Unit 5
Linear Classification & Logistic Regression

Prof. Phil Schniter

The Ohio State University

ECE 5300: Introduction to Machine Learning, Sp21

Learning objectives

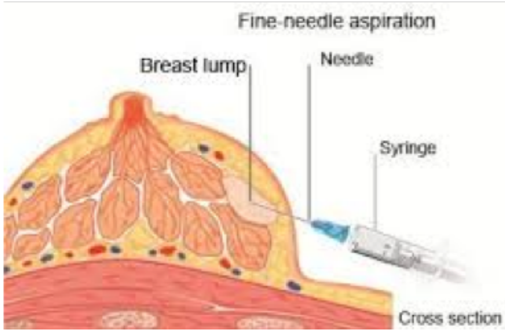
- Understand classification problems in machine learning:
  - Identify features, labels; binary vs multiclass; linear vs nonlinear
  - visualize scatterplots and decision regions
- For binary classification problems, understand ...
  - linear classifiers, separating hyperplanes, linearly separable data
  - 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

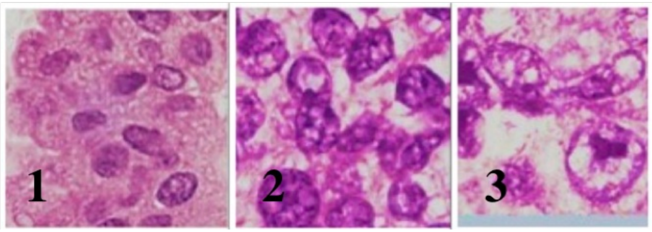
Outline

- Motivating Example: Diagnosing Breast Cancer
- Binary Classification
- Binary Logistic Regression
- Multiclass Classification
- Multinomial Logistic Regression
- Measuring Accuracy in Classification

Diagnosing Breast Cancer

- Fine-needle aspiration of suspicious breast lumps:
  - Tissue is stained & viewed under microscope
  - Cytopathologist visually inspects cells
  - Tries to classify as benign or malignant
  - If malignant, also provides grading
- Would like to improve accuracy
- 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

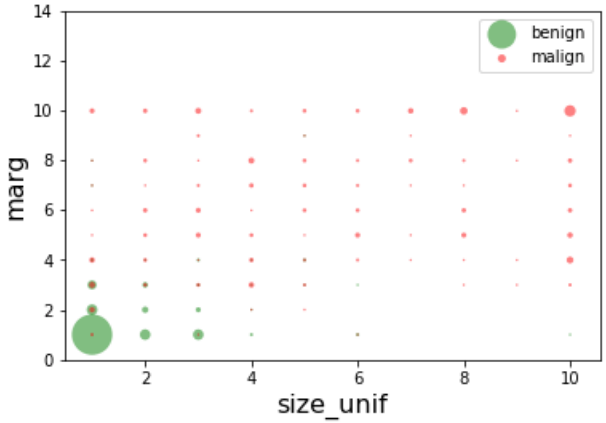
Explanation:
https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.names

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 $\boldsymbol{x} = [x_1, x_2]$?

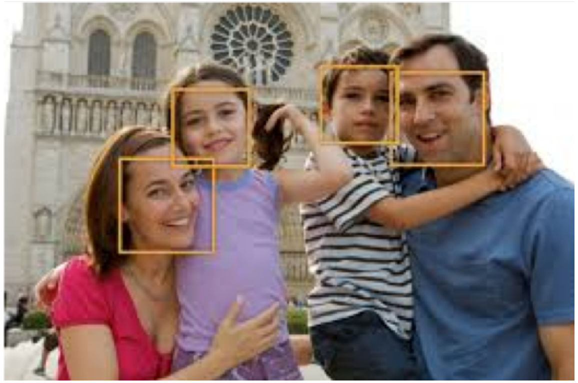
Outline

- Motivating Example: Diagnosing Breast Cancer
- Binary Classification
- Binary Logistic Regression
- Multiclass Classification
- Multinomial Logistic Regression
- Measuring Accuracy in Classification

