Unit 5
Linear Classification & Logistic Regression

Prof. Phil Schniter

THE OHIO STATE UNIVERSITY

ECE 5307: Introduction to Machine Learning, Sp23

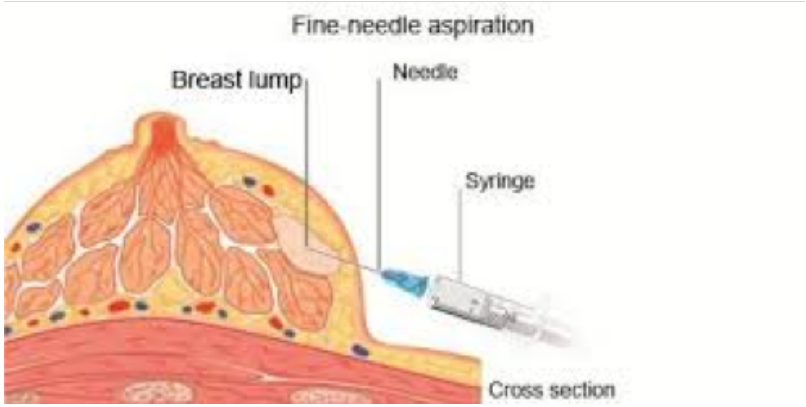
Learning objectives

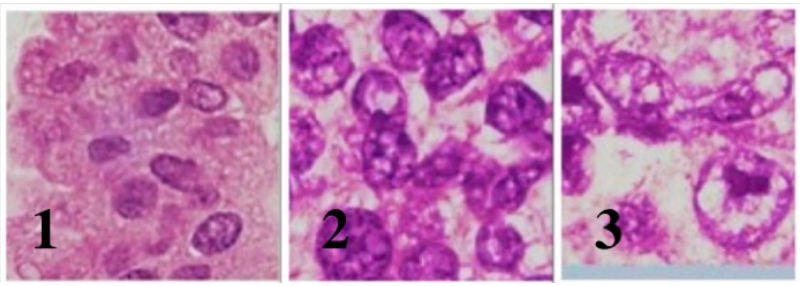
- Understand classification problems in machine learning:
  - Identify features, labels; binary vs multiclass; linear vs nonlinear
  - visualize scatterplots and decision regions
- For binary classification problems, understand ...
  - linear classifiers, separating hyperplanes, linearly separable data
  - effect of feature transformations
  - why LS linear regression doesn't work well
  - logistic regression: logistic function, cross-entropy loss, ML fitting, regularization
  - common error metrics: accuracy, precision, recall, F1
  - the effect of the decision threshold, ROC, AUC
- For multiclass classification problems, understand ...
  - solutions that use multiple binary classifiers
  - multinomial logistic regression: softmax function, cross-entropy loss, ML fitting
- How to implement and assess classification using sklearn

Outline

Diagnosing Breast Cancer

- Fine-needle aspiration of suspicious breast lumps:
  - Tissue is stained & viewed under microscope
  - Cytopathologist visually inspects cells and takes several measurements
  - Also tries to diagnose as benign or malignant
- Would like to improve diagnosis
- Can machine-learning help?





Grades of carcinoma cells
http://breast-cancer.ca/5a-types/

The Wisconsin Breast Cancer Data Set

Univ. of Wisconsin study:

- 683 samples
- 9 features (on right)
- target: malignant or benign
  - ground-truth was assessed using a biopsy

#	Attribute	Domain
1.	Sample code number	id number
2.	Clump Thickness	1 - 10
3.	Uniformity of Cell Size	1 - 10
4.	Uniformity of Cell Shape	1 - 10
5.	Marginal Adhesion	1 - 10
6.	Single Epithelial Cell Size	1 - 10
7.	Bare Nuclei	1 - 10
8.	Bland Chromatin	1 - 10
9.	Normal Nucleoli	1 - 10
10.	Mitoses	1 - 10
11.	Class:	(2 for benign, 4 for malignant)

Data:

https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data

Explanation:

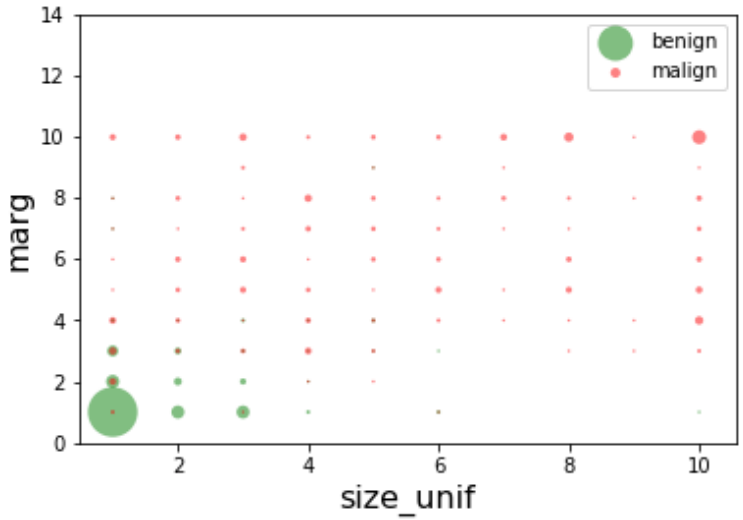
https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.names

Original paper:

O.L. Mangasarian, W.N. Street, and W.H. Wolberg, "Breast cancer diagnosis and prognosis via linear programming," *Operations Research*, 1995.

Visualizing the data

- Choose two features to visualize
- Plot target count vs. features using variable-radius dots (all are discrete)




```
# Compute the bin edges for the 2d histogram
x1val = np.array(list(set(X[:,0]))).astype(float)
x2val = np.array(list(set(X[:,1]))).astype(float)
x1, x2 = np.meshgrid(x1val,x2val)
x1e= np.hstack((x1val,np.max(x1val)+1))
x2e= np.hstack((x2val,np.max(x2val)+1))

# Make a plot for each class
yval = list(set(y))
color = ['g','r']
for i in range(len(yval)):
    I = np.where(y==yval[i])[0]
    cnt, x1e, x2e = np.histogram2d(X[I,0],X[I,1],[x1e,x2e])
    x1, x2 = np.meshgrid(x1val,x2val)
    plt.scatter(x1.ravel(), x2.ravel(), s=2*cnt.ravel(),alpha=0.5,
                c=color[i],edgecolors='none')
plt.ylim([0,14])
plt.legend(['benign','malign'], loc='upper right')
plt.xlabel(xnames[0], fontsize=16)
plt.ylabel(xnames[1], fontsize=16)
```

- How would we classify a test feature $\boldsymbol{x} = [x_1, x_2]$?

