CS-GY 6923: Lecture 4
Naive Bayes Classifier, Bayesian Linear Regression + Regularization

NYU Tandon School of Engineering, Prof. Christopher Musco

BAYESIAN PROBABILISTIC MODELING

Specific approach to machine learning:

1. Design parametric probabilistic model (usual pretty simple) that plausibly could have generated our data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$.
2. Learn unknown parameters of the model based on observed training data.
3. Given new input $\mathbf{x}$, predict the label y which is most likely under the now fixed probabilistic model.

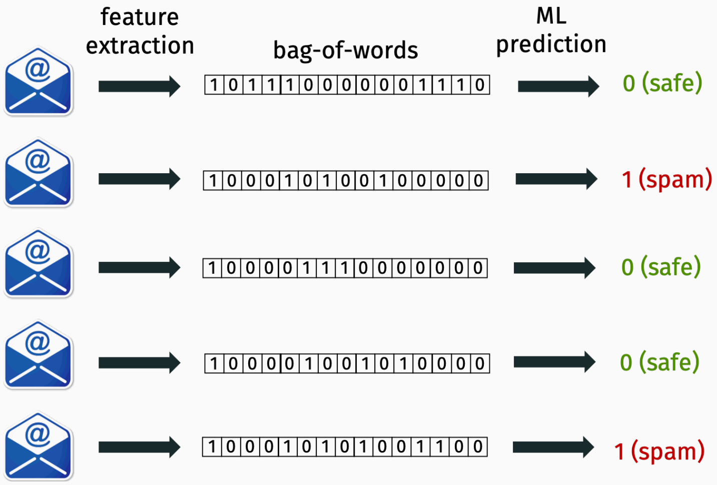
Two strategies covered: Maximum a posteriori (MAP) estimation and maximum likelihood estimation (MLE).

NAIVE BAYES CLASSIFIER

Goal:

- Build a probabilistic model for a binary classification problem.
- Estimate parameters of the model.
- From the model derive a Maximum a posteriori (MAP) estimation classification rule for future predictions. Specifically we will see the Naive Bayes Classifier.

SPAM PREDICTION



Both target labels and data vectors are binary.

PROBABILISTIC MODEL FOR EMAIL

Probabilistic model for (bag-of-words, label) pair $(\mathbf{x}, y)$:

- Set $y = 0$ with probability p_0, $y = 1$ with probability $p_1 = 1 - p_0$.
    - p_0 is probability an email is not spam (e.g. 99%).
    - p_1 is probability an email is spam (e.g. 1%).
- If $y = 0$, for each i, set $x_i = 1$ with prob. p_{i0}.
- If $y = 1$, for each i, set $x_i = 1$ with prob. p_{i1}.

Unknown model parameters:

- p_0, p_1,
- $p_{10}, p_{20}, \ldots p_{n0}$, one for each of the n vocabulary words.
- $p_{11}, p_{21}, \ldots p_{n1}$, one for each of the n vocabulary words.

PARAMETER ESTIMATION

How to set parameters:

- Set p_0 and p_1 to the empirical fraction of not spam/spam emails.
- For each word i, set p_{i0} to the empirical probability word i appears in a non-spam email.
- For each word i, set p_{i1} to the empirical probability word i appears in a spam email.

Estimating these parameters from previous data examples is the only "training" we will do.

DONE WITH MODELING

ON TO PREDICTION

PROBABILITY REVIEW

- **Probability:** p(x) – the probability event *x* happens.
- **Joint probability:** p(x,y) – the probability that event *x* <u>and</u> event *y* happen.
- **Conditional Probability** $p(x \mid y)$ – the probability *x* happens <u>given</u> that *y* happens.

$$p(x|y) =$$

bayes theorem/rule

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

Proof:

CLASSIFICATION RULE

Given unlabeled input $(\mathbf{w}, \underline{\quad})$, choose the label $y \in \{0, 1\}$ which is most likely given the data. Recall $\mathbf{w} = [0, 0, 1, \ldots, 1, 0]$.

Classification rule: maximum a posterior (MAP) estimate.

Step 1. Compute:

- $p(y = 0 \mid \mathbf{w})$: prob. $y = 0$ given observed data vector $\mathbf{w}$.
- $p(y = 1 \mid \mathbf{w})$: prob. $y = 1$ given observed data vector $\mathbf{w}$.

Step 2. Output: 0 or 1 depending on which probability is larger.

$p(y = 0 \mid \mathbf{w})$ and $p(y = 1 \mid \mathbf{w})$ are called **posterior** probabilities.

EVALUATING THE POSTERIOR

How to compute the posterior? **Bayes rule!**

$$p(z = 0 \mid \mathbf{w}) = \frac{p(\mathbf{w} \mid y = 0)p(y = 0)}{p(\mathbf{w})} \tag{1}$$

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}} \tag{2}$$

- **Prior:** Probability in class 0 prior to seeing any data.
- **Posterior:** Probability in class 0 after seeing the data.

