Day 3: Classification, logistic regression
Topics so far

• Supervised learning, linear regression

• Yesterday
  o Overfitting,
  o Ridge and lasso Regression
  o Gradient descent

• Today
  o Bias variance trade-off
  o Classification
  o Logistic regression
  o Regularization for logistic regression
  o Classification metrics
Bias-variance tradeoff
Empirical vs population loss

- **Population distribution** Let \((x, y) \sim \mathcal{D}\)
- We have
  - Loss function \(\ell(\hat{y}, y)\)
  - Hypothesis class \(\mathcal{H}\)
  - Training data \(S = \{(x^{(i)}, y^{(i)}): i = 1, 2, \ldots, N\} \sim_{\text{iid}} \mathcal{D}^N\)
    - Think of \(S\) as random variable
- What we really want \(f \in \mathcal{H}\) to minimize **population loss**
  \[
  L_{\mathcal{D}}(f) \triangleq \mathbb{E}_{\mathcal{D}}[\ell(f(x), y)] = \int_{(x,y)} \ell(f(x), y) \Pr(x, y)
  \]
- ERM minimizes **empirical loss**
  \[
  L_{\mathcal{S}}(f) \triangleq \hat{\mathbb{E}}_{\mathcal{S}}[\ell(f(x), y)] = \frac{1}{N} \sum_{i=1}^{N} \ell(f(x^{(i)}), y^{(i)})
  \]

e.g, \(\Pr(x) = \text{uniform}(0,1)\)
\(y = w^* x + \epsilon\) where \(\epsilon = \mathcal{N}(0, 0.1)\)
\(\Rightarrow \Pr(y|x) = \mathcal{N}(w^* x, 0.1)\)
\(\Pr(x, y) = \Pr(x) \Pr(y|x)\)
Empirical vs population loss

\[ L(f) \triangleq \mathbb{E}_D[\ell(f(x), y)] = \int_{(x,y)} \ell(f(x), y) \Pr(x, y) \]

\[ L_S(f) \triangleq \hat{\mathbb{E}}_S[\ell(f(x), y)] = \frac{1}{N} \sum_{i=1}^{N} \ell(f(x^{(i)}), y^{(i)}) \]

- \( \hat{f}_S \) from some model **overfits** to \( S \) if there is \( f^* \in \mathcal{H} \) with

  \[ \hat{\mathbb{E}}_S[\ell(\hat{f}_S(x), y)] \leq \hat{\mathbb{E}}_S[\ell(f^*(x), y)] \text{ but } \mathbb{E}_D[\ell(\hat{f}_S(x), y)] \gg \mathbb{E}_D[\ell(f^*(x), y)] \]

- If \( f \) is independent of \( S_{train} \) then both \( L_{S_{train}}(f) \) and \( L_{S_{test}}(f) \) are good approximations of \( L_D(f) \)

- But generally, \( \hat{f} \) depends on \( S_{train} \). Why?
  - \( L_{S_{train}}(\hat{f}_{S_{train}}) \) is no more a good approximation of \( L_D(f) \)
  - \( L_{S_{test}}(\hat{f}_{S_{train}}) \) is still a good approximation of \( L_D(f) \) since \( \hat{f}_{S_{train}} \) is independent of \( S_{test} \)
Optimum Unrestricted Predictor

• Consider population squared loss

\[
\arg\min_{f \in \mathcal{H}} L(f) \triangleq \mathbb{E}_D[\ell(f(x), y)] = \mathbb{E}_{(x,y)}[(f(x) - y)^2]
\]