Outline

Binary classification

- Given training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, our goal is to classify a test vector $\boldsymbol{x}$ as $y \in \{-1, 1\}$ (one of two "classes")
  - Unlike regression, the target y is now binary
  - Could also use $\{0, 1\}$ or other target labels
- Many applications:
  - Are these cells cancerous or not?
  - Is this a weed or a crop?
- Mathematically, want to design a classifier

$$f(\boldsymbol{x}) = \widehat{y} \in \{-1, 1\}$$

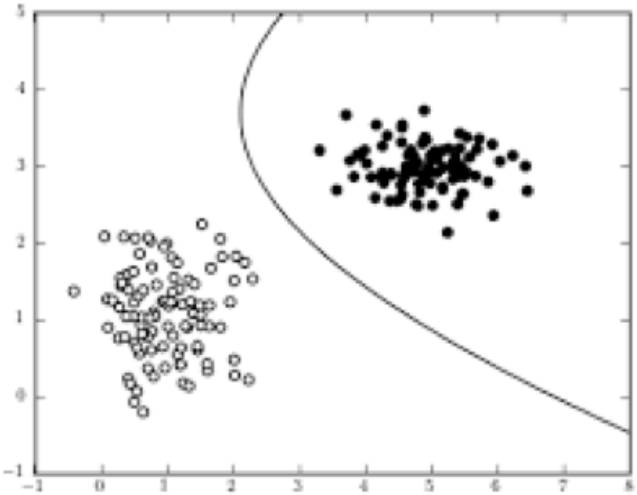
such that $\widehat{y} = y$ with high probability

- Note: $f(\cdot)$ is defined by its decision regions:

$\mathcal{R}_{-1} \triangleq \{\boldsymbol{x} : f(\boldsymbol{x}) = -1\}$ and $\mathcal{R}_1 \triangleq \{\boldsymbol{x} : f(\boldsymbol{x}) = 1\}$

Is this a weed or a crop?

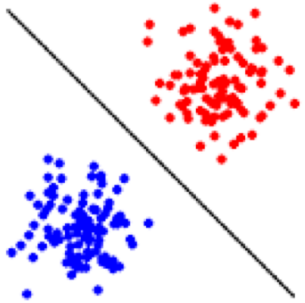
Example with $d = 2$:



Binary linear classification

- One option is binary *linear* classification:
  1) compute the "score" or "discriminant", $z = b + \sum_{j=1}^{d} x_j w_j$
     - z is linear in the parameters b and w_j
     - z is linear in the features x_j
  2) threshold the score to obtain $\widehat{y} = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$

  The weights $\boldsymbol{w} \triangleq [w_1, \ldots, w_d]^{\mathsf{T}}$ and intercept b are learned from the training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$



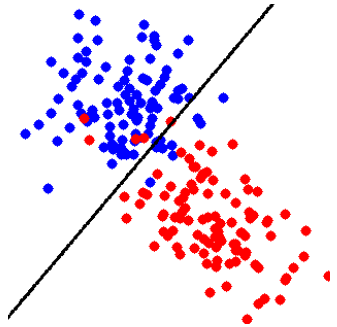
- Note that, at the decision boundary, we have

$$0 = z = b + \boldsymbol{x}^{\mathsf{T}} \boldsymbol{w}.$$

  Thus, the hyperplane

$$\{\boldsymbol{x} : b + \boldsymbol{x}^{\mathsf{T}} \boldsymbol{w} = 0\}$$

  separates the decision regions



- This decision boundary is linear in $\boldsymbol{x}$ (see figures). When does this perform well?

Linear separability

- Linear classification tends to perform well when the training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ is "linearly separable"
- Linearly separable means that there exists a hyperplane

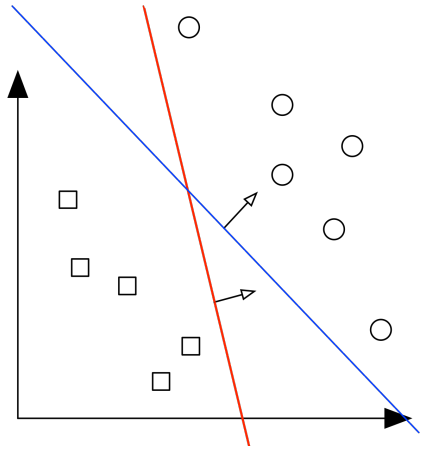
$$\{\boldsymbol{x} : b + \boldsymbol{x}^\mathsf{T}\boldsymbol{w} = 0\}$$

(for some b and $\boldsymbol{w}$) that separates the samples $\boldsymbol{x}_i$ according to their class $y_i \in \{-1, 1\}$

  - On one side of hyperplane lies all $\boldsymbol{x}_i$ for which $y_i = 1$, while on other other side lies all $\boldsymbol{x}_i$ for which $y_i = -1$

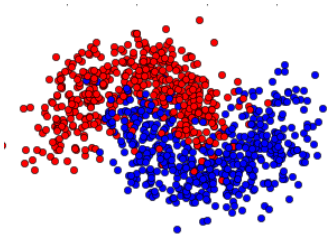
- Note: When such a separating hyperplane exists, it is usually non-unique

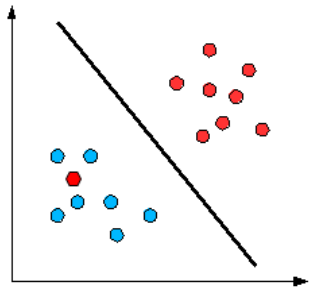
linear separability:

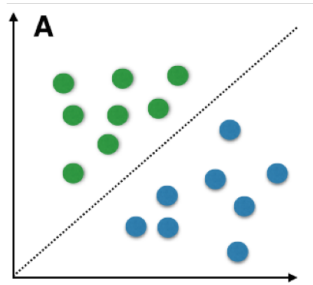


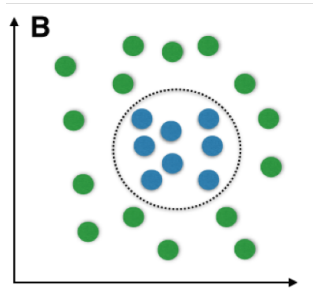
Linear versus nonlinear classification

- Most datasets are *not* linearly separable!
  - There are many possible reasons why
  - See examples on right (except Fig. A)
- Still, linear classification is worth considering
  - It is relatively easy to understand
  - It facilitates feature selection (i.e., provides intuition about which features matter most)
  - It can incorporate nonlinear feature transformations
    - Fig. A: boundary is linear in (x_1, x_2)
    - Fig. B: boundary is nonlinear in (x_1, x_2) but linear in a new feature $x' \triangleq \sqrt{x_1^2 + x_2^2}$
  - It is often used as a building block (e.g., for neural networks, decision trees, etc.)