EVALUATING THE POSTERIOR

Goal is to determine which is larger:

$$p(y=0 \mid \mathbf{w}) = \frac{p(\mathbf{w} \mid y=0)p(y=0)}{p(\mathbf{w})} \qquad \text{vs.}$$

$$p(y=1 \mid \mathbf{w}) = \frac{p(\mathbf{w} \mid y=1)p(y=1)}{p(\mathbf{w})}$$

- We can ignore the evidence $p(\mathbf{w})$ since it is the same for both sides!
- $p(y=0)$ and $p(y=1)$ already known (computed from training data). These are our computed parameters p_0, p_1.
- $p(\mathbf{w} \mid y=0) = ?$ $p(\mathbf{w} \mid y=1) = ?$

EVALUATING THE POSTERIOR

Consider the example $\mathbf{w} = [0, 1, 1, 0, 0, 0, 1, 0]$.

Recall that, under our model, index i is 1 with probability p_{i0} if we are not spam, and 1 with probability p_{i1} if we are spam .

$$p(\mathbf{w} \mid y = 0) =$$

$$p(\mathbf{w} \mid y = 1) =$$

Final Naive Bayes Classifier

Training/Modeling: Use existing data to compute:

- $p_0 = p(y = 0), p_1 = p(y = 1)$
- For all i compute:
    - $p_{i0} = p(x_i = 1 \mid y = 0)$ and $(1 - p_{i0}) = p(x_i = 0 \mid y = 0)$
    - $p_{i1} = p(x_i = 1 \mid y = 1)$ and $(1 - p_{i1}) = p(x_i = 0 \mid y = 1)$

Prediction:

- For new input $\mathbf{w}$:
    - Compute $p(\mathbf{w} \mid y = 0) = \prod_i p(w_i \mid y = 0)$
    - Compute $p(\mathbf{w} \mid y = 1) = \prod_i p(w_i \mid y = 1)$
- Return

$$\arg\max \left[p\left(\mathbf{w} \mid y = 0\right) \cdot p\left(y = 0\right), p\left(\mathbf{w} \mid y = 1\right) \cdot p\left(y = 1\right) \right].$$

OTHER APPLICATIONS OF
THE BAYESIAN PERSPECTIVE

BAYESIAN REGRESSION

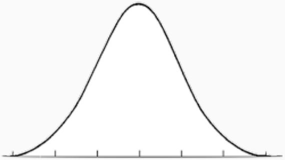
The Bayesian view offers an interesting alternative perspective on many machine learning techniques.

Example: Linear Regression.

Probabilistic model:

$$y = \langle \mathbf{x}, \boldsymbol{\beta} \rangle + \eta$$

where the η drawn from $N(0, \sigma^2)$ is **random Gaussian noise**.



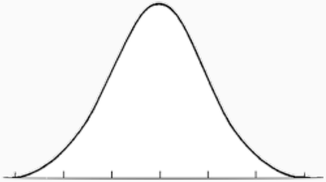
$$Pr(\eta = z) \sim$$

The symbol $\sim$ means "is proportional to".

GAUSSIAN DISTRIBUTION REFRESHER

Names for same thing: Normal distribution, Gaussian distribution, bell curve.

Parameterized by mean μ and variance σ^2.

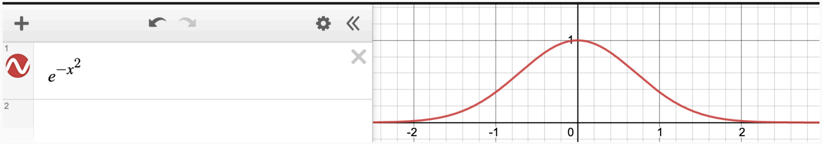


η is a continuous random variable, so it has a probability density function $p(\eta)$ with $\int_{-\infty}^{\infty} p(\eta) d\eta = 1$

$$p(\eta) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\eta-\mu}{\sigma}\right)^2}$$

GAUSSIAN DISTRIBUTION REFRESHER

The important thing to remember is that the the PDF falls off exponentially as we move further from the mean.



The normalizing constant in front 1/2, etc. don’t matter so much.

QUICK CHECK

Example: Linear Regression.

Probabilistic model:

$$y = \langle \mathbf{x}, \boldsymbol{\beta} \rangle + \eta$$

where the η drawn from $N(0, \sigma^2)$ is **random Gaussian noise**. The noise is independent for different inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$.

If we knew β what is the maximum a posterior (MAP) estimate for y given new data observation $\mathbf{x}$?

I.e. what value of y maximizes $\Pr(y \mid \mathbf{x})$?

How should be select β for our model?

Also use a Bayesian approach!

Choose β to maximize:

$$\text{posterior} = \Pr(\beta \mid X, y) = \frac{\Pr(X, y \mid \beta)\Pr(\beta)}{\Pr(X, y)} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}.$$

In this case, we don't have a prior – no values of β are inherently more likely than others.