• Say \( \mathcal{H} \) is unrestricted – any function \( f: x \rightarrow y \) is allowed

\[
L(f) = \mathbb{E}_{(x,y)}[(f(x) - y)^2] = \mathbb{E}_x\left[\mathbb{E}_y[(f(x) - y)^2 | x]\right]
\]

\[
= \mathbb{E}_x\left[\mathbb{E}_y\left[(f(x) - \mathbb{E}_y[y|x] + \mathbb{E}_y[y|x] - y)^2 | x\right]\right]
\]

\[
= \mathbb{E}_x\left[\mathbb{E}_y\left[(f(x) - \mathbb{E}_y[y|x])^2 | x\right]\right] + \mathbb{E}_x\left[\mathbb{E}_y\left[(\mathbb{E}_y[y|x] - y)^2 | x\right]\right]
\]

\[
+ 2 \mathbb{E}_x\left[\mathbb{E}_y\left[(f(x) - \mathbb{E}_y[y|x])(\mathbb{E}_y[y|x] - y)|x\right]\right]
\]

\[
= 0
\]

\[
= \mathbb{E}_x[(f(x) - \mathbb{E}_y[y|x])^2] + \mathbb{E}_{x,y}[(\mathbb{E}_y[y|x] - y)^2]
\]

minimized for \( f = \mathbb{E}_y[y|x] \)

Noise
Bias variance decomposition

• Best unrestricted predictor $f^{**}(x) = E_y[y|x]$

• $L(f_S) = E_x[(f_S(x) - f^{**}(x))^2] + E_{x,y}[(f^{**}(x) - y)^2]$

• $E_S L(f_S) = E_S E_x[(f_S(x) - f^{**}(x))^2] + noise$

$$
E_S E_x [(f_S(x) - f^{**}(x))^2] = E_x [E_S [(f_S(x) - f^{**}(x))^2|x]]
= E_x E_S [(f_S(x) - E_S[f_S(x)] + E_S[f_S(x)] - f^{**}(x))^2|x]
= E_x E_S [(f_S(x) - E_S[f_S(x)])^2|x] + E_x [(E_S[f_S(x)] - f^{**}(x))^2]
+ 2E_x [E_S [(E_S[f_S(x)] - f^{**}(x))(f_S(x) - E_S[f_S(x)])|x]]
= E_{S,x}[(f_S(x) - E_S[f_S(x)])^2] + E_x [(E_S[f_S(x)] - f^{**}(x))^2]
$$

$E_S L(f_S) = E_{S,x} [(f_S(x) - E_S[f_S(x)])^2]$
$+ E_x [(E_S[f_S(x)] - f^{**}(x))^2]
+ E_{x,y}[(f^{**}(x) - y)^2] = variance$
$+ bias^2$
$+ noise$
Bias-variance tradeoff

$$E_S L(f_S) = E_{S,x} [(f_S(x) - E_S[f_S(x)])^2]$$
$$+ E_x [(E_S[f_S(x)] - f^{**}(x))^2]$$
$$+ E_{x,y} [(f^{**}(x) - y)^2]$$

- $f_S \in \mathcal{H}$
- noise is irreducible
- variance can be reduced by
  - get more data
  - make $f_S$ less sensitive to $S$
    - less number of candidates in $\mathcal{H}$ to choose from $\Rightarrow$ less variance
    - reducing the “complexity” of model class $\mathcal{H}$ decreases variance
- $bias^2 \geq \min_{f \in conv(\mathcal{H})} E_x [(f(x) - f^{**}(x))^2]$
  - expanding model class $\mathcal{H}$ decreases bias
Model complexity

- reducing the complexity of model class $\mathcal{H}$ decreases variance
- expanding model class $\mathcal{H}$ decreases bias
- Complexity $\approx$ number of choices in $\mathcal{H}$
  - For any loss $L$, for all $f \in \mathcal{H}$ with probability greater than $1 - \delta$
    \[ L(f) \leq L_s(f) + \sqrt{\log|\mathcal{H}| + \log \frac{1}{\delta}} \frac{1}{N} \]
  - many other variants for infinite cardinality classes
  - often bounds are loose
- Complexity $\approx$ number of degrees of freedom
  - e.g., number of parameters to estimate
  - more data $\Rightarrow$ can fit more complex models
- Is $\mathcal{H}_1 = \{ x \rightarrow w_0 + w_1 \cdot x - w_2 \cdot x \}$ more complex than $\mathcal{H}_2 = \{ x \rightarrow w_0 + w_1 \cdot x \}$?
  - What we need is how many different “behaviors” we can get on same $S$
Summary

