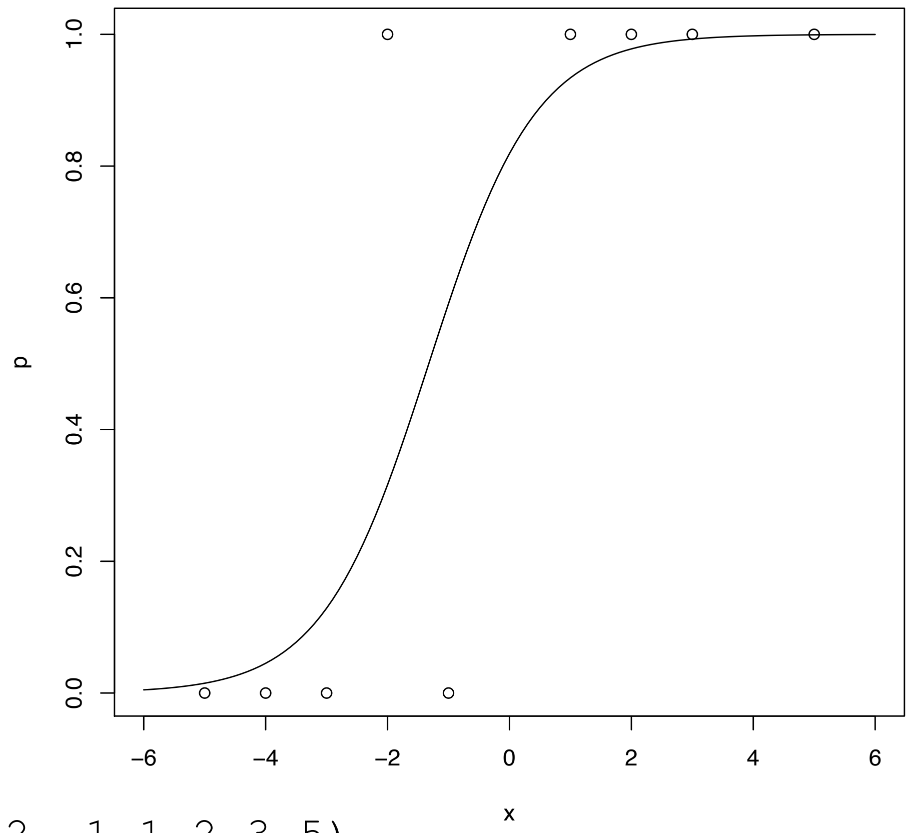
ROC Curves

ROC curve for LDA and logistic regression classification of spam dataset. LDA = unbroken black line; LR = broken red line.



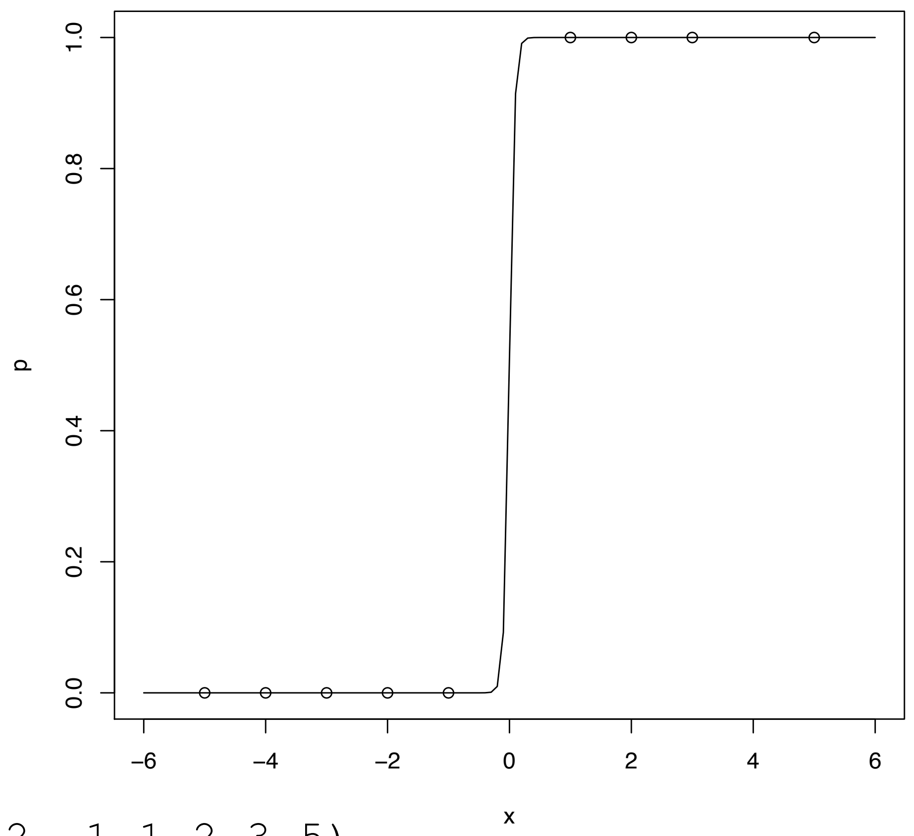
Obvious now that LR is better for this dataset than LDA, contrary to the first impression.