Choose β to maximize just the likelihood:

$$\frac{\Pr(X, y \mid \beta)\cancel{\Pr(\beta)}}{\cancel{\Pr(X, y)}} = \frac{\text{likelihood} \times \cancel{\text{prior}}}{\cancel{\text{evidence}}}.$$

This is called the maximum likelihood estimate.

MAXIMUM LIKELIHOOD ESTIMATE

Data:

$$\mathbf{X} = \begin{bmatrix} - & \mathbf{x}_1 & - \\ - & \mathbf{x}_2 & - \\ & \vdots & \\ - & \mathbf{x}_n & - \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Model: $y_i = \langle \mathbf{x}_i, \boldsymbol{\beta} \rangle + \eta_i$ where $p(\eta_i = z) \sim e^{-z^2/2\sigma^2}$ and $\eta_1, \ldots, \eta_n$ are independent.

$$\Pr(\mathbf{X}, \mathbf{y} \mid \boldsymbol{\beta}) \sim$$

LOG LIKELIHOOD

Easier to work with the log likelihood:

$$
\begin{aligned}
\arg\max_{\boldsymbol{\beta}} \Pr(\mathbf{X}, \mathbf{y} \mid \boldsymbol{\beta}) &= \arg\max_{\boldsymbol{\beta}} \prod_{i=1}^{n} e^{-(y_i - \langle \mathbf{x}_i, \boldsymbol{\beta} \rangle)^2 / 2\sigma^2} \\
&= \arg\max_{\boldsymbol{\beta}} \ \log \left(\prod_{i=1}^{n} e^{-(y_i - \langle \mathbf{x}_i, \boldsymbol{\beta} \rangle)^2 / 2\sigma^2} \right) \\
&= \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{n} -(y_i - \langle \mathbf{x}_i, \boldsymbol{\beta} \rangle)^2 / 2\sigma^2 \\
&= \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \langle \mathbf{x}_i, \boldsymbol{\beta} \rangle)^2.
\end{aligned}
$$

MAXIMUM LIKELIHOOD ESTIMATOR

Conclusion: Choose $\boldsymbol{\beta}$ to minimize:

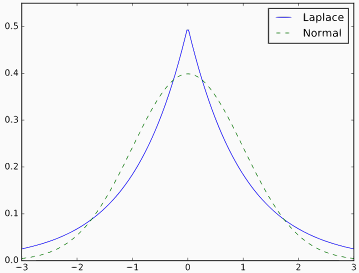
$$\sum_{i=1}^{n}(y_i - \langle \mathbf{x}_i, \boldsymbol{\beta} \rangle)^2 = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2.$$

This is a completely different justification for squared loss!

Minimizing the ℓ_2 loss is optimal in a certain sense when you assume your data follows a linear model with i.i.d. Gaussian noise.

BAYESIAN REGRESSION

If we had modeled our noise η as Laplace noise, we would have found that minimizing $\|\mathbf{y} - \mathbf{X}\beta\|_1$ was optimal.



$$Pr(\eta = z) \sim$$

Laplace noise has “heavier tails”, meaning that it results in more outliers.

This is a completely different justification for ℓ_1 loss.

BAYESIAN REGULARIZATION

We can add another layer of probabilistic modeling by also assuming $\boldsymbol{\beta}$ is random and comes from some distribution, which encodes our prior belief on what the parameters are.

Return to Maximum a posteriori (MAP estimation):

$$\Pr(\boldsymbol{\beta} \mid \mathrm{X}, \mathrm{y}) = \frac{\Pr(\mathrm{X}, \mathrm{y} \mid \boldsymbol{\beta}) \Pr(\boldsymbol{\beta})}{\Pr(\mathrm{X}, \mathrm{y})}.$$

Assume values in $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_d]$ come from some distribution.

- **Common model:** Each β_i drawn from $N(0, \gamma^2)$, i.e. normally distributed, independent.
- Encodes a belief that we are unlikely to see models with very large coefficients.

BAYESIAN REGULARIZATION

Goal: choose $\boldsymbol{\beta}$ to maximize:

$$\Pr(\boldsymbol{\beta} \mid \mathbf{X}, \mathbf{y}) = \frac{\Pr(\mathbf{X}, \mathbf{y} \mid \boldsymbol{\beta}) \Pr(\boldsymbol{\beta})}{\Pr(\mathbf{X}, \mathbf{y})}.$$

- We can still ignore the "evidence" term $\Pr(\mathbf{X}, \mathbf{y})$ since it is a constant that does not depend on $\boldsymbol{\beta}$.
- $\Pr(\boldsymbol{\beta}) = \Pr(\beta_1) \cdot \Pr(\beta_2) \cdot \ldots \cdot \Pr(\beta_d)$
- $\Pr(\boldsymbol{\beta}) \sim$

