Classification: Linear Models

Oliver Schulte
Deep Learning 980
Classification Problems
Machine Learning as Program Synthesis

- We can think of a machine learning system as a program that produces a program
Classifying Examples

- Many predictive problems in machine learning seek to build a program with the following I/O specs.
  - **Input**: A list/tuple/vector of *features* $x_1, \ldots, x_m$
  - **Output**: A discrete class label.
    - If there are only two possible classes/labels, we have a **binary** classification problem.
Examples

- Will the person vote conservative, given age, income, previous votes?
- Is the patient at risk of diabetes given body mass, age, blood test measurements?
- Predict Earthquake vs. nuclear explosion given body wave magnitude and surface wave magnitude.
Learning to Classify

• For classification, there are many datasets of the form 
  \((\text{input}_1, \text{output}_1), \ldots, (\text{input}_n, \text{output}_n)\)

• Mathematical notation: \((x_1, y_1), \ldots, (x_n, y_n)\).
  • If \(y\) is discrete \(\rightarrow\) **classification** problem
  • If \(y\) is continuous \(\rightarrow\) **regression** problem

- Supervised Learning: the data tell us the right answer
  • The basis of most neural net methods
Example Data

x1 = surface wave magnitude
x2 = body wave magnitude

white = earthquake
black = nuclear explosion
Example: Classifying Digits

- Classify image as “7” vs. “not 7”.
- Represent input image as vector $\mathbf{x}$ with $28 \times 28 = 784$ numbers.
- Discussion: is this representation a good idea?
Linear Classification Models
Linear Classification Models

- General Idea:
  1. Simple linear model $y(x) = \mathbf{w} \cdot \mathbf{x}$
     - $\mathbf{w}$ is a list of weight parameters to be learned
  2. Classify as positive if $y(x)$ crosses a threshold, typically 0.
     - The decision boundary $y(x) = 0$ defines a line for 2 input features, a hyperplane for $> 2$. 

![Diagram of linear classification models](image-url)
Example: Linear Separation

Russell and Norvig Figure 18.15

- $x_1$ = surface wave magnitude
- $x_2$ = body wave magnitude
- white = earthquake
- black = nuclear explosion

Graph showing two dimensions $x_1$ and $x_2$ with points representing different types of events.
Example: Classifying Digits

- Classify input vector as “7” vs. “not 7”.
- Represent input image as vector $\mathbf{x}$ with $28 \times 28 = 784$ numbers.
- Target $t = 1$ for “positive”, -1 for “negative”.
- Prediction function $y: \mathbb{R}^{784} \rightarrow \mathbb{R}$.
- Classify $\mathbf{x}$ as positive if $y(\mathbf{x}) > 0$, negative o.w.
- Discussion: how can we handle multiple classes, e.g. digits from 1..10?
The Bias Weight

- Usually a linear includes a bias term (aka intercept, baseline) $b$.
  
  - E.g. $y(x_1, x_2) = b + w_1 x_1 + w_2 x_2$

- Equivalently: add an imaginary constant 1 input $x_0 = 1$ and set $b = w_0$
  
  - E.g. $y(x_0 = 1, x_1, x_2) = \mathbf{w} \cdot \mathbf{x} = w_0 x_0 + w_1 x_1 + w_2 x_2 = w_0 \cdot 1 + w_1 x_1 + w_2 x_2$
Strengths of Linear Classifiers

- Efficient to learn
- Interpretable
  - in many applications, the “effects” are most important – which features receive the biggest weight.
- Can quantify predictive uncertainty
  - derive confidence bounds on accuracy of predictions.
- In machine learning, interpretability and quantifying uncertainty often go together, but trade off against accuracy.
- Science manages to combine all three.
Convexity and Linear Separability

- A set of points $C$ is convex if for any two points $x, y$ in $C$, fraction $0 \leq \alpha \leq 1$ we have $\alpha x + (1-\alpha)y$ is also in $C$.
- If two classes are linearly separable, they are convex.
- **Separating Hyperplane Theorem**: There exists a linear separator for two disjoint sets (classes) if and only if each set (class) is convex.
- **Illustration**
Linear Separability and Convexity

Data with outliers removed:
convex, separable

Original data:
non-convex, non-separable
Nonseparability

- Linear discriminants can solve problems only if the classes can be separated by a line (hyperplane).
- Canonical example of non-separable problem is X-OR.

\[ (a) \quad x_1 \text{ and } x_2 \]
\[ (b) \quad x_1 \text{ or } x_2 \]
\[ (c) \quad x_1 \text{ xor } x_2 \]
Responses to Nonseparability

Classes cannot be separated by a linear discriminant

- separate classes not completely but “well”
- add unobserved features

- Fisher discriminant (not covered)
- logistic regression

- neural network
- support vector machine
Gradient Descent Learning
Learning a Linear Classifier

- Most learning for continuous data follows a decision-theoretic (Bayesian) approach to learning.