Overfitting in Logistic Regression



```
dx <- c(-5,-4,-3,-2,-1,1,2,3,5)
d <- data.frame(dx)
x <- seq(-6,6,.1)
y <- c(0,0,0,1,0,1,1,1,1)
lr <- glm(y ~ ., data=d,family=binomial)
p <- predict(lr,newdata=data.frame(dx=x),type="response")
plot(x,p,type="l")
points(dx,y)
```

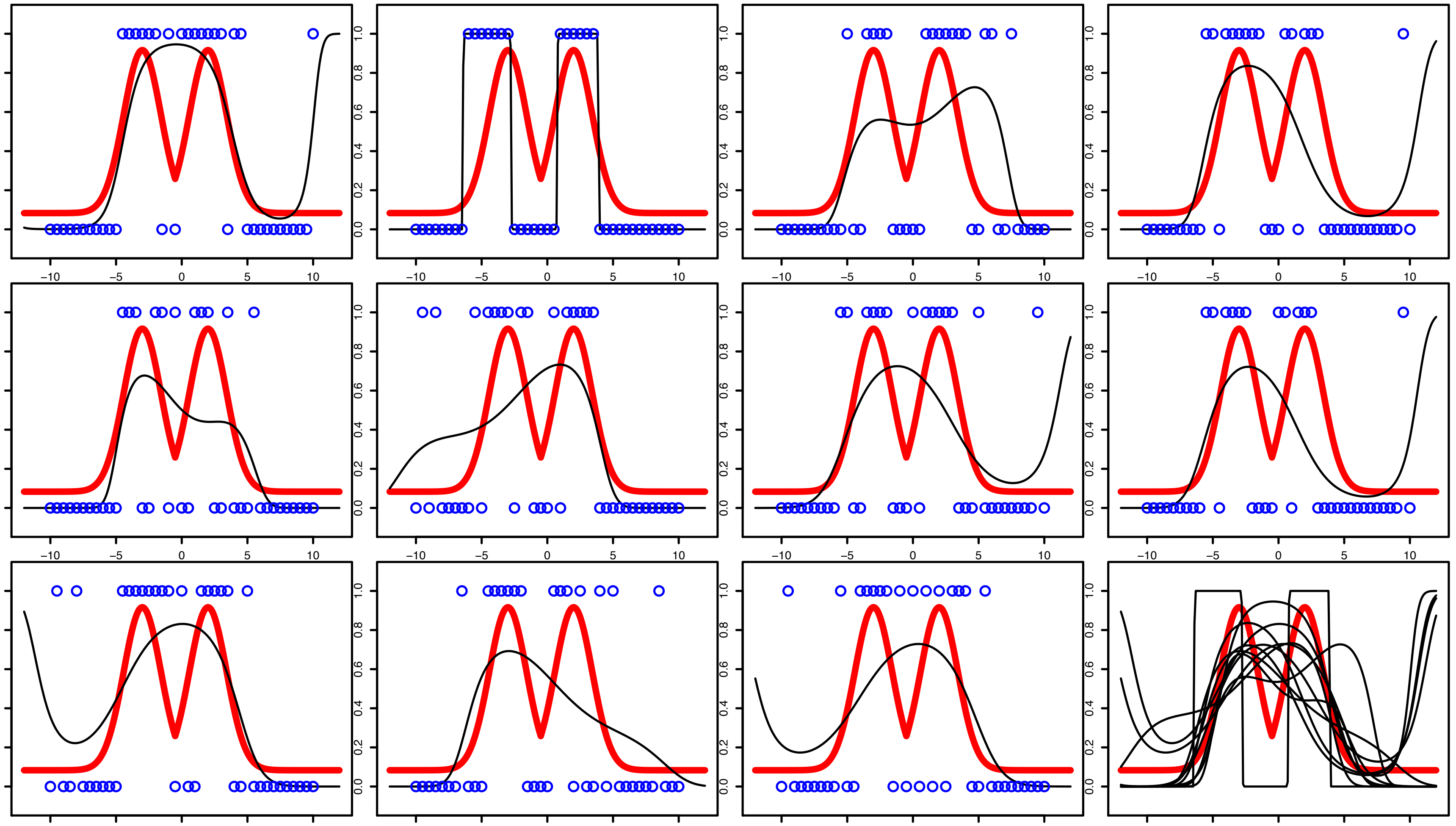
Overfitting in Logistic Regression



```
dx <- c(-5,-4,-3,-2,-1,1,2,3,5)
d <- data.frame(dx)
x <- seq(-6,6,.1)
y <- c(0,0,0,0,0,1,1,1,1)
lr <- glm(y ~ ., data=d,family=binomial)
p <- predict(lr,newdata=data.frame(dx=grid),type="response")
plot(x,p,type="l")
points(dx,y)
```