• Overfitting
  o What is overfitting?
  o How to detect overfitting?
  o Avoiding overfitting using model selection

• Bias – variance tradeoff
Classification

• Supervised learning: estimate a mapping $f$ from input $x \in X$ to output $y \in Y$
  
  o **Regression** $Y = \mathbb{R}$ or other continuous variables
  
  o **Classification** $Y$ takes discrete set of values
  
  ▪ Examples:
    
    □ $Y = \{\text{spam, nospam}\}$,
    
    □ digits (not values) $Y = \{0, 1, 2, \ldots, 9\}$

• Many successful applications of ML in vision, speech, NLP, healthcare
Classification vs Regression

• Label-values do not have meaning
  o $Y = \{\text{spam, nospam}\}$ or $Y = \{0,1\}$ or $Y = \{-1,1\}$
• Ordering of labels does not matter (for most parts)
  o $f(x) = "0"$ when $y = "1"$ is as bad as $f(x) = "9"$ when $y = "1"$
• Often $f(x)$ does not return labels $y$
  o e.g. in binary classification with $Y = \{-1,1\}$ we often estimate $f: \mathcal{X} \rightarrow \mathbb{R}$ and then post process to get $\hat{y}(f(x)) = 1[f(x) \geq 0]$
    o mainly for computational reasons
      ▪ remember, we need to solve $\min_{f \in \mathcal{H}} \sum_i \ell(f(x^{(i)}), y^{(i)})$
      ▪ discrete values $\rightarrow$ combinatorial problems $\rightarrow$ hard to solve
  o more generally $\mathcal{H} \subset \{f: \mathcal{X} \rightarrow \mathbb{R}\}$ and loss $\ell: \mathbb{R} \times Y \rightarrow \mathbb{R}$
    ▪ compare to regression, where typically $\mathcal{H} \subset \{f: \mathcal{X} \rightarrow Y\}$ and loss $\ell: Y \times Y \rightarrow \mathbb{R}$
Non-parametric classifiers
Nearest Neighbor (NN) Classifier

- Training data $S = \{(x^{(i)}, y^{(i)}) : i = 1, 2, ..., N\}$
- Want to predict label of new point $x$
- Nearest Neighbor Rule
  - Find the closest training point: $i^* = \arg \min_i \rho(x, x^{(i)})$
  - Predict label of $x$ as $\hat{y}(x) = y^{(i^*)}$
- Computation
  - Training time: Do nothing
  - Test time: search the training set for a NN

Figure credit: Nati Srebro
Nearest Neighbor (NN) Classifier

• Where is the main model?
  o \( i^* = \arg \min_i \rho(x, x^{(i)}) \)
  o What is the right “distance” between images? Between sound waves? Between sentences?
  o Often \( \rho(x, x') = \|\phi(x) - \phi(x')\|_2 \) or other norms \( \|x - x'\|_1 \)

Slide credit: Nati Srebro
k-Nearest Neighbor (kNN) classifier

• Training data $S = \{(x^{(i)}, y^{(i)}): i = 1, 2, \ldots, N\}$

• Want to predict label of new point $x$

• $k$-Nearest Neighbor Rule
  
  o Find the $k$ closest training point: $i_1^*, i_2^*, \ldots, i_k^*$

  o Predict label of $x$ as
  
  $\hat{y}(x) = \text{majority}(y^{(i_1^*)}, y^{(i_2^*)}, \ldots, y^{(i_k^*)})$

• Computation
  
  o Training time: Do nothing

  o Test time: search the training set for $k$ NNs
$k$-Nearest Neighbor

**Advantages**
- no training
- universal approximator – non-parametric

**Disadvantages**
- not scalable
  - test time memory requirement
  - test time computation
- easily overfits with small data
Training vs test error

1-NN
• Training error?
  • 0
• Test error?
  • Depends on Pr(\(x, y\))

k-NN
• Training error: can be greater than 0
• Test error: again depends on Pr(\(x, y\))

Figure credit: Nati Srebro
**k-Nearest Neighbor: Data Fit / Complexity Tradeoff**

\[ S = \]

\[ h^* = \]

Slide credit: Nati Srebro
Space partition

• kNN partitioning of $\mathcal{X}$ (or $\mathbb{R}^d$) into regions of +1 and -1

• What about discrete valued features $x$?