BAYESIAN REGULARIZATION

Easier to work with the **log likelihood**:

$$
\begin{aligned}
&\arg\max_{\boldsymbol{\beta}} \Pr(\mathbf{X}, \mathbf{y} \mid \boldsymbol{\beta}) \cdot \Pr(\boldsymbol{\beta}) \\
&= \arg\max_{\boldsymbol{\beta}} \prod_{i=1}^{n} e^{-(y_i - \langle \mathbf{x}_i, \boldsymbol{\beta}\rangle)^2/2\sigma^2} \cdot \prod_{i=1}^{n} e^{-(\beta_i)^2/2\gamma^2} \\
&= \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{n} -(y_i - \langle \mathbf{x}_i, \boldsymbol{\beta}\rangle)^2/2\sigma^2 + \sum_{i=1}^{d} -(\beta_i)^2/2\gamma^2 \\
&= \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \langle \mathbf{x}_i, \boldsymbol{\beta}\rangle)^2 + \frac{\sigma^2}{\gamma^2} \sum_{i=1}^{d} (\beta_i)^2/\sigma^2.
\end{aligned}
$$

Choose $\boldsymbol{\beta}$ to minimize $\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \frac{\sigma^2}{\gamma^2}\|\boldsymbol{\beta}\|_2^2$.

Completely different justification for ridge regularization!

BAYESIAN REGULARIZATION

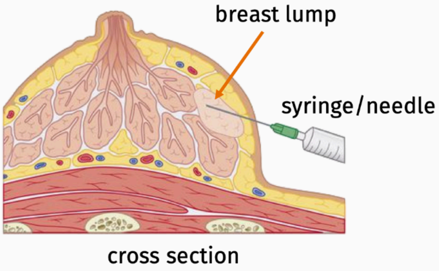
Test your intuition: What modeling assumption justifies LASSO regularization: $\min \|y - X\beta\|_2^2 + \lambda\|\beta\|_1$?

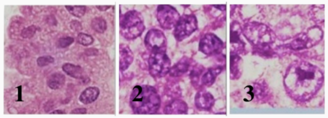
LINEAR CLASSIFICATION

MOTIVATING PROBLEM

Breast Cancer Biopsy: Determine if a breast lump in a patient is malignant (cancerous) or benign (safe).

- Collect cells from lump using fine needle biopsy.
- Stain and examine cells under microscope.
- Based on certain characteristics (shape, size, cohesion) determine if likely malignant or not).

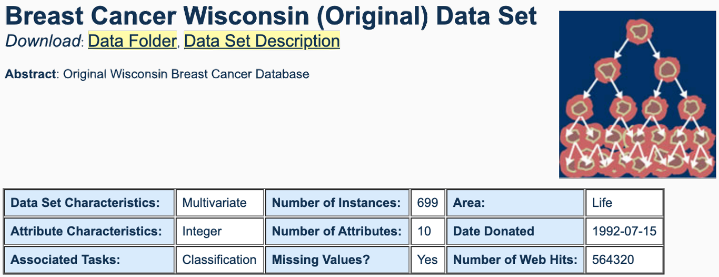




MOTIVATING PROBLEM

Demo: `demo_breast_cancer.ipynb`

Data: UCI machine learning repository


Breast Cancer Wisconsin (Original) Data Set
Download: Data Folder, Data Set Description

Abstract: Original Wisconsin Breast Cancer Database

Data Set Characteristics:	Multivariate	**Number of Instances:**	699	**Area:**	Life
Attribute Characteristics:	Integer	**Number of Attributes:**	10	**Date Donated**	1992-07-15
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	564320

`https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original)`

Features: 10 numerical scores about cell characteristics (Clump Thickness, Uniformity, Marginal Adhesion, etc.)

MOTIVATING PROBLEM

Data: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$.

$\mathbf{x}_i = [1, 5, 4 \ldots, 2]$ contains score values.

Label $y_i \in \{0, 1\}$ is 0 if benign cells, 1 if malignant cells.

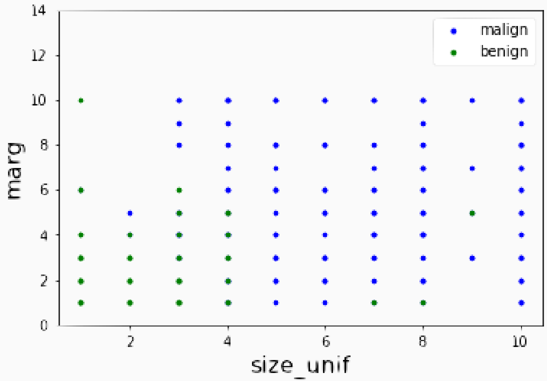
Goal: Based on scores (which would be collected manually, or even learned on their own using an ML algorithm) predict if a sample of cells is malignant or benign.

Approach:

- Naive Bayes Classifier can be extended to $\mathbf{x}$ with numerical values (instead of binary values as seen before). Will see on homework.
- **Today:** Learn a different type of classifier.