Demo on Overfitting in Logistic Regression

True conditional probabilities in Red. Blue circles are training data, Black curve is predicted conditional probability. 11 datasets are sampled from true distribution and used to learn a logistic regression model with non-linear features $\phi(x) = (1, x, x^2, \ldots, x^{p-1})$.



Demo on Overfitting in Logistic Regression

```
## true conditional probabilities
truep <- function(x) {
  return((pmax(exp(-(x-2)^2/4),exp(-(x+3)^2/4))+.1)/1.2)
}
## features are {x^i}
phi <- function(x,deg) {
  d <- matrix(0,length(x),deg+1)
  for (i in 0:deg) {
    d[,i+1] <- x ^ i
  }
  return (data.frame(d))
}
## demo learning logistic regression, with different datasets generated,
## and using different degree polynomials as features
demolearn <- function(trainx,testx,truep,deg) {
  trainp <- truep(trainx)
  testp  <- truep(testx)
  par(mfrow=c(3,4),ann=FALSE,cex=.3,mar=c(1,1,1,1))
  predp <- matrix(0,length(testx),11)
  for (i in 1:11) {
    trainy <- as.numeric(runif(length(trainx)) < trainp)
    lr <- glm(trainy ~ .,data=phi(trainx,deg),family=binomial)
    predp[,i] <- predict(lr,newdata=phi(testx,deg),type="response")
    plot(testx,testp,type="l",col=2,lwd=3,ylim=c(-.1,1.1))
    lines(testx,predp[,i],type="l")
    points(trainx,trainy,pch=1,col=4,cex=2)
  }
  plot(testx,testp,type="l",lwd=3,col=2,ylim=c(-.1,1.1))
  for (i in 1:11) {
    lines(testx,predp[,i],type="l")
  }
  return(predp)
}

trainx <- seq(-10,10,.5)
testx  <- seq(-12,12,.1)
pp <- demolearn(trainx,testx,truep,4)
```