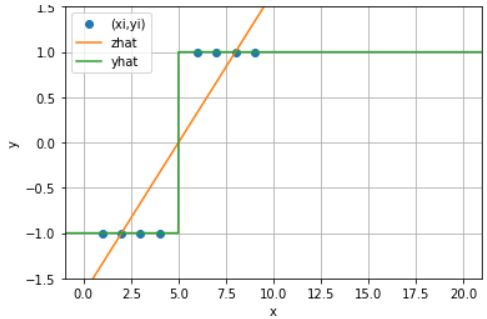


Linear classification vs. LS linear regression

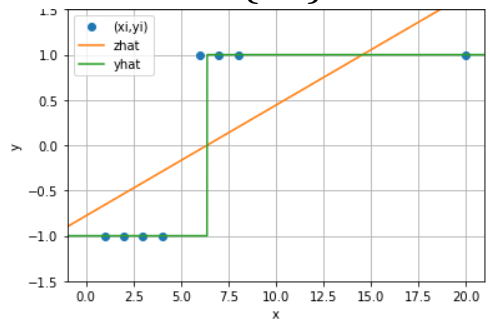
- Suppose we want to do *linear* classification. How exactly do we fit $(b, \boldsymbol{w})$?
- Can we just use LS, as we have been doing?
  - In other words, choose $(b, \boldsymbol{w})$ to minimize $\mathrm{RSS} = \left\| \boldsymbol{y} - \boldsymbol{A} \begin{bmatrix} b \\ \boldsymbol{w} \end{bmatrix} \right\|^2$ and then output

$$\widehat{y} = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}, \quad \text{where } z = [1 \ \boldsymbol{x}^{\mathsf{T}}] \begin{bmatrix} b_{\mathsf{ls}} \\ \boldsymbol{w}_{\mathsf{ls}} \end{bmatrix}$$

balanced $\{x_i\}$:



imbalanced $\{x_i\}$:



- No, this can fail terribly!
  - Consider simple case of $d = 1$ feature
  - When $\{x_i\}$ is "balanced" (top figure), works okay
  - But when $\{x_i\}$ is "imbalanced" (bottom), the least-squares regression line gets pulled to one side, and the threshold at $z = 0$ makes errors
- What's the problem?
  - RSS is not the right loss function for classification! (More details soon)

LS linear regression fails on breast-cancer classification!

- Let's try the same LS approach on the breast-cancer data (after converting targets to $y \in \pm 1$)
- Again, visualize decision boundary

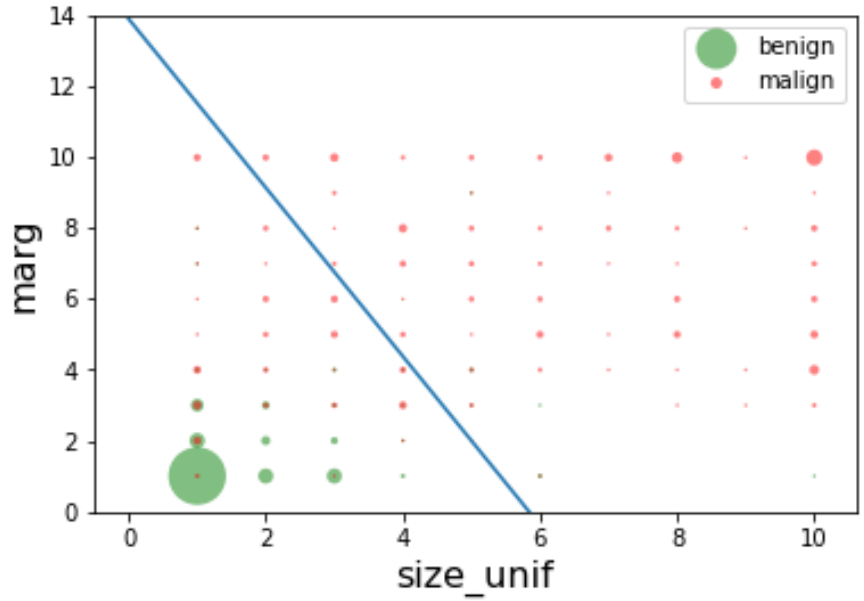
$$\{\boldsymbol{x} : b + \boldsymbol{x}^{\mathsf{T}}\boldsymbol{w} = 0\}$$

which, in this $d = 2$ case, becomes

$$\{(x_1, x_2) : x_2 = -\frac{b}{w_2} - \frac{w_1}{w_2}x_1\}$$

- Because the $\{\boldsymbol{x}_i\}$ are imbalanced, the decision boundary is pulled north-east, and many red $\boldsymbol{x}_i$ are misclassified!
- Again we see that designing $(b, \boldsymbol{w})$ to minimize RSS does not work well for linear classification!

blue line shows decision boundary




```
yhat = regr.predict(X)
yhati = (yhat >=0.5).astype(int)
acc = np.mean(yhati == y)
print("Accuracy on training data using two features = %f" % acc)
```

```
Accuracy on training data using two features = 0.922401
```

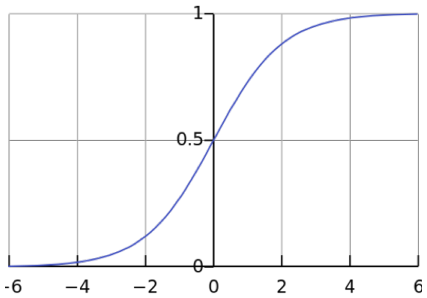
Outline

Binary logistic regression

- Linear classification computes $z = b + \boldsymbol{x}^{\mathsf{T}}\boldsymbol{w}$ and sets $\widehat{y} = \begin{cases} 1 & z \geq 0 \\ -1 & z < 0 \end{cases}$
- How do we design the parameters $(b, \boldsymbol{w})$?
  - We saw that minimizing RSS does not work well
- Idea: Given the score z, model the true label $y \in \pm 1$ as a random variable
- The most popular version of this uses the probabilistic model

$$\Pr\{y{=}1 \,|\, z\} = \frac{e^z}{1+e^z}, \quad \Pr\{y{=}{-}1 \,|\, z\} = \frac{1}{1+e^z}$$

  Note $\Pr\{y{=}{-}1 \,|\, z\} + \Pr\{y{=}1 \,|\, z\} = 1 \; \forall z$, as required for a valid pmf.
  - The larger that z is, the more likely that $y = 1$
  - When $z = 0$, it's equally likely that $y = 1$ or $y = -1$
- $\dfrac{e^z}{1+e^z} = \dfrac{1}{1+e^{-z}}$ is known as the "logistic function" or the "sigmoid"



Understanding the logistic model

Previously we considered $\Pr\{y=1\,|\,z\}$. What about $\Pr\{y=1\,|\,\boldsymbol{x}\}$?

- Consider the simple case of a single scalar feature x and intercept $b=0$:

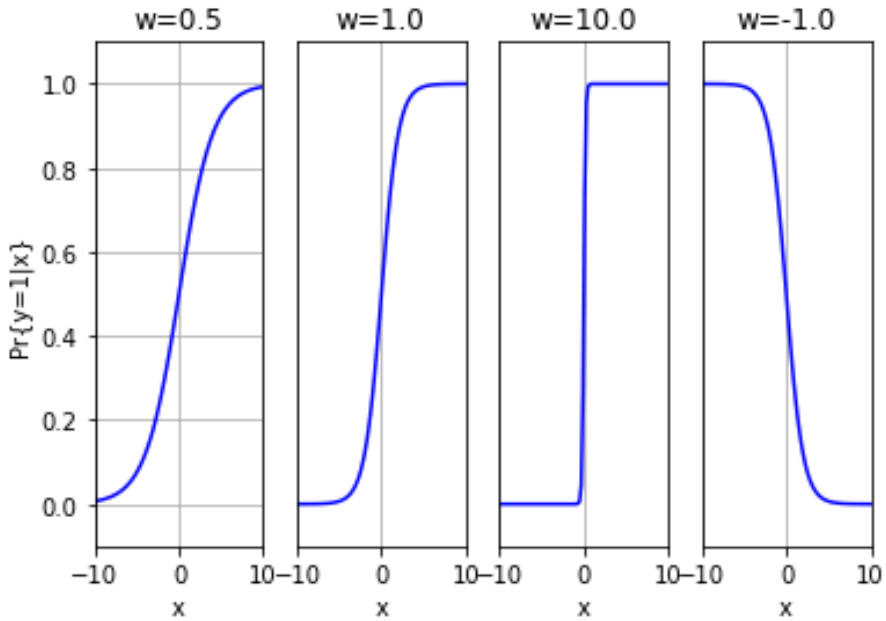
$$\Pr\{y=1\,|\,x\} = \frac{1}{1+e^{-z}} \text{ for } z = wx$$