1. For a given data set D, define an error function $E(w, D)$ that measures how well the weights $w$ fit the data D.
2. Find $w$ that minimize the error for a given input data set D.
3. In neural net learning, the basic minimization algorithm is gradient descent.
Gradient Descent: Choosing a direction

- Intuition: think about trying to find street number 1000 on a block. You stop and see that you are at number 100. Which direction should you go, left or right?
- You initially check every 50 houses or so where you are. What happens when you get closer to the goal 1000?
- The fly and the window: the fly sees that the wall is darker, so the light gradient goes down: bad direction.
- See here for illustration (around 38 sec)
Gradient Descent Scheme

1. Initialize weight vectors somehow.
   - Typically randomly, more on this later.

2. Update
   \[ w_{i+1} := w_i - \eta \frac{\partial E}{\partial w_i} \]

3. Until some convergence criterion is true
   - Complexity comment: assuming gradients are easy to compute, every update step is linear in the number of weights.
   - Learning time depends on number of steps required until convergence
Gradient Descent in One Dimension

- Update Rule
  \[ x := x - f'(x) \] where \( \eta \) is a step size

Example:
- Try to find \( y \) that minimizes \( f(y) = y^2 \).
- Your current location is \( y = -3 \).
- What is \( f'(y) \)?
- Answer: the derivative function is \( 2y \).
- Evaluated at the location \(-3\), the derivative is \( \nabla = -6 \).
- To minimize, we move in the opposite direction \(-\nabla\).
- Letting the step size \( \eta = 1 \), your new location is \(-3 - (-6) = -3 + 6 = 3\).
Gradient Descent In Multiple Dimensions

Gradient

$$\nabla E[\vec{w}] \equiv \left[ \frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n} \right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$
Gradient Descent: Example.

• Try to find x, y that minimize
  \[ f(x, y) = 3x + y^2. \]

• Your current location is \( x = 10, y = -3 \).

• What is \( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \)?

• Answer: the gradient vector is \( (3, 2y) \).

• Evaluated at the location \( (10, -3) \), the gradient is \( \nabla f = (3, -6) \).

• To minimize, we move in the opposite direction \(-\nabla f\).

• Letting the step size \( \eta = 1 \), your new location is
  \[ (10, -3) - (3, -6) = (7, 3). \]
Gradient Descent: Exercise

- Try to find $x,y$ that minimize
  $$f(x,y) = 3x + y^2.$$  
- Your current location is $x = 7, y = 3$.
- The gradient vector is $(3, 2y)$.
- Letting the step size $\eta = 1$, what is your new location?
- Demos
  - [Visualization](#)
  - [Excel Demo](#)
  - UBC neural net tool
Perceptron Learning
Defining an Error Function

- General idea:
  1. Encode class label using a real number $t$.
     - e.g., “positive” = 1, “negative” = 0 or “negative” = -1.
     - This is the first example we see of embedding.
  2. Measure error or **loss** by comparing continuous linear output $y$ and class label code $t$.
  3. Obvious loss function is 0-1:
     0 if prediction correct, 1 otherwise.
  4. In practice use **convex** upper bounds on 0-1 loss:
     $loss(y,t) \geq 0$ if prediction correct
     $loss(y,t) \geq 1$ if prediction false
Perceptrons

- Perceptrons are a precursor to neural nets.
- Analog implementation by Rosenblatt in the 1950s.
The Perceptron Error Function

- Let $y = 1$ for positive class, $y = -1$ for negative
- An example is misclassified if and only if $(x_j \cdot w)y_j < 0$
  - Exercise: Take a moment to verify this.
- Perceptron Error (fixed dataset $D$)
  $$E(w) = \sum_{j \in M} (x_j \cdot w)y_j$$
  where $M$ is the set of misclassified inputs, the mistakes.
- Exercise: find the gradient of the error function for a single input $x_j$.
- Solution:
  - 0 if $x_j$ is correctly classified.
  - $-x_jy_j$ o.w.
- What is the gradient descent weight update formula?
weight vector = black
Perceptron Learning Analysis

• **Theorem** If the classes are linearly separable, the perceptron learning algorithm converges to a weight vector that separates them.

• Number of steps to convergence has a theoretical upper bound.
  • In deep learning, usually no bound

• Sensitive to initialization.