• Even for continuous $x$, can we get more structured partitions?
  - easy to describe
    - e.g., $R_2 = \{x: x_1 < t_1 \text{ and } x_2 > t_2\}$
  - reduces degrees of freedom

• Any non-overlapping partition using only (hyper) rectangles
  $\rightarrow$ representable by a tree

Figure credit: Greg Shaknarovich
Decision trees

• Focus on binary trees (trees with at most two children at each node)

• How to create trees?

• What is a “good” tree?
  
  o Measure of “purity” at each leaf node where each leaf node corresponding to a region $R_i$

  $$\text{purity}(\text{tree}) = \sum_{R_i} |\text{# blue at } R_i - \# \text{ red at } R_i|$$

  There are various metrics of (im)purity that are used in practice, but the rough idea is the same
Decision trees

• How to create trees?
• Training data $S = \{(x^{(i)}, y^{(i)}): i = 1, 2, \ldots, N\}$, where $y^{(i)} \in \{\text{blue, red}\}$
• At each point,

$$\text{purity}(\text{tree}) = \sum_{\text{leaf}} |\# \text{ blue at leaf} - \# \text{ red at leaf}|$$

• Start with all data at root
  o only one leaf = root. What is purity(tree)?
Decision trees

• How to create trees?
• Training data \( S = \{(x^{(i)}, y^{(i)}): i = 1, 2, \ldots, N\} \), where \( y^{(i)} \in \{\text{blue, red}\} \)
• At each point,

\[
purity(tree) = \sum_{\text{leaf}} |\# \text{ blue at leaf} - \# \text{ red at leaf}| \]

• Start with all data at root
  o only one leaf = root. What is purity(tree)?
• Create a split based on a rule that increases the amount of “purity” of tree.
  o How complex can the rules be?
Decision trees

• How to create trees?
• Training data $S = \{(x^{(i)}, y^{(i)}): i = 1, 2, \ldots, N\}$, where $y^{(i)} \in \{\text{blue, red}\}$
• At each point,
  
  \[
  \text{purity}(\text{tree}) = \sum_{\text{leaf}} |\# \text{ blue at leaf} - \# \text{ red at leaf}|
  \]
• Start with all data at root
  o only one leaf = root. What is purity(tree)?
• Create a split based on a rule that increases the amount of “purity” of tree.
  o How complex can the rules be?
• Repeat

When to stop?
what is the complexity of a DT?
• Limit the number of leaf nodes
Decision trees

• **Advantages**
  - interpretable
  - easy to deal with non-numeric features
  - natural extensions to multi-class, multi-label

• **Disadvantages**
  - not scalable
  - hard decisions – non-smooth decisions
  - often overfits in spite of regularization

• Check CART package in scikit-learn
Parametric classifiers

• What is the equivalent of linear regression?
  o something easy to train
  o something easy to use at test time

• \( f(x) = f_w(x) = w \cdot x + w_0 \)

• \( \mathcal{H} = \{f_w = x \rightarrow w \cdot x + w_0 : w \in \mathbb{R}^d, w_0 \in \mathbb{R}\} \)

• but \( f(x) \notin \{-1,1\}! \) how do we get labels?
  o reasonable choice
    \( \hat{y}(x) = 1 \) if \( f_{\hat{w}}(x) \geq 0 \) and \( \hat{y}(x) = -1 \) otherwise
  o linear classifier: \( \hat{y}(x) = \text{sign}(\hat{w} \cdot x + \hat{w}_0) \)
Parametric classifiers