  - Sign of w determines increasing vs. decreasing in x
  - Magnitude of w determines sharpness of x-domain transition
  - Transition centered at $x=0$

- For a more general b, we have

$$z = b + wx = w\Big(\frac{b}{w} + x\Big).$$

  Thus transition is centered at $x = -\frac{b}{w}$



$\Pr\{y=1\,|\,x\}$ versus x for various w

Maximum likelihood estimation

- Given training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, we can fit the model parameters $(b, \boldsymbol{w})$ using maximum likelihood (ML) estimation:

ML Estimation

1) Define a likelihood function $p(\boldsymbol{y}|\boldsymbol{X}, \left[\begin{smallmatrix} b \\ \boldsymbol{w} \end{smallmatrix}\right])$ with model parameters $\left[\begin{smallmatrix} b \\ \boldsymbol{w} \end{smallmatrix}\right]$
   - As usual, $\boldsymbol{y} \triangleq [y_1, \ldots, y_n]^{\mathsf{T}}$ and $\boldsymbol{X} \triangleq [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^{\mathsf{T}}$

2) The ML model parameters are
$$\begin{aligned}(b_{\text{ml}}, \boldsymbol{w}_{\text{ml}}) &\triangleq \arg\max_{b,\boldsymbol{w}} p(\boldsymbol{y}|\boldsymbol{X}, \left[\begin{smallmatrix} b \\ \boldsymbol{w} \end{smallmatrix}\right]) \\ &= \arg\max_{b,\boldsymbol{w}} \ln p(\boldsymbol{y}|\boldsymbol{X}, \left[\begin{smallmatrix} b \\ \boldsymbol{w} \end{smallmatrix}\right]) \\ &= \arg\min_{b,\boldsymbol{w}} \left\{ -\ln p(\boldsymbol{y}|\boldsymbol{X}, \left[\begin{smallmatrix} b \\ \boldsymbol{w} \end{smallmatrix}\right]) \right\}\end{aligned}$$

- Recall that we previously applied ML estimation to linear regression:
  - There, the likelihood was $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\beta}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{A}\boldsymbol{\beta}, \sigma^2 \boldsymbol{I})$ with $\boldsymbol{A} = [\boldsymbol{1}\ \boldsymbol{X}]$
  - $\Rightarrow \quad -\ln p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\beta}) = \frac{1}{2\sigma^2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\beta}\|^2 + \text{constant}$
  - $\Rightarrow \quad \boldsymbol{\beta}_{\text{ml}} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\beta}\|^2 = \arg\min_{\boldsymbol{\beta}} \text{RSS}(\boldsymbol{\beta}) = \boldsymbol{\beta}_{\text{ls}}$ . . . the LS fit!
- For logistic regression, we use ML estimation with a different likelihood

ML estimation for logistic regression (LR)

- Logistic regression assumes that y_i depends on $\left[\begin{smallmatrix} b \\ \boldsymbol{w} \end{smallmatrix}\right]$ as follows:

$$\Pr\{y_i=1 \,|\, \boldsymbol{X}, \left[\begin{smallmatrix} b \\ \boldsymbol{w} \end{smallmatrix}\right]\} = \tfrac{e^{z_i}}{1+e^{z_i}}, \quad \Pr\{y_i=-1 \,|\, \boldsymbol{X}, \left[\begin{smallmatrix} b \\ \boldsymbol{w} \end{smallmatrix}\right]\} = \tfrac{1}{1+e^{z_i}}, \quad z_i = b + \boldsymbol{x}_i^{\mathsf{T}}\boldsymbol{w}$$