Binary classification

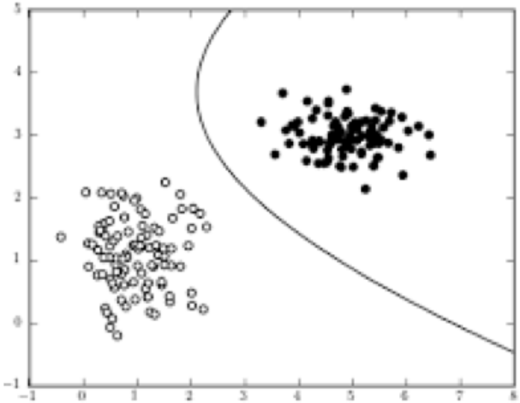
or $\{0,1\}$

- 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
  - Using $\{-1, 1\}$ leads to cleaner notation later



- Many applications:
  - Face detection: is a face present here or not?
  - Are these cells cancerous or not?

Example with $d = 2$:



- Mathematically, want to design a classifier

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

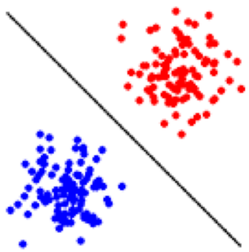
such that $\widehat{y} = y$ with high probability

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

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

Binary linear classification

- One option is binary *linear* classification:
  1) define 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 weights x_j
  2) threshold the score to obtain $\widehat{y} = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$
  3) use weights $\boldsymbol{w} \triangleq [w_1, \dots, w_d]^\mathsf{T}$ and intercept b that are learned from the training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$



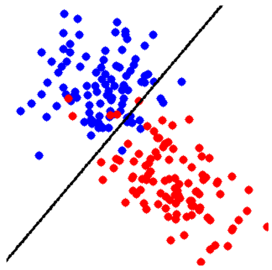
- Note that, at the decision boundary, we have

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

  Thus, the hyperplane

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

  separates the decision regions

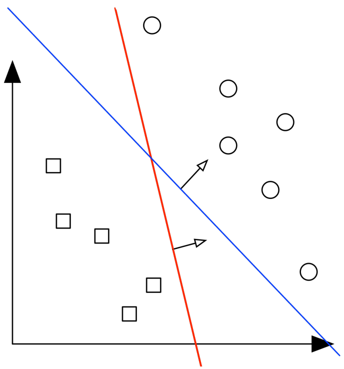


- This decision boundary is linear in $\boldsymbol{x}$ (see figures). When does this perform well?

Linear separability

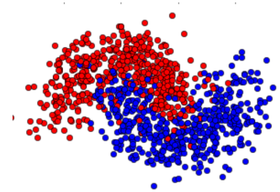
- Linear classification performs well when the data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ is "linearly separable"
- Linearly separable means that there exists a hyperplane
$$\{\boldsymbol{x} : b + \boldsymbol{x}^\mathsf{T}\boldsymbol{w} = 0\}$$
(for some b and $\boldsymbol{w}$) that separates the samples $\boldsymbol{x}_i$ 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 not unique.

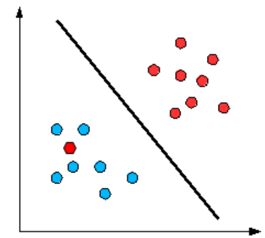
linear separability:

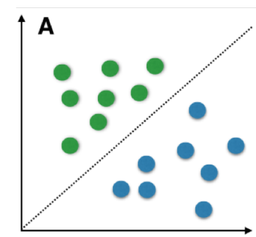


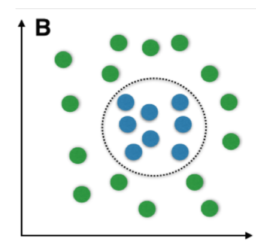
Linear versus nonlinear classification

- 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., shows which features matter)
  - 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 can be used as a building block (e.g., for neural networks, decision trees, etc.)