- $\mathcal{H} = \{ f_w = x \rightarrow w \cdot x + w_0 : w \in \mathbb{R}^d, w_0 \in \mathbb{R} \}$

- $\hat{y}(x) = \text{sign}(\hat{w} \cdot x + \hat{w}_0)$

- $\hat{w} \cdot x + \hat{w}_0 = 0$ (linear) decision boundary or separating hyperplane
  - that separates $\mathbb{R}^d$ into two halfspaces (regions)
    - $\hat{w} \cdot x + \hat{w}_0 > 0$ and $\hat{w} \cdot x + \hat{w}_0 < 0$

- more generally, $\hat{y}(x) = \text{sign} \left( \hat{f}(x) \right)$
  - decision boundary is $\hat{f}(x) = 0$
Linear classifier
Classification vs Regression

• Label-values do not have meaning
  o $\mathcal{Y} = \{\text{spam, nospam}\}$ or $\mathcal{Y} = \{0,1\}$ or $\mathcal{Y} = \{-1,1\}$

• Ordering of labels does not matter (for most parts)
  o $f(x) = “0”$ when $y = “1”$ is as bad as $f(x) = “9”$ when $y = “1”$

• Often $f(x)$ does not return labels $y$
  o e.g. in binary classification with $\mathcal{Y} = \{-1,1\}$ we often estimate
    $f: \mathcal{X} \to \mathbb{R}$ and then post process to get $\hat{y}(f(x)) = 1[f(x) \geq 0]$
  o mainly for computational reasons
    ▪ remember, we need to solve $\min_{f \in \mathcal{H}} \sum_i \ell(f(x^{(i)}), y^{(i)})$
    ▪ discrete values $\rightarrow$ combinatorial problems $\rightarrow$ hard to solve
  o more generally $\mathcal{H} \subset \{f: \mathcal{X} \to \mathbb{R}\}$ and loss $\ell: \mathbb{R} \times \mathcal{Y} \rightarrow \mathbb{R}$
    ▪ compare to regression, where typically $\mathcal{H} \subset \{f: \mathcal{X} \rightarrow \mathcal{Y}\}$ and
      loss $\ell: \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$
Classification vs Regression

• Label-values do not have meaning
  o \( y = \{ \text{spam}, \text{nospam} \} \) or \( y = \{ 0, 1 \} \) or \( y = \{ -1, 1 \} \)

• Ordering of labels does not matter (for most parts)
  o \( f(\mathbf{x}) = \text{"0" when } y = \text{"1"} \)

• Often \( f(\mathbf{x}) \) does not return labels
  o e.g. in binary classification with \( y = \{ -1, 1 \} \) we often estimate \( f: \mathcal{X} \rightarrow \mathbb{R} \) and then post-process to get \( y(\mathbf{x}) = 1 \text{ if } f(\mathbf{x}) \geq 0 \) mainly for computational reasons

What if we ignore above and solve classification using regression?

• discrete values \( \rightarrow \) combinatorial problems \( \rightarrow \) hard to solve
  o more generally \( \mathcal{H} \subset \{ f: \mathcal{X} \rightarrow \mathbb{R} \} \) and loss \( \ell: \mathbb{R} \times \mathcal{Y} \rightarrow \mathbb{R} \)
  ▪ compare to regression, where typically \( \mathcal{H} \subset \{ f: \mathcal{X} \rightarrow \mathcal{Y} \} \) and loss \( \ell: \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R} \)
Classification as regression

- Binary classification $\mathcal{Y} = \{-1, 1\}$ and $\mathcal{X} \in \mathbb{R}^d$
- Treat it as regression with squared loss, say linear regression
  - Training data $S = \{(x^{(i)}, y^{(i)}): i = 1, 2, ..., N\}$
  - ERM
    $$\hat{w}, \hat{w}_0 = \arg\min_{w, w_0} \sum_{i} (w \cdot x^{(i)} + w_0 - y^{(i)})^2$$
Classification as regression

\[ \hat{y}(x) = \text{sign}(wx + w_0) \]