BEGIN BY PLOTTING DATA

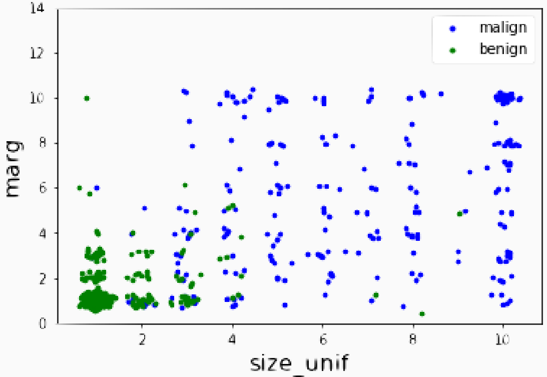
We pick two variables, Margin Adhesion and Size Uniformity and plot a scatter plot. Points with label 1 (malignant) are plotted in blue, those with label 2 (benign) are plotted in green.



Lots of overlapping points! Hard to get a sense of the data.

PLOTTING WITH JITTER

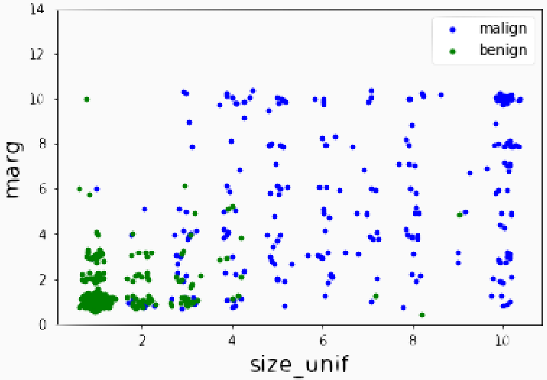
Simple + Useful Trick: data <u>jittering</u>. Add tiny random noise (using e.g. `np.random.randn`) to data to prevent overlap.



Noise is only for plotting. It is not added to the data for training, testing, etc.

BRAINSTORMING

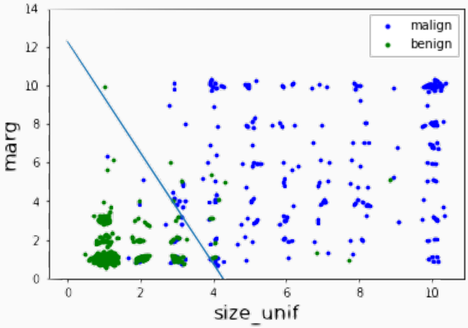
Any ideas for possible classification rules for this data?



LINEAR CLASSIFIER

Given vector of predictors $\mathbf{x}_i \in \mathbb{R}^d$ (here $d = 2$) find a parameter vector $\boldsymbol{\beta} \in \mathbb{R}^d$ and threshold λ.

- Predict $y_i = 0$ if $\langle \mathbf{x}_i, \boldsymbol{\beta} \rangle \leq \lambda$.
- Predict $y_i = 1$ if $\langle \mathbf{x}_i, \boldsymbol{\beta} \rangle > \lambda$

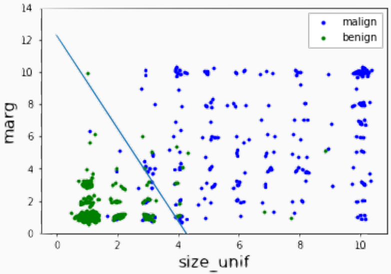


Line has equation $\langle \mathbf{x}, \boldsymbol{\beta} \rangle = \lambda$.

LINEAR CLASSIFIER

As long as we append a 1 onto each data vector $\mathbf{x}_i$ (i.e. a column of ones onto the data matrix $\mathbf{X}$) like we did for linear regression, an equivalent function is:

- Predict $y_i = 0$ if $\langle \mathbf{x}_i, \boldsymbol{\beta} \rangle \leq 0$.
- Predict $y_i = 1$ if $\langle \mathbf{x}_i, \boldsymbol{\beta} \rangle > 0$



Line has equation $\langle \mathbf{x}, \boldsymbol{\beta} \rangle = 0$.

$0 - 1$ loss

Question: How do we find a good linear classifier automatically?

Loss minimization approach (first attempt):

- **Model**[1]:

$$f_{\boldsymbol{\beta}}(\mathbf{x}) = \mathbb{1}\left[\langle \mathbf{x}, \boldsymbol{\beta} \rangle > 0\right]$$

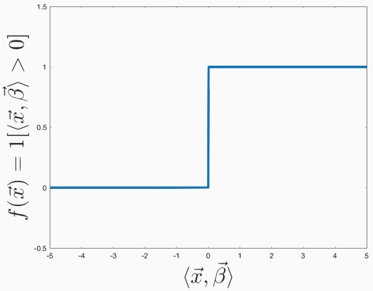
- **Loss function**: "$0 - 1$ Loss"

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} |f_{\boldsymbol{\beta}}(\mathbf{x}_i) - y_i|$$

[1] $\mathbb{1}$[event] is the indicator function: it evaluates to 1 if the argument inside is true, 0 if false.