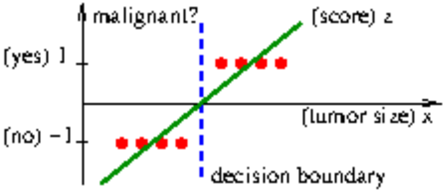
Linear classification vs. LS linear regression

- 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 $\text{RSS} = \left\| \boldsymbol{y} - \boldsymbol{A} \begin{bmatrix} b \\ \boldsymbol{w} \end{bmatrix} \right\|^2$ and then output

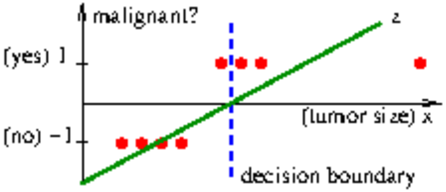
$$\widehat{y} = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}, \quad \text{where } z = [1 \ \boldsymbol{x}]^\mathsf{T} \begin{bmatrix} b_{\mathsf{ls}} \\ \boldsymbol{w}_{\mathsf{ls}} \end{bmatrix}$$

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

balanced $\{x_i\}$:



imbalanced $\{x_i\}$:



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

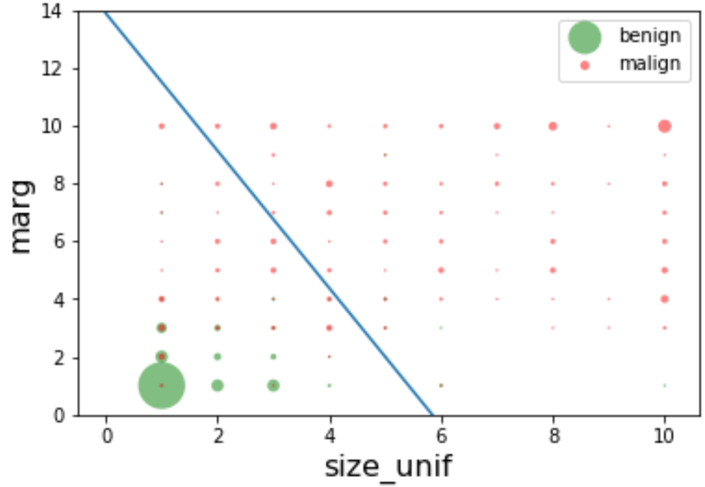
$$\{\boldsymbol{x} : b + \boldsymbol{x}^\mathsf{T}\boldsymbol{w} = 0\}$$

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

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

- 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

- Motivating Example: Diagnosing Breast Cancer
- Binary Classification
- Binary Logistic Regression
- Multiclass Classification
- Multinomial Logistic Regression
- Measuring Accuracy in Classification

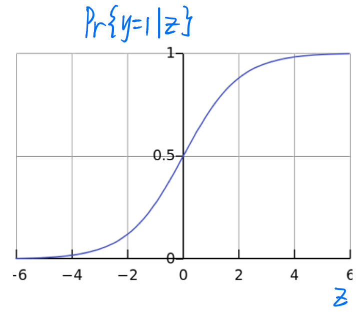
Binary logistic regression

- 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

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

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 "sigmoid function"



Understanding the logistic model

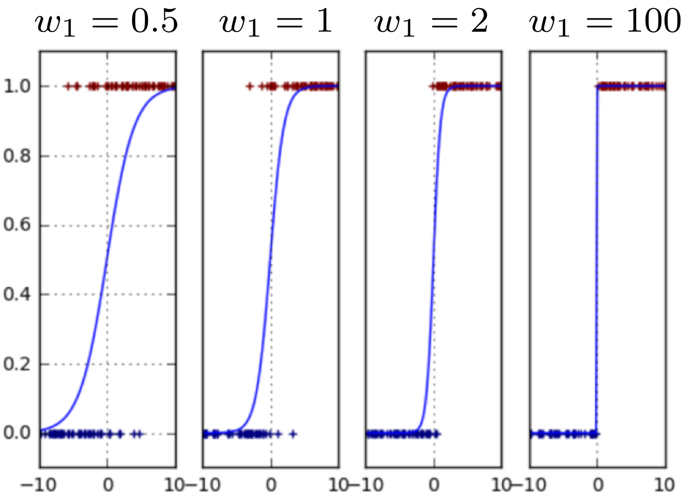
- Previously considered $\Pr\{y=1 \mid z\}$. What about $\Pr\{y=1 \mid \boldsymbol{x}\}$?
- Consider the simple case of a single scalar feature x and no intercept b:

$$\Pr\{y=1 \mid x\} = \frac{1}{1+e^{-z}} \text{ for } z = w_1 x$$

  - As the weight w_1 increases, the x-domain transition sharpens
  - Equivalently, as w_1 increases, there is less randomness in y (for a given x)
- If we add the intercept term b, then

$$z = b + w_1 x = w_1\left(\frac{b}{w_1} + x\right)$$

and the transition occurs at $x = -\frac{b}{w_1}$



curve: $\Pr\{y=1 \mid x\}$ versus x
dots: random $\{(x_i, y_i)\}_{i=1}^n$

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 $p(\boldsymbol{y}|b, \boldsymbol{w})$ with model parameters $(b, \boldsymbol{w})$
   - As usual, $\boldsymbol{y} \triangleq [y_1, \ldots, y_n]^\mathsf{T}$. Also, $\boldsymbol{X} \triangleq [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^\mathsf{T}$ is embedded in p

2) The ML model parameters are
$$\begin{aligned}(b_{\mathsf{ml}}, \boldsymbol{w}_{\mathsf{ml}}) &\triangleq \arg\max\nolimits_{b,\boldsymbol{w}} p(\boldsymbol{y}|b, \boldsymbol{w}) \\ &= \arg\max\nolimits_{b,\boldsymbol{w}} \ln p(\boldsymbol{y}|b, \boldsymbol{w}) \\ &= \arg\min\nolimits_{b,\boldsymbol{w}} \big\{ -\ln p(\boldsymbol{y}|b, \boldsymbol{w}) \big\}\end{aligned}$$

- Recall that we previously applied ML estimation to linear regression:
  - There, the likelihood was $p(\boldsymbol{y}|\boldsymbol{\beta}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{A}\boldsymbol{\beta}, \sigma_\epsilon^2 \boldsymbol{I})$ with $\boldsymbol{A} = [\boldsymbol{1} \;\; \boldsymbol{X}]$
  - thus $-\ln p(\boldsymbol{y}|\boldsymbol{\beta}) = \frac{1}{2\sigma^2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\beta}\|^2 +$ constant
  - and so $\boldsymbol{\beta}_{\mathsf{ml}} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\beta}\|^2 = \arg\min_{\boldsymbol{\beta}} \mathrm{RSS}(\boldsymbol{\beta}) = \boldsymbol{\beta}_{\mathsf{ls}}$

- Both logistic regression, we use ML estimation with an appropriate likelihood!

ML estimation for logistic regression

- Logistic regression assumes that y_i depends on $(b, \boldsymbol{w})$ as follows:

$$\Pr\{y_i = 1 \,|\, b, \boldsymbol{w}\} = \tfrac{e^{z_i}}{1+e^{z_i}}, \quad \Pr\{y_i = -1 \,|\, b, \boldsymbol{w}\} = \tfrac{1}{1+e^{z_i}}, \quad z_i = b + \boldsymbol{x}_i^\mathsf{T} \boldsymbol{w}$$