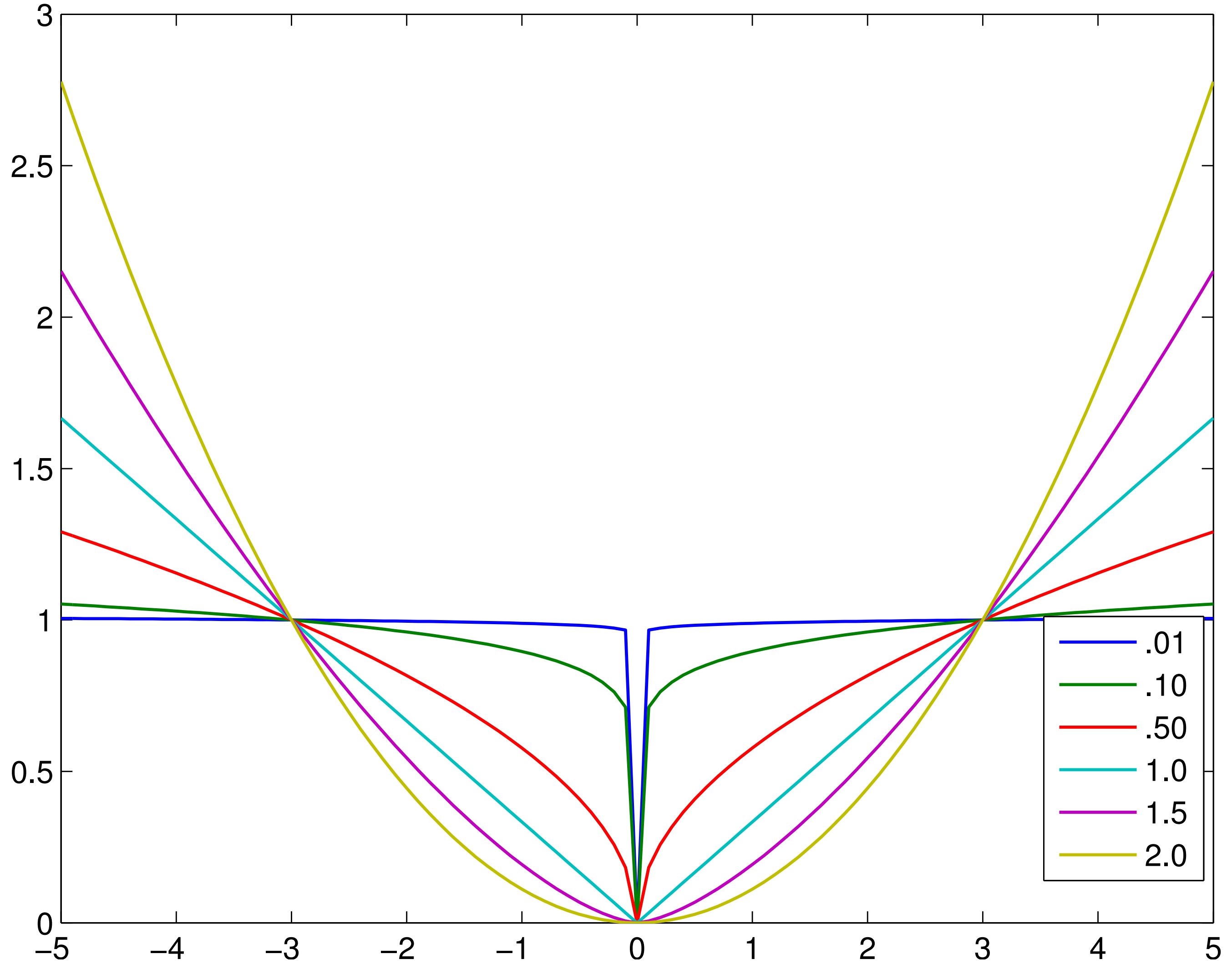
Regularization

- Flexible models for high-dimensional problems require many parameters.
- With many parameters, learners can easily overfit to the noise in the training data.
- **Regularization**: Limit flexibility of model to prevent overfitting.
- Typically: add term penalizing large values of parameters θ.