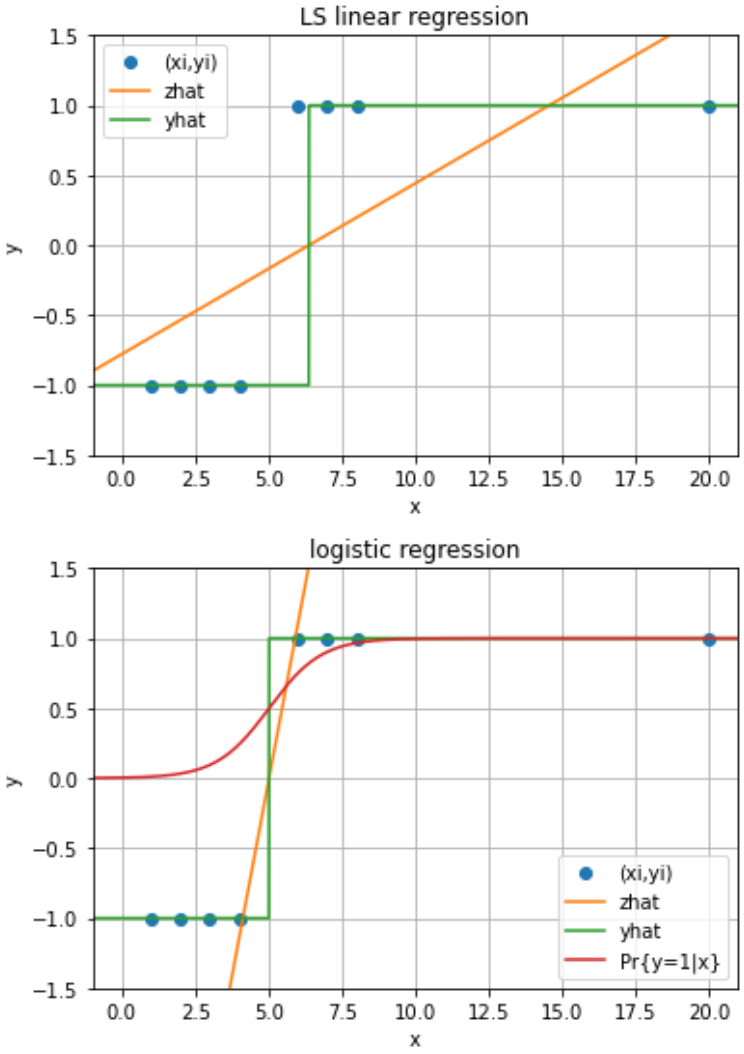
- Thus we have

$$\begin{aligned}
p(\boldsymbol{y}|\boldsymbol{X}, \left[\begin{smallmatrix} b \\ \boldsymbol{w} \end{smallmatrix}\right]) &= \textstyle\prod_{i=1}^n p(y_i|\boldsymbol{X}, \left[\begin{smallmatrix} b \\ \boldsymbol{w} \end{smallmatrix}\right]) && \text{via independence} \\
-\ln p(\boldsymbol{y}|\boldsymbol{X}, \left[\begin{smallmatrix} b \\ \boldsymbol{w} \end{smallmatrix}\right]) &= -\textstyle\sum_{i=1}^n \ln p(y_i|\boldsymbol{X}, \left[\begin{smallmatrix} b \\ \boldsymbol{w} \end{smallmatrix}\right]) && \text{since } \ln(cd) = \ln c + \ln d \\
&= -\textstyle\sum_{i=1}^n \tfrac{y_i+1}{2} \ln \Pr\{y_i=1 \,|\, \boldsymbol{X}, \left[\begin{smallmatrix} b \\ \boldsymbol{w} \end{smallmatrix}\right]\} + (1-\tfrac{y_i+1}{2}) \ln \Pr\{y_i=-1 \,|\, \boldsymbol{X}, \left[\begin{smallmatrix} b \\ \boldsymbol{w} \end{smallmatrix}\right]\} \\
&\qquad \text{since } \tfrac{y_i+1}{2} = \begin{cases} 1 & y_i = 1 \\ 0 & y_i = -1 \end{cases} \quad \text{...switches "on" or "off"} \\
&= -\textstyle\sum_{i=1}^n \tfrac{y_i+1}{2}\big(z_i - \ln[1+e^{z_i}]\big) + (1-\tfrac{y_i+1}{2})\big(0 - \ln[1+e^{z_i}]\big) \\
&= -\textstyle\sum_{i=1}^n \Big(\tfrac{y_i+1}{2} z_i - \tfrac{y_i+1}{2}\ln[1+e^{z_i}] - \ln[1+e^{z_i}] + \tfrac{y_i+1}{2}\ln[1+e^{z_i}]\Big) \\
&= -\textstyle\sum_{i=1}^n \big(\tfrac{y_i+1}{2} z_i - \ln[1+e^{z_i}]\big) && \text{where } z_i = b + \boldsymbol{x}_i^{\mathsf{T}}\boldsymbol{w}
\end{aligned}$$

- The ML estimate of $\left[\begin{smallmatrix} b \\ \boldsymbol{w} \end{smallmatrix}\right]$ is the one minimizing this negative log likelihood

Comparison of LS vs. LR fits on a simple dataset

- Consider ML-fitting of the imbalanced 1D dataset that we saw earlier:
  - The top plot shows the LS regression line $z_{\text{ls}}(x) \triangleq b_{\text{ls}} + w_{\text{ls}}x$, as well as the binary prediction $\widehat{y}_{\text{ls}}(x) = \text{sgn}\left(z_{\text{ls}}(x)\right)$
  - The bottom plot shows the LR score $z_{\text{lr}}(x) \triangleq b_{\text{lr}} + w_{\text{lr}}x$, as well as the binary prediction $\widehat{y}_{\text{lr}}(x) = \text{sgn}\left(z_{\text{lr}}(x)\right)$
  - The plots show that LS is distracted by the outlier sample, while LR is not!
  - The bottom plot also shows $\Pr\{y = 1 \,|\, x\}$ for b_{lr} and w_{lr}. As x increases, LR becomes more confident that $y = 1$



Binary cross-entropy loss

In summary...

- When $y_i \in \{-1, 1\}$, the ML weights for binary logistic regression are

$$(b_{\text{ml}}, \boldsymbol{w}_{\text{ml}}) \triangleq \arg\min_{b,\boldsymbol{w}} \sum_{i=1}^{n} \left(\ln[1 + e^{z_i}] - \tfrac{y_i+1}{2} z_i \right) \quad \text{for} \quad z_i = b + \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{w}$$

  which must be solved numerically

- When $y_i \in \{0, 1\}$, the ML weights for binary logistic regression are

$$(b_{\text{ml}}, \boldsymbol{w}_{\text{ml}}) \triangleq \arg\min_{b,\boldsymbol{w}} \sum_{i=1}^{n} \left(\ln[1 + e^{z_i}] - y_i z_i \right) \quad \text{for} \quad z_i = b + \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{w}$$

  which is a popular alternative expression

- The summation is known as the "logistic loss" or "binary cross-entropy loss"

Adding regularization

Assuming $y_i \in \{0, 1\}$ for the following expressions ...

- Usually, L2 regularization is used with the binary cross-entropy loss:

$$(b_{\text{lr}}, \boldsymbol{w}_{\text{lr}}) = \arg\min_{b,\boldsymbol{w}} \left\{ \sum_{i=1}^{n} \big(\ln[1 + e^{z_i}] - y_i z_i \big) + \alpha \|\boldsymbol{w}\|^2 \right\} \quad \text{for} \quad z_i = b + \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{w}$$

  - When the training data is linearly separable, $\|\boldsymbol{w}_{\text{ml}}\| = \infty$ without regularization! Using L2 regularization with small positive α will keep $\boldsymbol{w}_{\text{lr}}$ finite
  - With correlated features, larger α may be helpful. Tune via cross-val

- Could instead use L1 regularization, and thus perform feature selection:

$$(b_{\text{lr}}, \boldsymbol{w}_{\text{lr}}) = \arg\min_{b,\boldsymbol{w}} \left\{ \sum_{i=1}^{n} \big(\ln[1 + e^{z_i}] - y_i z_i \big) + \alpha \|\boldsymbol{w}\|_1 \right\} \quad \text{for} \quad z_i = b + \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{w}$$

  - As before, use L1 for feature-selection only. Once features are selected, use LR (with L2 & small α). Tune α via cross-val performance of feature-selected LR
  - Computationally, L1 is much more expensive than L2

Logistic regression in sklearn

In sklearn, there is a convenient `LogisticRegression` method:

- Note: L2 regularization is used by default
  - Don't forget to standardize $\boldsymbol{X}$!
- The *inverse* regularization strength is controlled by the parameter `C` > 0
  - So small regularization weight α corresponds to *large* `C`

```
logreg = linear_model.LogisticRegression(C=1e5)
```

```
Xs = preprocessing.scale(X)
logreg.fit(Xs, y)
```

```
LogisticRegression(C=100000.0, class_weight=None, dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=100,
          multi_class='ovr', n_jobs=1, penalty='l2', random_state=None,
          solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
```

```
yhat = logreg.predict(Xs)
acc = np.mean(yhat == y)
print("Accuracy on training data = %f" % acc)
```

```
Accuracy on training data = 0.969253
```

Outline

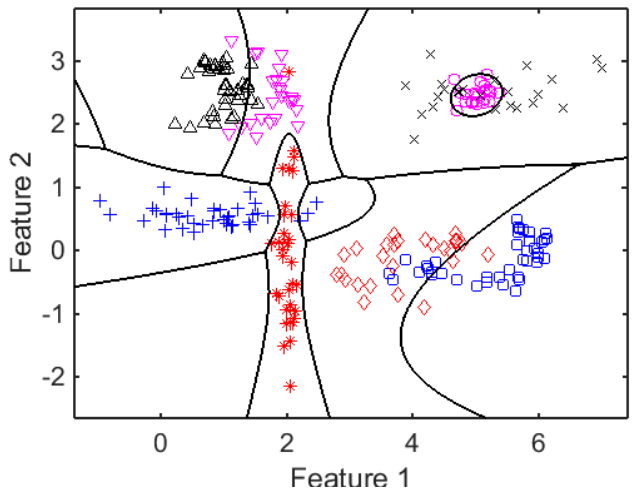
Multiclass classification

- What if there are $K > 2$ classes?
  - Binary classification corresponds to $K = 2$
- Goal: Given training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{1, \ldots, K\}$, we want to classify a test vector $\boldsymbol{x}$ as $y \in \{1, \ldots, K\}$
  - Unlike regression, the target y is categorical
- Mathematically, we want to design a classifier

$$f(\boldsymbol{x}) = \widehat{y} \in \{1, \ldots, K\}$$

such that $\widehat{y} = y$ with high probability

example where $K = 10$:



- Again, $f(\cdot)$ is defined by its decision regions:

$$\mathcal{R}_k \triangleq \{\boldsymbol{x} : f(\boldsymbol{x}) = k\} \quad \text{for} \quad k = 1, \ldots, K$$

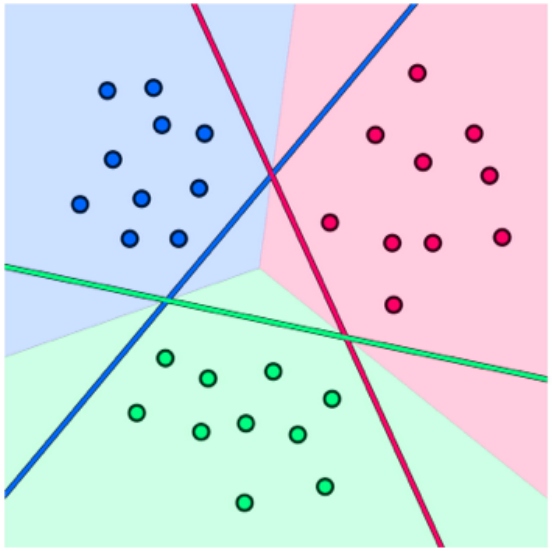
- Important: Classification problems have categorical targets, while regression problems have ordinal targets
  - If the target is discrete, but ordinal, it's probably a regression problem

Multiclass classification using binary classifiers

Multiclass classification is difficult. Can we tackle it using *binary* methods? Yes!

Two main approaches:

- 1-vs-rest (also called 1-vs-all)
  - for each class $k = 1...K$, compute the score z_k that $\boldsymbol{x}$ is in-k versus not-in-k
  - choose the class with highest score: $\widehat{k} = \arg\max_k z_k$
  - most common approach
- 1-vs-1:
  - for each class pair $(k,l)|_{l \neq k}$, compute the score z_{kl} that $\boldsymbol{x}$ is in-k versus in-l
  - choose the class with highest total score: $\widehat{k} = \arg\max_k \sum_{l \neq k} z_{kl}$
  - less common



Outline

Multinomial logistic regression (MLR)

Multi-class extensions of binary logistic regression based on "1-vs-rest":

- Test features $\boldsymbol{x}$ are classified using a two-step approach:
  - For each class $k = 1, \ldots, K$, compute a linear score $z_k = b_k + \sum_{j=1}^{d} x_j w_{kj}$ that indicates whether $\boldsymbol{x}$ is in-k or not-in-k,
  - Choose the winning class hypothesis as: $\widehat{y} = \arg\max_k z_k$
- Two options to design the coefs $\{(b_k, \boldsymbol{w}_k)\}_{k=1}^{K}$, where $\boldsymbol{w}_k \triangleq [w_{k1}, ..., w_{kj}]^{\mathsf{T}}$:
  1. train K separate binary logistic regression models, or
  2. maximize a likelihood function constructed from the "softmax" model:

$$\Pr\{y=k \,|\, \boldsymbol{z}\} = \frac{e^{z_k}}{\sum_{l=1}^{K} e^{z_l}} \quad \text{(denominator ensures } \sum_{k=1}^{K} \Pr\{y=k \,|\, \boldsymbol{z}\} = 1 \;\forall \boldsymbol{z})$$

It's called "softmax" because, if $z_{k_{\max}} \gg z_k$ for all $k \neq k_{\max}$, then

$$\frac{e^{z_k}}{\sum_{l=1}^{K} e^{z_l}} \approx \begin{cases} 1 & \text{if } k = k_{\max} \\ 0 & \text{if } k \neq k_{\max} \end{cases} = \text{ one-hot coding of } k_{\max}$$

One-hot label-coding for MLR

- Goal: Design $\boldsymbol{b} = [b_1, \ldots, b_K]^{\mathsf{T}}$ & $\boldsymbol{W} \triangleq [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K]$ using ML estimation:

$$(\boldsymbol{b}_{\mathsf{ml}}, \boldsymbol{W}_{\mathsf{ml}}) \triangleq \arg\max_{\boldsymbol{b},\boldsymbol{W}} p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{b}, \boldsymbol{W}) = \arg\min_{\boldsymbol{b},\boldsymbol{W}} \left\{ -\sum_{i=1}^{n} \ln p(y_i|\boldsymbol{X}, \boldsymbol{b}, \boldsymbol{W}) \right\}$$

- To help formulate this, we turn the categorical label $y_i \in \{1, \ldots, K\}$ into a binary vector $\boldsymbol{y}_i \triangleq [y_{i1}, \ldots, y_{iK}]^{\mathsf{T}}$ using one-hot-coding, i.e.,

$$\boldsymbol{y}_i = [0 \ldots 0, \underset{\uparrow\, y_i\text{th entry}}{1}, 0 \ldots 0]^{\mathsf{T}} \quad \Leftrightarrow \quad y_{ik} = \begin{cases} 1 & \text{if } k = y_i \\ 0 & \text{if } k \neq y_i \end{cases} \quad \text{for all } i = 1, \ldots, n$$

- Then, similar to the binary $y_i \in \{0, 1\}$ case, we can write

$$\begin{aligned} -\ln p(y_i|\boldsymbol{X}, \boldsymbol{b}, \boldsymbol{W}) &= -\textstyle\sum_{k=1}^{K} y_{ik} \ln \Pr\{y_i = k \,|\, \boldsymbol{X}, \boldsymbol{b}, \boldsymbol{W}\} \\ &= -\textstyle\sum_{k=1}^{K} y_{ik} \ln \frac{e^{z_{ik}}}{\sum_l e^{z_{il}}} \quad \text{with } z_{ik} = b_k + \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{w}_k \\ &= -\textstyle\sum_{k=1}^{K} y_{ik} \big(z_{ik} - \ln \big[\sum_{l=1}^{K} e^{z_{il}} \big] \big) \\ &= \ln \big[\textstyle\sum_{l=1}^{K} e^{z_{il}} \big] - \textstyle\sum_{k=1}^{K} y_{ik} z_{ik}, \quad \text{since } \sum_{k=1}^{K} y_{ik} = 1 \ \forall i \end{aligned}$$