- Thus we have

$$\begin{aligned}
p(\boldsymbol{y}|b,\boldsymbol{w}) &= \textstyle\prod_{i=1}^n p(y_i|b,\boldsymbol{w}) && \text{via independence assumption} \\
-\ln p(\boldsymbol{y}|b,\boldsymbol{w}) &= -\textstyle\sum_{i=1}^n \ln p(y_i|b,\boldsymbol{w}) && \text{since } \ln(ab) = \ln a + \ln b \\
&= -\textstyle\sum_{i=1}^n \tfrac{y_i+1}{2} \ln \Pr\{y_i = 1 \,|\, \boldsymbol{x}_i; b, \boldsymbol{w}\} + (1 - \tfrac{y_i+1}{2}) \ln \Pr\{y_i = -1 \,|\, \boldsymbol{x}_i; b, \boldsymbol{w}\} \\
& \qquad \text{since } \tfrac{y_i+1}{2} = \begin{cases} 1 & y_i = 1 \\ 0 & y_i = -1 \end{cases} \\
&= -\textstyle\sum_{i=1}^n \tfrac{y_i+1}{2} \big(z_i - \ln[1+e^{z_i}]\big) + (1 - \tfrac{y_i+1}{2}) \big(0 - \ln[1+e^{z_i}]\big) \\
&= -\textstyle\sum_{i=1}^n \Big(\tfrac{y_i+1}{2} z_i - \tfrac{y_i+1}{2} \ln[1+e^{z_i}] - \ln[1+e^{z_i}] + \tfrac{y_i+1}{2} \ln[1+e^{z_i}]\Big) \\
&= -\textstyle\sum_{i=1}^n \big(\tfrac{y_i+1}{2} z_i - \ln[1+e^{z_i}]\big) && \text{where } z_i = b + \boldsymbol{x}_i^\mathsf{T} \boldsymbol{w}
\end{aligned}$$

- The ML estimates of $(b, \boldsymbol{w})$ are the ones minimizing this negative log likelihood

Binary cross-entropy loss

In summary...

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

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

  which must be solved numerically

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

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

  which is a common 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 cross-entropy loss:

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

  - Why? When the training data is linearly separable, the cross-entropy loss decreases as $|w_j|$ increases, and so $\|\boldsymbol{w}_{\mathsf{ml}}\| = \infty$. L2 regularization keeps $\boldsymbol{w}_{\mathsf{lr}}$ finite

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

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

- With L2 regularization, could use small (but positive) α to avoid bias
- Or, with L1 or L2 regularization, could tune α using K-fold cross-validation

Logistic regression in sklearn

In sklearn, there is a nice `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 $\texttt{C} > 0$
  - So use *large* `C` to avoid regularization-induced bias

$C = \frac{1}{\alpha}$

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

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

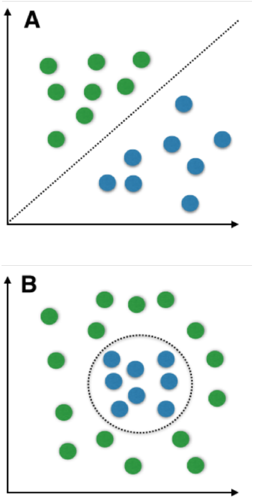
```
LogisticRegression(C=100000.0, class_weight=None, dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=100,
          multi_class='ovr', n_jobs=1, penalty='l2', random_state=None,
          solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
```

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

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

Feature transformations

- Can we implement *curved* boundaries with *linear* classification methods like logistic regression?
- Yes! Through feature transformation ...
  - Example: given raw features (x_1, x_2), we could transform to $\boldsymbol{x} \triangleq [x_1, x_2, x_1^2, x_2^2]^\mathsf{T}$
  - Then the "linear" score $z = b + \boldsymbol{x}^\mathsf{T}\boldsymbol{w}$ would be *quadratic* in the raw features (x_1, x_2)
- One-hot coding is another important feature transformation. We saw how it can be used for categorical features like $x_j \in \{\text{Ford}, \text{BMW}, \text{GM}\}$



Outline

- Motivating Example: Diagnosing Breast Cancer
- Binary Classification
- Binary Logistic Regression
- Multiclass Classification
- Multinomial Logistic Regression
- Measuring Accuracy in Classification

Multiclass classification

- What if there are $K \geq 2$ classes?
  - Binary classification is the special case $K = 2$
- 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