$$R^{\text{emp}}(\theta) + \lambda\|\theta\|_\rho^\rho = \frac{1}{n}\sum_{i=1}^{n} \log(1+\exp(-y_i(a+b^\top x_i))) + \lambda\|b\|_\rho^\rho$$

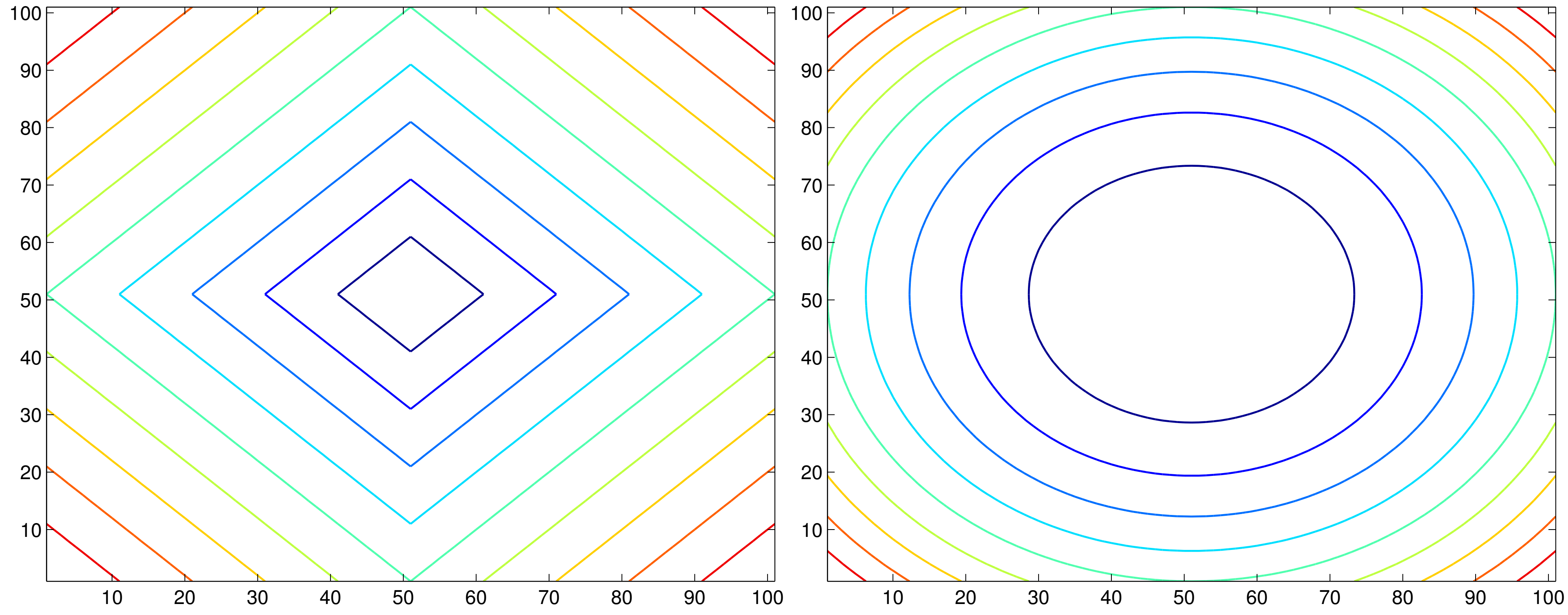
  where $\rho \in [1,2]$, and $\|z\|_\rho = (\sum_{j=1}^{p} |z_j|^\rho)^{1/\rho}$ is the L_ρ norm of b (also of interest when $\rho \in [0,1)$, but is no longer a norm).
- Typical cases are $\rho = 2$ (Euclidean norm, **ridge regression**) and $\rho = 1$ (**LASSO**). When $\rho \leq 1$ it is called a **sparsity** inducing regularization.
- λ is a **tuning parameter** (or **hyperparameter**) and controls the amount of regularization, and resulting complexity of the model.