Multinomial cross-entropy loss

- Combining the results from the previous page, we get

$$(\boldsymbol{b}_{\text{ml}}, \boldsymbol{W}_{\text{ml}}) = \arg\min_{\boldsymbol{b},\boldsymbol{W}} \underbrace{\left\{ \sum_{i=1}^{n} \left(\ln \left[\sum_{k=1}^{K} e^{z_{ik}} \right] - \sum_{k=1}^{K} y_{ik} z_{ik} \right) \right\}}_{\triangleq J_{\text{mlr}}(\boldsymbol{b}, \boldsymbol{W})}, \quad z_{ik} = b_k + \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{w}_k$$

  where $J_{\text{mlr}}(\boldsymbol{b}, \boldsymbol{W})$ is known as the cross-entropy loss

- In practice, we add $\alpha \|\boldsymbol{W}\|_F^2$ (L2 regularization) or $\alpha \|\boldsymbol{W}\|_1$ (L1 regularization), and tune α via cross-validation

- With or without regularization, there is no closed-form expression for the optimal $(\boldsymbol{b}, \boldsymbol{W})$, but the optimization problems are convex and can be solved numerically. (More details given in Unit 6)

Logistic Regression: Multinomial vs. Binary-OVR

- The `LogisticRegression` method in `sklearn` handles both binary and multiclass classification. It has many options:

```
LogisticRegression(C=100000.0, class_weight=None, dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=100,
          multi_class='ovr', n_jobs=1, penalty='l2', random_state=None,
          solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
```

- With multiple classes, it is important to understand the `multi_class` option:
  - `multi_class='multinomial'`: This jointly trains $(\boldsymbol{b}, \boldsymbol{W})$ according to the MLR approach described in the last few pages
  - `multi_class='ovr'`: For each $k = 1...K$, this separately trains a one-vs-rest binary classifier $(b_k, \boldsymbol{w}_k)$ using binary LR (i.e., option #1 on page 27)

  The MLR version usually has better performance, but takes longer to train

Outline

Performance metrics for binary classification

- In binary classification, there are 2 types of error:
  - False Positive (or false alarm)
  - False Negative (or missed detection)

contingency table or confusion matrix

	$\widehat{y}=1$	$\widehat{y}=0$
$y=1$	TP	FN
$y=0$	FP	TN

- The implications of these errors can be very different!
  - e.g., consider breast cancer diagnosis

- Common machine-learning performance metrics:
  - accuracy: $\Pr\{\widehat{y}=y\} = \frac{\mathsf{TP}+\mathsf{TN}}{\mathsf{TP}+\mathsf{FN}+\mathsf{TN}+\mathsf{FP}}$ — how often is the test correct?
  - precision: $\Pr\{y=1 \,|\, \widehat{y}=1\} = \frac{\mathsf{TP}}{\mathsf{TP}+\mathsf{FP}}$ — given a positive test, how often is the patient actually sick?
  - recall: $\Pr\{\widehat{y}=1 \,|\, y=1\} = \frac{\mathsf{TP}}{\mathsf{TP}+\mathsf{FN}}$ — given a sick patient, how often is the test correct?
  - F1-score: $[\frac{1}{2}(\frac{1}{\mathsf{precision}} + \frac{1}{\mathsf{recall}})]^{-1}$ — harmonic mean of precision & recall

- Common metrics in medicine:
  - sensitivity: $\Pr\{\widehat{y}=1 \,|\, y=1\} = \frac{\mathsf{TP}}{\mathsf{TP}+\mathsf{FN}}$ — given a sick patient, how often is the test correct?
  - specificity: $\Pr\{\widehat{y}=0 \,|\, y=0\} = \frac{\mathsf{TN}}{\mathsf{TN}+\mathsf{FP}}$ — given a healthy patient, how often is the test correct?

https://en.wikipedia.org/wiki/Evaluation_of_binary_classifiers

Breast cancer demo

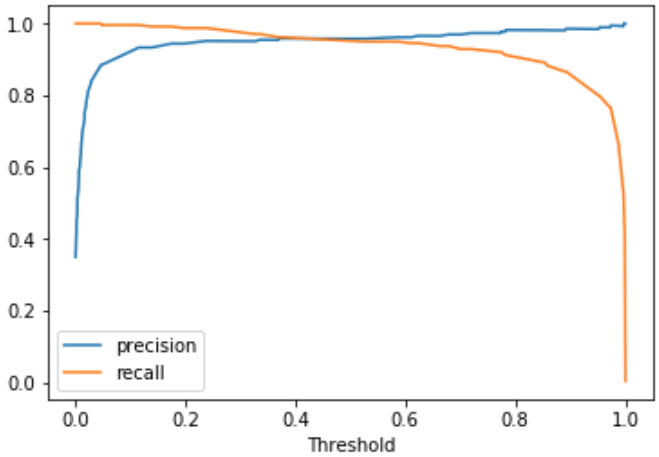
- We now assess classification performance on the breast cancer demo
- Use 10-fold cross-validation
- sklearn includes support for computing precision, recall, F1 score, and accuracy (as defined on previous page)

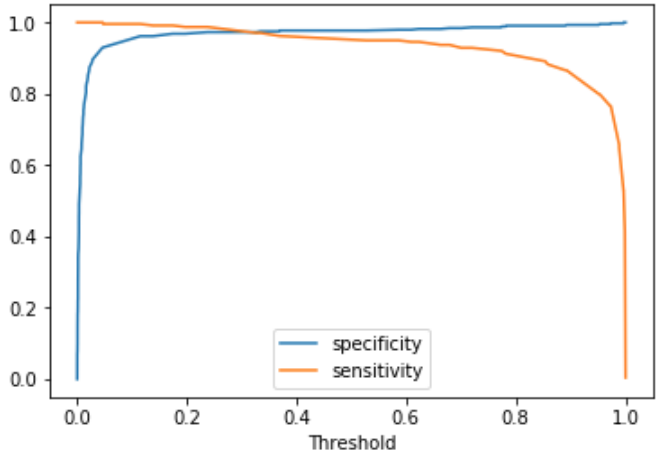
```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_validate

# Instantiate KFold object
nfold = 10
kf = KFold(n_splits=nfold,shuffle=True,random_state=0)

# Do cross-validation
scores = ('precision', 'recall', 'f1', 'accuracy')
cv_results = cross_validate(logreg, Xs, y, cv=kf, scoring=scores)

# Extract test metrics
prec = cv_results['test_precision']
rec = cv_results['test_recall']
f1 = cv_results['test_f1']
acc = cv_results['test_accuracy']

# Take average values of the metrics
precm = np.mean(prec)
recm = np.mean(rec)
f1m = np.mean(f1)
accm= np.mean(acc)

# Compute the standard errors
prec_se = np.std(prec,ddof=1)/np.sqrt(nfold)
rec_se = np.std(rec,ddof=1)/np.sqrt(nfold)
f1_se = np.std(f1,ddof=1)/np.sqrt(nfold)
acc_se = np.std(acc,ddof=1)/np.sqrt(nfold)

print('Precision = {0:.4f}, SE={1:.4f}'.format(precm,prec_se))
print('Recall =    {0:.4f}, SE={1:.4f}'.format(recm, rec_se))
print('f1 =        {0:.4f}, SE={1:.4f}'.format(f1m, f1_se))
print('Accuracy =  {0:.4f}, SE={1:.4f}'.format(accm, acc_se))
```

```
Precision = 0.9432, SE=0.0107
Recall =    0.9612, SE=0.0081
f1 =        0.9518, SE=0.0078
Accuracy =  0.9556, SE=0.0067
```