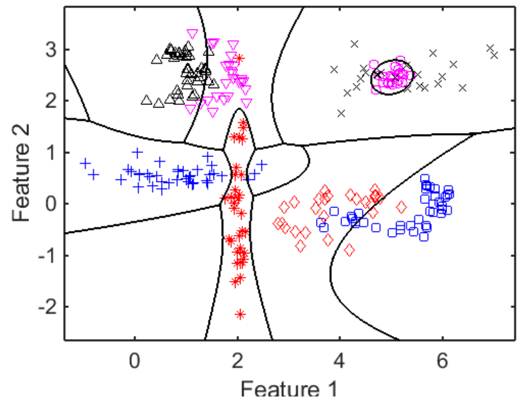
$$f(\boldsymbol{x}) = \widehat{y} \in \{1, \ldots, K\}$$

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

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

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

example where $K = 10$:



- 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

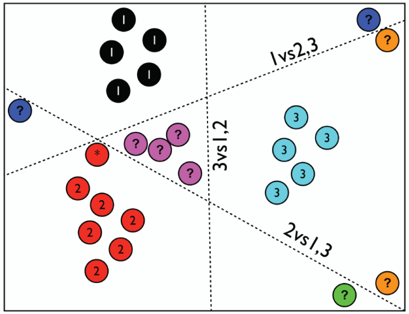
Multiclass classification is difficult. Can we tackle it using *binary* methods? Yes!

Two main approaches:

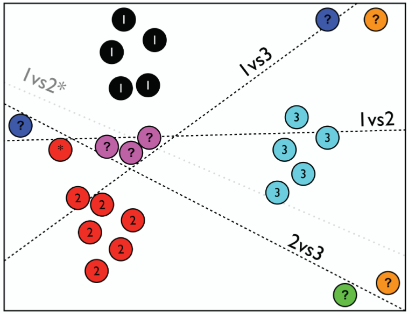
- Voting on binary outcomes
  - 1-vs-rest: For each class k, decide if $\boldsymbol{x}$ is in-k vs not-in-k, then choose the k with most votes
  - 1-vs-1: For each pair (k, l), decide if $\boldsymbol{x}$ is in-k vs in-l, then choose k with most votes

  Problem: Tied votes lead to ambiguities! (see right)
- Choosing the highest confidence value
  - 1-vs-rest: For each class k, compute confidence that $\boldsymbol{x}$ is in-k vs not-in-k, then choose the k with highest confidence

  Ambiguities avoided by continuous-valued confidence!
  We'll study one example (MLR) in more detail ...



(b) 1-vs-All



(a) 1-vs-1

Outline

Multinomial logistic regression (MLR)

Suppose we combine multiple binary classifiers using the "1-vs-rest confidence-value" approach (p. 25). With binary *linear* classifiers, we would do:

- For each class $k = 1, \ldots, K$,
  - given b_k and $\boldsymbol{w}_k = [w_{k1}, \ldots, w_{kd}]^\mathsf{T}$ that classify in-k or not-in-k
  - compute a linear score on k from $\boldsymbol{x}$, i.e., $z_k = b_k + \sum_{j=1}^d x_j w_{kj}$

  then choose the winner via: $\widehat{y} = \arg\max_k \underbrace{\Pr\{y{=}k \,|\, \boldsymbol{z}\}}_{\text{confidence value}}$

- In multinomial logistic regression, we adopt the "softmax" likelihood function:

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

probability mass function

- The softmax has the property that, if $z_{k_{\max}} \gg z_k$ for all $k \neq k_{\max}$, then

$$\frac{e^{z_k}}{\sum_{l=1}^K e^{z_l}} \approx \begin{cases} 1 & \text{if } k = k_{\max} \\ 0 & \text{if } k \neq k_{\max} \end{cases} \quad \text{so it's a "soft" version of "}\delta_{k-k_{\max}}\text{"}$$

One-hot label-coding for MLR

- We design $\boldsymbol{b} = [b_1, \ldots, b_K]^\mathsf{T}$ & $\boldsymbol{W} \triangleq [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K]$ using ML estimation, i.e.,