$0-1$ loss

Problem with $0-1$ loss:



- The loss function $L(\boldsymbol{\beta})$ is not differentiable because $f_{\boldsymbol{\beta}}(\mathbf{x})$ is discontinuous.
- Impossible to take the gradient, very hard to minimize loss to find optimal $\boldsymbol{\beta}$.
- Non-convex function (will make more sense next lecture).

LINEAR CLASSIFIER VIA SQUARE LOSS

Loss minimization approach (second attempt):

- Model:

$$f_{\boldsymbol{\beta}}(\mathbf{x}) = \mathbb{1}\left[\langle \mathbf{x}, \boldsymbol{\beta} \rangle > 1/2\right]$$

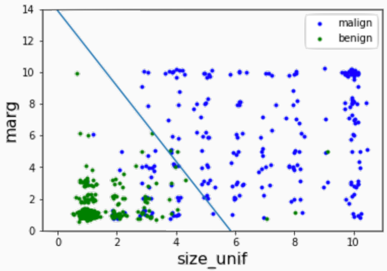
- Loss function: "Square Loss"

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} (\langle \mathbf{x}, \boldsymbol{\beta} \rangle - y_i)^2$$

Intuitively tries to make $\langle \mathbf{x}, \boldsymbol{\beta} \rangle$ close to 0 for examples in class 0, close to 1 for examples in class 1.

LINEAR CLASSIFIER VIA SQUARE LOSS

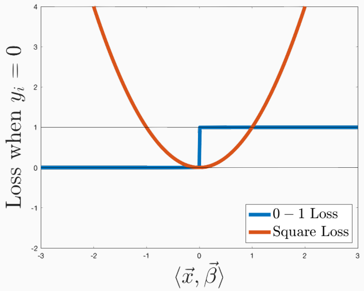
We can solve for β by just solving a least squares multiple linear regression problem.



Do you see any issues here?

LINEAR CLASSIFIER VIA SQUARE LOSS

Problem with square loss:

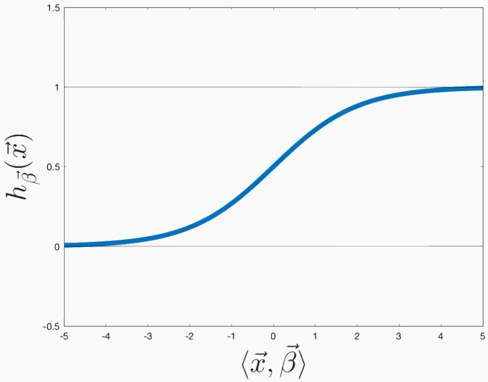


- Loss increases if $\langle \mathbf{x}, \boldsymbol{\beta} \rangle > 1$ even if correct label is 1. Or if $\langle \mathbf{x}, \boldsymbol{\beta} \rangle < 0$ even if correct label is 0.
- Intuitively we don’t want to “punish” these cases.

LOGISTIC REGRESSION

Let $h_{\boldsymbol{\beta}}(\mathbf{x})$ be the logistic function:

$$h_{\boldsymbol{\beta}}(\mathbf{x}) = \frac{1}{1 + e^{-\langle \boldsymbol{\beta}, \mathbf{x} \rangle}}$$



Loss minimization approach (this works!):

- **Model**: Let $h_{\boldsymbol{\beta}}(\mathbf{x}) = \frac{1}{1+e^{-\langle \boldsymbol{\beta}, \mathbf{x} \rangle}}$

$$f_{\boldsymbol{\beta}}(\mathbf{x}) = \mathbb{1}\left[h_{\boldsymbol{\beta}}(\mathbf{x}) > 1/2\right]$$

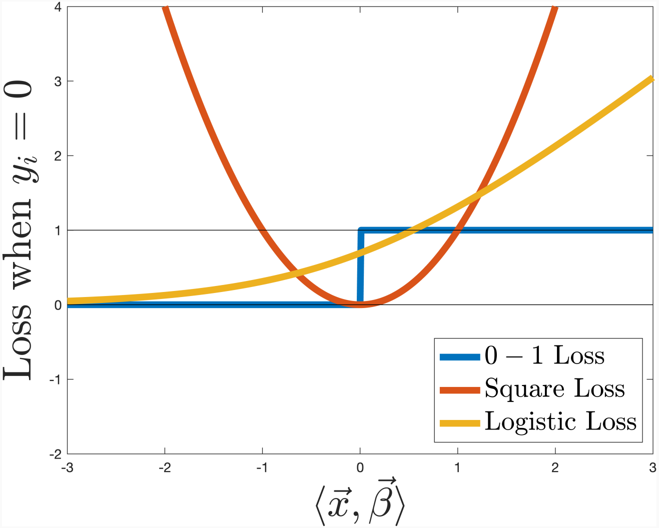
- **Loss function**: "Logistic loss" aka "binary cross-entropy loss"

$$L(\beta) = -\sum_{i=1}^{n} y_i \log(h_{\boldsymbol{\beta}}(\mathbf{x})) + (1 - y_i) \log(1 - h_{\boldsymbol{\beta}}(\mathbf{x}))$$