Regularization



L_ρ regularization profile for different values of ρ.

Regularization



L_1 and L_2 norm contours.

Regularization

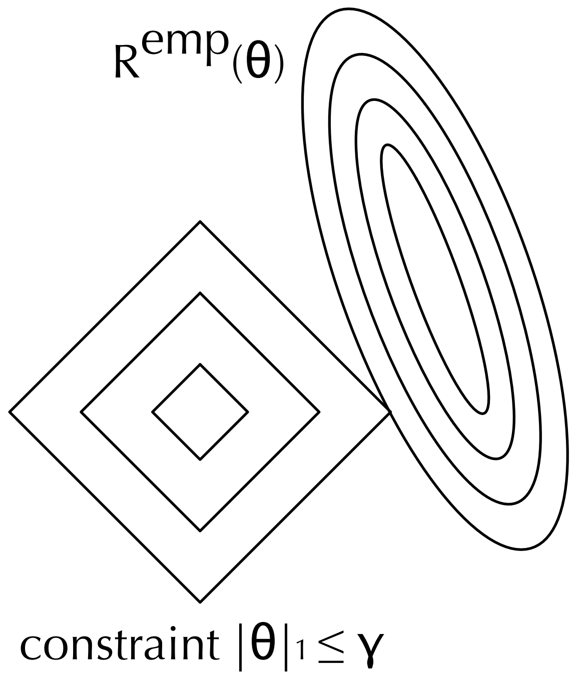
- Consider constrained optimization problem

$$\min_{\theta} R^{\text{emp}}(\theta) \text{ s.t. } \|\theta\|_1 < \gamma$$

- Lagrange multiplier $\lambda > 0$ to enforce constraint,

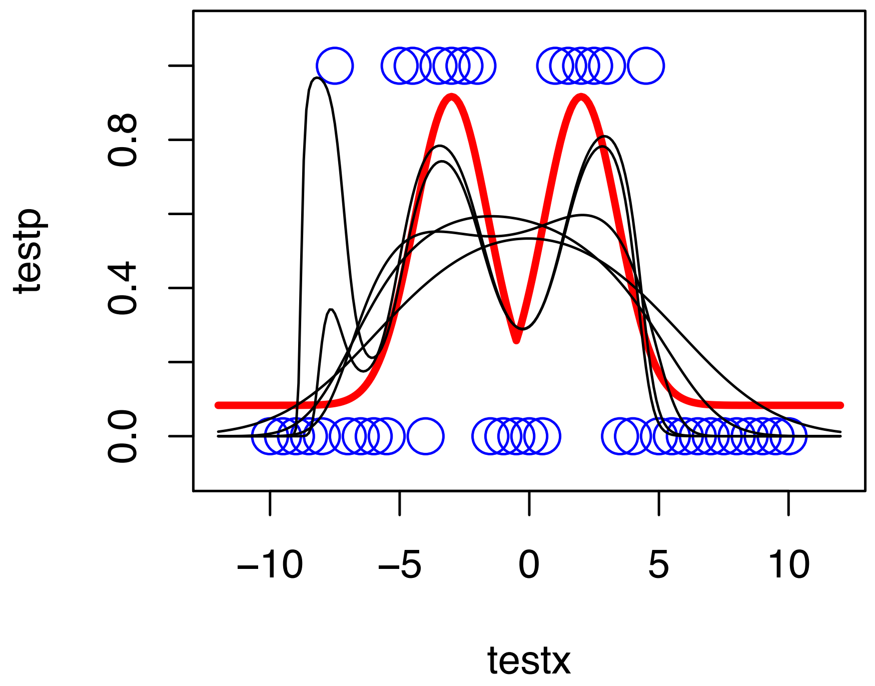
$$\min_{\theta} R^{\text{emp}}(\theta) + \lambda(\|\theta\|_1 - \gamma)$$