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

- 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.,

$$y_{ik} = \begin{cases} 1 & \text{if } y_i = k \\ 0 & \text{if } y_i \neq k \end{cases} \quad \text{for all } i = 1, \ldots, n$$

$y_i = k$

$\underline{y}_i = (0\ 0\ 1\ 0\ 0)$

↑ k location

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

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

Multinomial cross-entropy loss

- Combining the results from the previous page, we get

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

  where $J_{\mathsf{lr}}(\boldsymbol{b}, \boldsymbol{W})$ 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

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 designs $(\boldsymbol{b}, \boldsymbol{W})$ according to the MLR approach described in the last few pages
  - `multi_class=ovr`: For each $k = 1...K$, this separately designs a one-vs-rest binary classifier $(b_k, \boldsymbol{w}_k)$

  The MLR version usually has slightly better performance but at the cost of significantly higher complexity

Outline

- Motivating Example: Diagnosing Breast Cancer
- Binary Classification
- Binary Logistic Regression
- Multiclass Classification
- Multinomial Logistic Regression
- Measuring Accuracy in Classification

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

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

- The implications of these errors can be very different!
  - e.g., consider breast cancer diagnosis
- Common machine-learning performance metrics:
  - precision: $\Pr\{y=1 \mid \widehat{y}=1\} = \frac{\text{TP}}{\text{TP+FP}}$ — given a positive test, how often is the patient cancerous?
  - recall: $\Pr\{\widehat{y}=1 \mid y=1\} = \frac{\text{TP}}{\text{TP+FN}}$ — given that the patient has cancer, how often is it detected?
  - F1-score: $[\frac{1}{2}(\frac{1}{\text{precision}} + \frac{1}{\text{recall}})]^{-1}$ — harmonic mean of precision & recall
  - accuracy: $\Pr\{\widehat{y} = y\} = \frac{\text{TP+TN}}{\text{TP+FN+TN+FP}}$ — how often is the test correct?
- Common metrics in medicine:
  - sensitivity: $\Pr\{\widehat{y}=1 \mid y=1\} = \frac{\text{TP}}{\text{TP+FN}}$ — given that the patient has cancer, how often is it detected?
  - specificity: $\Pr\{\widehat{y}=0 \mid y=0\} = \frac{\text{TN}}{\text{TN+FP}}$ — given a healthy patient, how often is diagnosis correct?

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

Breast cancer demo

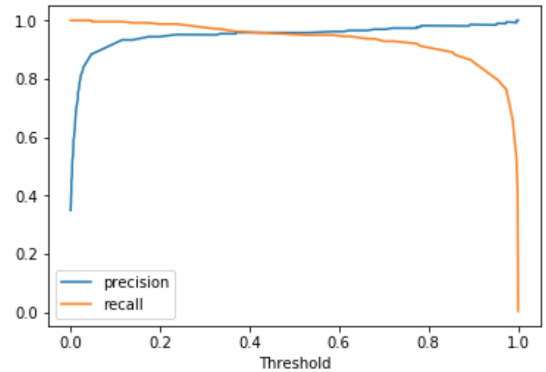
- We now assess classification performance on the breast cancer demo
- Use 10-fold cross-validation
- sklearn includes support for computing precision, recall, F1 score, and accuracy (as defined on previous page)

```
from sklearn.model_selection import KFold
from sklearn.metrics import precision_recall_fscore_support

nfold = 10
kf = KFold(n_splits=nfold,shuffle=True)
prec = []
rec = []
f1 = []
acc = []
for train, test in kf.split(Xs):
    # Get training and test data
    Xtr = Xs[train,:]
    ytr = y[train]
    Xts = Xs[test,:]
    yts = y[test]

    # Fit a model
    logreg.fit(Xtr, ytr)
    yhat = logreg.predict(Xts)

    # Measure performance
    preci,reci,f1i,_= precision_recall_fscore_support(yts,yhat,average='binary')
    prec.append(preci)
    rec.append(reci)
    f1.append(f1i)
    acci = np.mean(yhat == yts)
    acc.append(acci)

# 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)/np.sqrt(nfold-1)
rec_se = np.std(rec)/np.sqrt(nfold-1)
f1_se = np.std(f1)/np.sqrt(nfold-1)
acc_se = np.std(acc)/np.sqrt(nfold-1)
```