LOGISTIC LOSS

Logistic Loss:

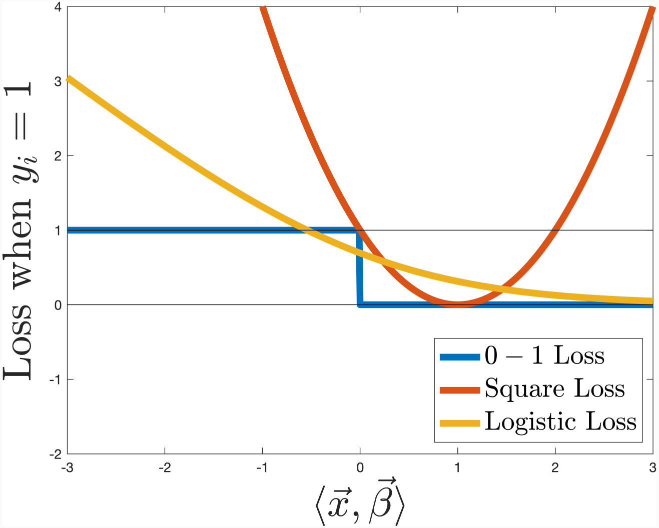
$L(\beta) = -\sum_{i=1}^{n} y_i \log(h_{\boldsymbol{\beta}}(\mathbf{x})) + (1 - y_i)\log(1 - h_{\boldsymbol{\beta}}(\mathbf{x}))$



LOGISTIC LOSS

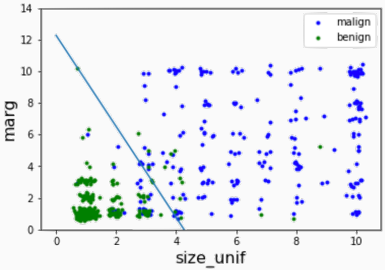
Logistic Loss:

$L(\boldsymbol{\beta}) = -\sum_{i=1}^{n} y_i \log(h_{\boldsymbol{\beta}}(\mathbf{x})) + (1 - y_i)\log(1 - h_{\boldsymbol{\beta}}(\mathbf{x}))$



LOGISTIC LOSS

- Convex function in β, can be minimized using gradient descent (next lecture).
- Works well in practice.
- Good Bayesian motivation: see posted lecture notes if you are interested.



Fit using logistic regression/log loss.

ERROR IN CLASSIFICATION

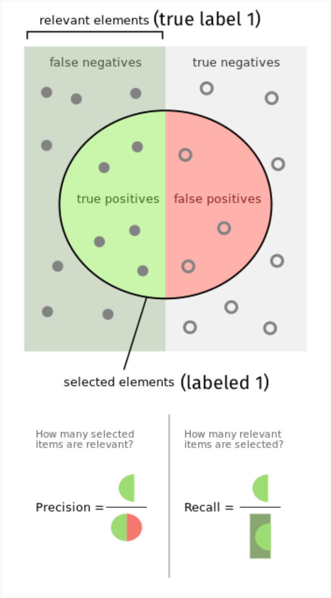
Once we have a classification algorithm, how do we judge its performance?

- **Simplest answer:** Error rate = fraction of data examples misclassified in test set.
- What are some issues with this approach?

ERROR IN CLASSIFICATION

- **Precision:** Fraction of positively labeled examples (label 1) which are correct.
- **Recall:** Fraction of true positives that we labeled correctly with label 1.

Question: Which should we optimize for medical diagnosis?


relevant elements (true label 1)
false negatives
true negatives
true positives
false positives
selected elements (labeled 1)
How many selected items are relevant?
How many relevant items are selected?
Precision =
Recall =

Logistic regression workflow:

- Select $\boldsymbol{\beta}$ via training and compute $h_{\boldsymbol{\beta}}(\mathrm{x}_i) = \frac{1}{1+e^{-\langle \mathrm{x}_i, \boldsymbol{\beta} \rangle}}$ for all x_i.
- Predict $y_i = 0$ if $h_{\boldsymbol{\beta}}(\mathrm{x}_i) \leq \lambda$, $y_i = 1$ if $h_{\boldsymbol{\beta}}(\mathrm{x}_i) > \lambda$.
- Default value of λ is 1/2. Increasing λ improves <u>precision</u>. Decreasing λ improves <u>recall</u>.