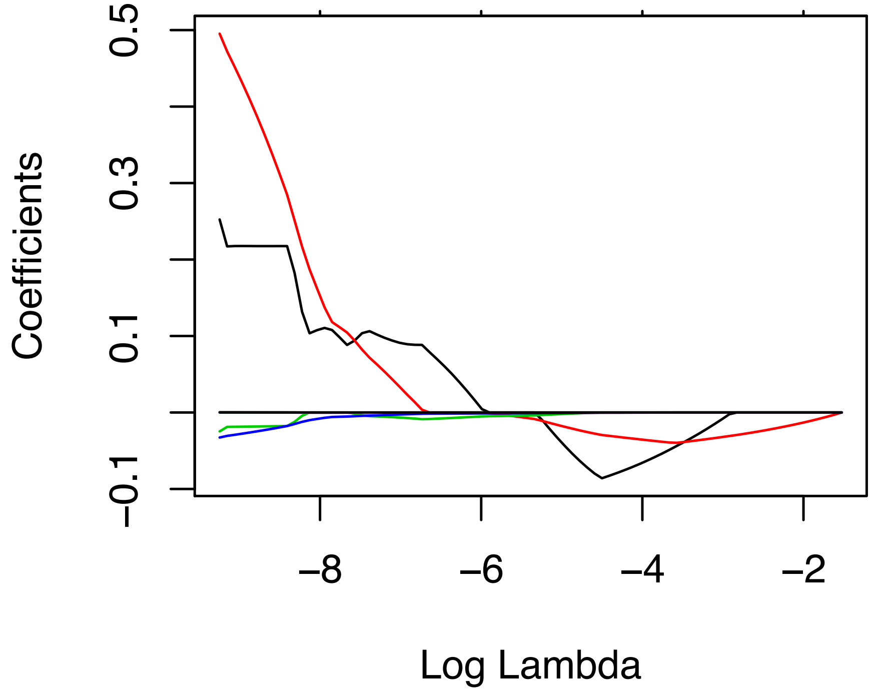
  - At the optimal value of λ, the parameter θ is the one minimizing the regularized empirical risk objective.
  - Conversely, given λ, there is a value of γ such that the corresponding optimal Lagrange multiplier is λ.
- Using L_1 regularization, optimal θ has $\theta_2 = 0$.
- Generally: L_1 regularization leads to optimal solutions with many zeros, i.e. the regression function depends only on the (small) number of features with non-zero parameters.



Demo on L_1 Regularized Logistic Regression

Use `glmnet` for regression with L_1, L_2 and combination regularization.





Demo on L_1 Regularized Logistic Regression

```
## true conditional probabilities
truep <- function(x) {
  return((pmax(exp(-(x-2)^2/4),exp(-(x+3)^2/4))+.1)/1.2)
}
## features are {x^i}
phi <- function(x,deg) {
  d <- matrix(0,length(x),deg+1)
  for (i in 0:deg) {
    d[,i+1] <- x ^ i
  }
  return (data.frame(d))
}
## demo L1 regularized learning of logistic regression,
## with different datasets generated, and using different
## degree polynomials as features

trainx <- seq(-10,10,.5)
testx  <- seq(-12,12,.1)

demolearnL1 <- function(trainx,testx,truep,deg) {
  trainp <- truep(trainx)
  testp  <- truep(testx)
  trainy <- as.numeric(runif(length(trainx)) < trainp)
  slr <- glmnet(as.matrix(phi(trainx,deg)),as.factor(trainy),
          family="binomial")
  s <- c(0,.0001,.001,.01,.05)
  predp <- predict(slr,newx=as.matrix(phi(testx,deg)),
        s=s,type="response")
  par(mfrow=c(1,2),mar=c(4,4,1,2))
  plot(testx,testp,type="l",col=2,lwd=3,ylim=c(-.1,1.1))
  points(trainx,trainy,pch=1,col=4,cex=2)
  for (i in 1:dim(predp)[2]) {
    lines(testx,predp[,i],type="l")
  }
  plot(slr,xvar="lambda")
  print(coef(slr,s))
  return(predp)
}

demolearnL1(trainx,testx,truep,10)
```