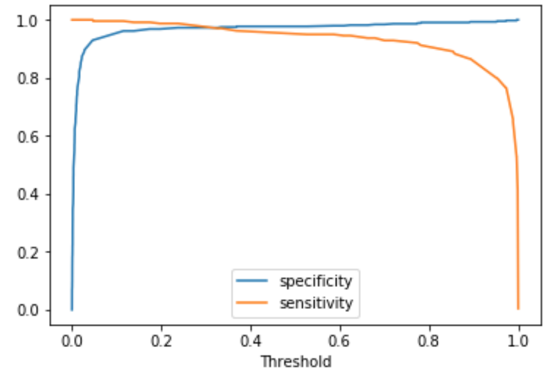
```
Precision = 0.9554, SE=0.0095
Recall =    0.9513, SE=0.0095
f1 =        0.9527, SE=0.0051
Accuracy =  0.9678, SE=0.0037
```

Making hard decisions

- Given test score $z = b + \boldsymbol{x}^\mathsf{T}\boldsymbol{w}$, logistic regression outputs a confidence value, $\Pr\{y=1 \,|\, z\} \in [0,1]$
- Can convert to a hard decision by thresholding:
  - Set $\widehat{y} = 1$ when $\Pr\{y=1 \,|\, z\} > t$ for threshold t
- $t = 1/2$ 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: $\Pr\{\widehat{y}=0 \,|\, y=0\}$



$\longrightarrow t$



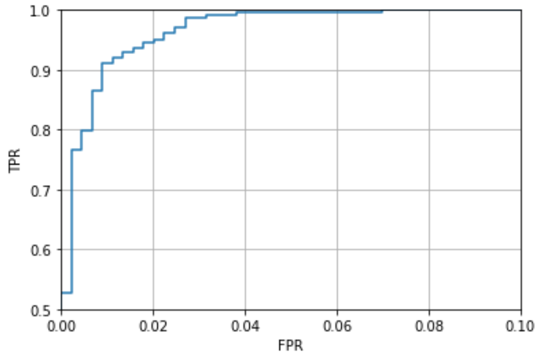
The ROC curve and the AUC

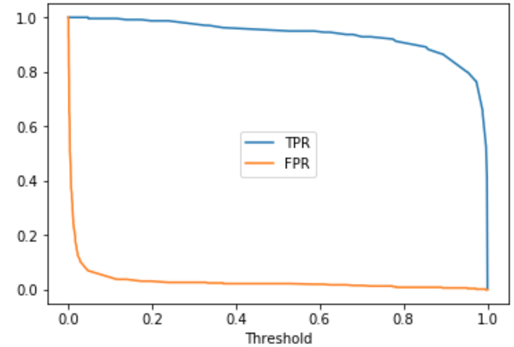
- The receiver operating characteristic (ROC) is the plot of TPR versus FPR
  - FPR: $\Pr\{\widehat{y}=1 \mid y=0\}$
  - TPR: $\Pr\{\widehat{y}=1 \mid y=1\}$
- The threshold t controls location on curve
- The area under the curve (AUC) is a threshold-independent performance metric

```
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

contingency table or **confusion matrix**

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

- If columns of confusion matrix are normalized to sum-to-one ...
  - (k,l)th entry becomes $\Pr\{\widehat{y}=k \,|\, y=l\}$
  - diagonal terms show per-class accuracy, $\Pr\{\widehat{y}=k \,|\, y=k\}$

- The overall accuracy can be computed as

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

- Careful with imbalanced labels: $\Pr\{y=k\} \ll \frac{1}{K}$ for one or more k
  - Accuracy may seem good but actually be meaningless!
  - Strategies: use different metric, resample data, modify classifier, ...
  - Read more here and here and here

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