ERROR IN CLASSIFICATION

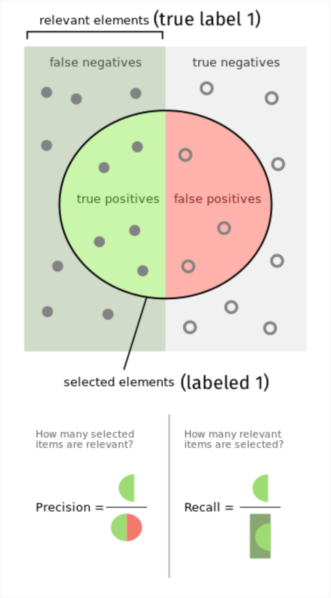
Once we have a classification algorithm, how do we judge its performance?

- **Simplest answer:** Error rate = fraction of data examples misclassified in test set.
- What are some issues with this approach?

ERROR IN CLASSIFICATION

- **Precision:** Fraction of positively labeled examples (label 1) which are correct.
- **Recall:** Fraction of true positives that we labeled correctly with label 1.

Question: Which should we optimize for medical diagnosis?



relevant elements (true label 1)
false negatives
true negatives
true positives
false positives
selected elements (labeled 1)
How many selected items are relevant?
How many relevant items are selected?
Precision =
Recall =

ERROR IN CLASSIFICATION

Possible logistic regression workflow:

- Learn $\vec{\beta}$ and compute $h_{\vec{\beta}}(\vec{x}_i) = \frac{1}{1+e^{-\langle \vec{x}_i, \vec{\beta} \rangle}}$ for all $\vec{x}_i$.
- Predict $y_i = 0$ if $h_{\vec{\beta}}(\vec{x}_i) \leq \lambda$, $y_i = 1$ if $h_{\vec{\beta}}(\vec{x}_i) > \lambda$.
- Default value of λ is 1/2. Increasing λ improves precision. Decreasing λ improves recall.

This is very heuristic. There are other methods for handling "class imbalance" which can often lead to good overall error, but poor precision or recall. Techniques include weighting the loss function to care more about false negatives, or subsampling the larger class.

MULTICLASS

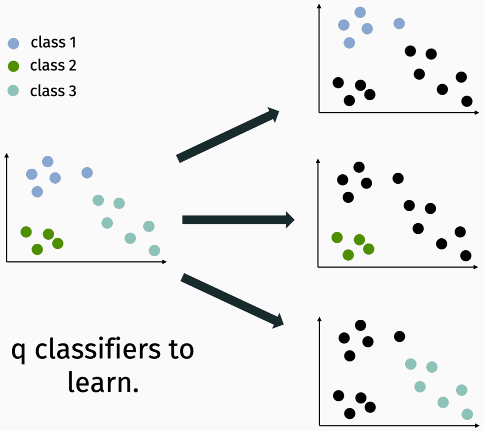
What about when $y \in \{1, \ldots, q\}$ instead of $y \in \{0, 1\}$

Two options for multiclass data:

- One-vs.-all (most common, also called one-vs.-rest)
- One-vs.-one (slower, but can be more effective)

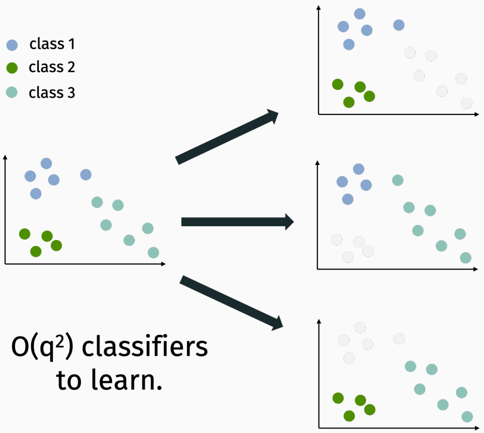
In both cases, we convert to multiple binary classification problem.

ONE VS. REST



- For q classes train q classifiers. Obtain parameters $\vec{\beta}_1, \ldots, \vec{\beta}_q$.
- Assign y to class i with maximum $\langle \vec{\beta}_i, \vec{x} \rangle$.

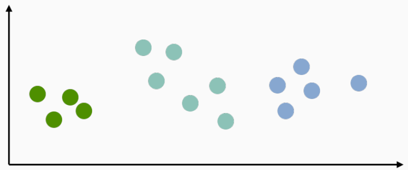
ONE VS. REST



- For q classes train $\frac{q(q-1)}{2}$ classifiers.
- Assign y to class which i which wins in the most number of head-to-head comparisons.

ONE VS. ONE

Hard case for one-vs.-all.



- One-vs.-one would be a better choice here.
- Also tends to work better when there is class in balance.

softmax/multiclass cross-entropy loss

We will see one more very common approach related to one-vs-rest either next lecture or when we do neural nets.

ERROR IN (MULTICLASS) CLASSIFICATION

Confusion matrix for k classes:

Pred--> Real↓	1	2	…	K
1				
2				
…				
K				

- Entry i, j is the fraction of class i items classified as class j.
- Useful to see whole matrix to visualize where errors occur.