Making hard decisions

- logistic regression gives test score $z = b + \boldsymbol{x}^{\mathsf{T}}\boldsymbol{w}$, which yields confidence $\Pr\{y=1 \,|\, z\} = \frac{1}{1+e^{-z}}$
- Can convert to a hard decision by thresholding:
  - Set $\widehat{y} = 1$ when $\Pr\{y=1 \,|\, z\} > t$ for threshold t
- $t = 0.5$ minimizes the error rate, $\Pr\{\widehat{y} \neq y\}$
  - i.e., maximizes accuracy
- $t \in [0, 1]$ trades precision for recall
  - precision: $\Pr\{y=1 \,|\, \widehat{y}=1\}$
  - recall: $\Pr\{\widehat{y}=1 \,|\, y=1\}$
- $t \in [0, 1]$ trades sensitivity for specificity
  - sensitivity: $\Pr\{\widehat{y}=1 \,|\, y=1\}$
  - specificity: $\Pr\{\widehat{y}=0 \,|\, y=0\}$

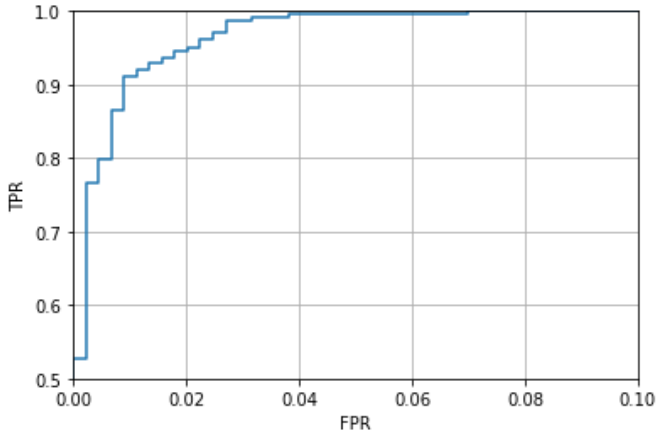


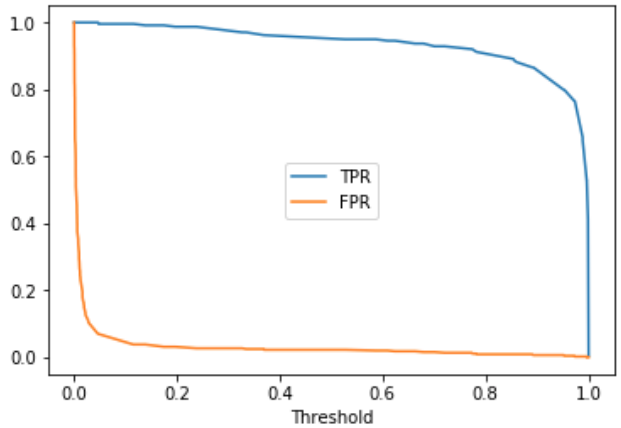


The ROC curve and the AUC

- The receiver operating characteristic (ROC) is the plot of TPR versus FPR
  - TPR: $\Pr\{\widehat{y}=1 \,|\, y=1\}$
  - FPR: $\Pr\{\widehat{y}=1 \,|\, y=0\}$

  where each depends on the decision threshold t
- Thus, t controls the location on the ROC curve
- The area under the curve (AUC) is a threshold-independent performance metric






```
from sklearn import metrics
yprob = logreg.predict_proba(Xs)
fpr, tpr, thresholds = metrics.roc_curve(y,yprob[:,1])
```

```
auc=metrics.roc_auc_score(y,yprob[:,1])
print("AUC=%f" % auc)
```

```
AUC=0.996344
```

Performance metrics for multiclass classification

- In multiclass classification, there are many possible error types
- If the rows of the confusion matrix are normalized to sum-to-one ...
  - (k,l)th entry becomes $\Pr\{\widehat{y}=l \,|\, y=k\}$
  - diagonal terms show per-class accuracy, $\Pr\{\widehat{y}=k \,|\, y=k\}$

contingency table or confusion matrix

	$\widehat{y}=1$	$\widehat{y}=2$	$\cdots$	$\widehat{y}=K$
$y=1$	10	3	$\cdots$	4
$y=2$	1	14	$\cdots$	3
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$y=K$	4	2	$\cdots$	7

- The overall accuracy can be computed by averaging the per-class accuracies:

$$\Pr\{\widehat{y}=y\} = \sum_{k=1}^{K} \Pr\{\widehat{y}=k \,|\, y=k\} \Pr\{y=k\}$$

- Using micro-averaging of k-vs-rest metrics, can generalize precision & recall:

$$\text{precision} = \frac{\sum_k \text{TP}_k}{\sum_k \text{TP}_k + \text{FP}_k} \;\&\; \text{recall} = \frac{\sum_k \text{TP}_k}{\sum_k \text{TP}_k + \text{FN}_k}$$

and generalize ROC (i.e., the plot of TPR vs FPR) to multiclass data:

$$\text{TPR} = \frac{\sum_k \text{TP}_k}{\sum_k \text{TP}_k + \text{FN}_k} \;\&\; \text{FPR} = \frac{\sum_k \text{FP}_k}{\sum_k \text{FP}_k + \text{TN}_k}$$

Imbalanced class labels

- Class imbalance refers to the case where some classes have many fewer samples than others, i.e., $\Pr\{y=k\} \ll \frac{1}{K}$ for one or more k
- In this case, accuracy is not a useful performance metric
  - It's possible to obtain high total accuracy $\Pr\{\widehat{y}=y\}$ even when per-class accuracy $\Pr\{\widehat{y}=k \,|\, y=k\}$ is very low for some k! This can be seen from
    $$\Pr\{\widehat{y}=y\} = \textstyle\sum_{k=1}^{K} \Pr\{\widehat{y}=k \,|\, y=k\} \Pr\{y=k\}$$
  - Instead, use balanced accuracy $\sum_{k=1}^{K} \Pr\{\widehat{y}=k \,|\, y=k\}$, or AUC-PRC
- Similarly, when training via cross-entropy loss, the minority classes are under-represented because they account for relatively few samples i in $\sum_i \text{loss}_i$
  - Remedy: train using class_weight='balanced' (read more here)
- Finally, when doing K-fold CV, use the stratified variant, which ensures that all folds have a similar class balance

Learning objectives

- Understand classification problems in machine learning:
  - Identify features, labels; binary vs multiclass; linear vs nonlinear
  - visualize scatterplots and decision regions
- For binary classification problems, understand . . .
  - linear classifiers, separating hyperplanes, linearly separable data
  - effect of feature transformation
  - why LS linear regression doesn't work well
  - logistic regression: logistic function, cross-entropy loss, ML fitting, regularization
  - common error metrics: accuracy, precision, recall, F1
  - the effect of the decision threshold, ROC, AUC
- For multiclass classification problems, understand . . .
  - solutions that use multiple binary classifiers
  - multinomial logistic regression: softmax function, cross-entropy loss, ML fitting
- How to implement and assess classification using sklearn