Example credit: Greg Shaknarovich
Classification as regression

classified correctly by
\( \hat{y}(x) = \text{sign}(w \cdot x) \)
but squared loss \((w \cdot x + 1)^2\) will be high

Example credit: Greg Shaknarovich
Classification as regression

\[ x \quad y = +1 \quad y = -1 \]

Example credit: Greg Shaknarovich
Classification as regression
Surrogate Losses

• The correct loss to use is 0-1 loss \textit{after} thresholding

\[ \ell^{01}(f(x), y) = 1[\text{sign}(f(x)) \neq y] \]

\[ = 1[\text{sign}(f(x)y) < 0] \]
Surrogate Losses

• The correct loss to use is 0-1 loss after thresholding
  \[ \ell^{01}(f(x), y) = 1[\text{sign}(f(x)) \neq y] \]
  \[ = 1[\text{sign}(f(x)y) < 0] \]

• Linear regression uses \( \ell^{LS}(f(x), y) = (f(x) - y)^2 \)

• Why not do ERM over \( \ell^{01}(f(x), y) \) directly?
  - non-continuous, non-convex
Surrogate Losses

• Hard to optimize over $\ell^{01}$, find another loss $\ell(\hat{y}, y)$
  o Convex (for any fixed $y$) $\rightarrow$ easier to minimize
  o An upper bound of $\ell^{01}$ $\rightarrow$ small $\ell \Rightarrow$ small $\ell^{01}$

• Satisfied by squared loss
  $\rightarrow$ but has “large” loss even when $\ell^{01}(\hat{y}, y) = 0$

• Two more surrogate losses in this course
  o Logistic loss
    $$\ell^{\log}(\hat{y}, y) = \log(1 + \exp(-\hat{y}y))$$
    (TODAY)
  o Hinge loss
    $$\ell^{hinge}(\hat{y}, y) = \max(0, 1 - \hat{y}y)$$
    (TOMORROW)
Logistic Regression
Logistic regression: ERM on surrogate loss

Logistic loss
\[ \ell(f(x), y) = \log(1 + \exp(-f(x)y)) \]

- \( S = \{(x^{(i)}, y^{(i)}): i = 1, 2, \ldots, N\}, \ X = \mathbb{R}^d, \ Y = \{-1, 1\} \)
- Linear model \( f(x) = f_w(x) = w \cdot x + w_0 \)
- Minimize training loss
  \[ \hat{w}, \hat{w}_0 = \arg\min_{w, w_0} \sum_i \log \left( 1 + \exp \left( -(w \cdot x^{(i)} + w_0)y^{(i)} \right) \right) \]
- Output classifier \( \hat{y}(x) = \text{sign}(w \cdot x + w_0) \)
Logistic regression

\[ \hat{w}, \hat{w}_0 = \arg\min_{w, w_0} \sum_i \log \left( 1 + \exp\left(-\left(w \cdot x^{(i)} + w_0\right)y^{(i)}\right) \right) \]

- Learns a linear decision boundary
  - \( \{x: w \cdot x + w_0 = 0\} \) is a hyperplane in \( \mathbb{R}^d \) - decision boundary
  - \( \{x: w \cdot x + w_0 = 0\} \) divides \( \mathbb{R}^d \) into two halfspace (regions)
  - \( \{x: w \cdot x + w_0 \geq 0\} \) will get label +1 and
    \( \{x: w \cdot x + w_0 < 0\} \) will get label -1

- Maps \( x \) to a 1D coordinate
  \[ x' = \frac{w \cdot x + w_0}{||w||} \]

Figure credit: Greg Shaknarovich
Logistic Regression

\[ \hat{w}, \hat{w}_0 = \arg\min_{w, w_0} \sum_i \log(1 + \exp(-(w \cdot x + w_0)y)) \]

- Convex optimization problem
- Can solve using gradient descent
- Can also add usual regularization: \( \ell_2, \ell_1 \)
  - More details in the next session