• If classes are not linearly separable (e.g. X-OR), typically fails to converge.
Least-Squares Error Function

• Venerable idea (19\textsuperscript{th} century)
  1. For each data point, find the difference between predicted and observed value
  2. Square the difference
  3. Sum/average the squared differences
• Does not work well for classification problems
• But widely used for regression

\[ E(w) = \frac{1}{2} \sum_j ((x_j \cdot w) - y_j)^2 \]
Logistic Regression
From Values to Probabilities

- Key idea: instead of predicting a class label, predict the probability of a class label.
- E.g., $p^+ = P(\text{class is positive} \mid \text{features})$
  $p^- = P(\text{class is negative} \mid \text{features})$
- Naturally a continuous quantity.
- How to turn a real number $y$ into a probability $p^+$?
The Logistic Sigmoid Function

- **Definition:** \( \sigma(y) = \frac{1}{1 + \exp(-y)} \)
- Squeezes the real line into \([0, 1]\).
- Differentiable: \( \frac{d\sigma}{dy} = \sigma(1 - \sigma) \) (nice exercise)
Soft threshold interpretation

- If $y > 0$, $\sigma(y)$ goes to 1 very quickly.
- If $y < 0$, $\sigma(y)$ goes to 0 very quickly.

Figure Russell and Norvig 18.17
Probabilistic Interpretation

- The sigmoid can be interpreted in terms of the **class odds** \( p^+/(1-p^+) \).
- Exercise: Show the following implication for the class odds:

\[
p^+ = \frac{1}{1 + \exp(-y)} \implies \frac{p^+}{1 - p^+} = \exp(y)
\]

- Therefore \( y = \ln\left( \frac{p^+}{1 - p^+} \right) = \) the **log class odds**.
Logistic Regression

- In logistic regression, the log-class odds are a linear function of the input features:

\[
\ln\left(\frac{p^+}{1 - p^+}\right) = \mathbf{x} \cdot \mathbf{w}
\]

- Learning logistic regression is conceptually similar to linear regression.

- Log-linear (or exponential) models are the “nicest” general family of statistical models.
Logistic Regression: Maximum Likelihood

- Notation: the probability that the n-th input example is positive = which depends on a weight vector $\mathbf{w}$.
- Positive example has $y_j = 1$, negative $y_j = 0$.
- Then the likelihood assigned to $N$ independent training data is

$$p(\mathbf{y}; \mathbf{w}) = \prod_{j=1}^{N} (p_j^+)^{y_j} (1 - p_j^+)^{y_j}$$

- The cross-entropy error

$$E(\mathbf{w}) = -\ln p(\mathbf{y}; \mathbf{w}) = -\sum_{j=1}^{N} y_j \ln(p_j^+) + (1 - y_j) \ln(1 - p_j^+)$$

- Equivalent to minimizing the KL divergence between the predicted class probabilities and the observed class frequencies.
Weight Learning

- Homework Exercise: find the gradient of the cross-entropy error function wrt a single input $x_n$

$$E(w) = - \sum_{j=1}^{N} y_j \ln(p_j^+) + (1 - y_j) \ln(1 - p_j^+)$$

- Hint: recall that $\frac{d\sigma}{dy} = \sigma(1-\sigma)$

- No closed form minimum since $p_j^+$ is non-linear function of input features.

- Can use gradient descent. See Stanford Demo with softmax.

- Better approach: Use Iterative Reweighted Least Squares (IRLS).
Example logistic regression model learned on non-separable data
Multi-Class Example

- Logistic regression can be extended to multiple classes.
- Here’s a picture of what decision boundaries can look like.
Multi-Class Problems and the SoftMax Function

- Generic multi-class probabilistic prediction
  1. Build prediction models $y_1(x), y_2(x), \ldots y_k(x)$, one for each of the $k$ classes.
  2. Transform the numbers into probabilities:
     1. Map each $y_i(x)$ to $\exp\{y_i(x)\}$
     2. Divide by the sum:
        $$\sigma(x)_j = \frac{\exp\{y_j(x)\}}{\sum_i \exp\{y_i(x)\}}$$
Linear Algebra Notation

• Suppose we build 10 linear models $y_1(x), y_2(x), \ldots y_{10}(x)$, for 10 classes.

• For a data matrix $X$, the linear model predictions for each data point can be written like this (including the bias terms) $Y = XW + B$

• What are the dimensions of $X, W, Y, B$ and what do they represent?
Feature Transformations

Basis Functions
Example

- \( \varphi_1(x_1, x_2) \) measures distance from left green cross
- \( \varphi_2(x_1, x_2) \) measures closeness to centre green cross
- [3D demo](#)

Figure Bishop 4.12 see also
Visualize Transformation

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Transformations can keep/increase/decrease original dimensionality
Summary I

- Linear classification: learn a linear function of features that separates positive and negative classes.
- Find feature weights by minimizing an error function.
- Different error functions used:
  - perceptron
  - cross-entropy (logistic regression)
  - max-margin (support vector machines, not discussed)
  - Least squares (for regression not classification)
- If classes are not linearly separable, no linear classifier is 100% accurate.
- Options:
  - use the best line you can → logistic regression
  - use non-linear prediction function → neural nets
Looking Ahead to Deep Learning

- Neural nets:
  - Hidden layers transform the input features
  - Output nodes perform linear classification/regression on the transformed features

- Deep neural nets: transform input features, transform the transformed features, …., again and again

- The transformations are *learned* not fixed