Machine Learning

March 13, 2017

- n sample points $x_i \in \mathbb{R}^d$, $i = 1, \dots, n$
- $d = 2$ where not stated.

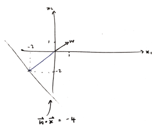
Classification

A **decision boundary** is a curve separating the plane (sample space) into two regions.

Some classifiers involve a **decision function** f, in which case $f(\mathbf{x}) = 0$ describes the decision boundary.

A **linear classifier** uses a linear decision function $f(x) = \mathbf{w} \cdot \mathbf{x} + \alpha$. This is scalar-valued: it's a plane over the plane (sample space). Its intersection defines a linear decision boundary.

In d-dimensions the decision boundary is a hyperplane ($(d-1)$-dimensional). This still separates the sample space into two regions.



Example: $f(x) = \begin{bmatrix}1\\1\end{bmatrix} \cdot \begin{bmatrix}x_1\\x_2\end{bmatrix} + 4$

- A plane sloping up at 45 in the north-east direction.
- Each input feature has equal influence on the classification.
- Decision boundary is line $x_1 + x_2 = -4$.
- $\mathbf{w}$ is normal to the decision boundary since $\mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) = -4 - (-4) = 0$.
- If one feature has a very high weight then $\mathbf{w}$ points close to that axis and the decision boundary is almost perpendicular to that axis (other features almost don't matter).

Distance from the decision boundary to a point: For some point $\mathbf{x}_i$, the height of the decision function plane above $\mathbf{x}_i$ is $\mathbf{w} \cdot \mathbf{x}_i + \alpha$. At the decision boundary, this height is zero. Looking "straight up" the slope of the decision function, its gradient is $\sqrt{w_1^2 + w_2^2} = |\mathbf{w}|$. So the distance of a point $\mathbf{x}_i$ from the hyperplane is $\frac{\mathbf{w} \cdot \mathbf{x}_i + \alpha}{|\mathbf{w}|}$. If $\mathbf{w}$ is not a unit vector, the problem can be rescaled

so that it is, in which case the distance is $\mathbf{w} \cdot \mathbf{x}_i + \alpha$.

Examples of linear classifiers:

- **Centroid method**: Decision boundary perpendicular to and bisects line connecting means of labeled training points.
- **Perceptron**:
- **Maximum margin classifier**:
- **LDA**: Fit Gaussians to each class, same covariance across classes.

Perceptron

Labels $y_i \in \{-1, 1\}$. Assume $\alpha = 0$ for now (decision boundary through origin).

Goal: find line separating points (separating hyperplane). I.e. Find $\mathbf{w}$ such that

$$\begin{cases} \mathbf{x}_i \cdot \mathbf{w} \leq 0, & y_i = -1 \\ \mathbf{x}_i \cdot \mathbf{w} \geq 0, & y_i = +1. \end{cases}$$

This is equivalent to the **constraint** $y_i \mathbf{x}_i \cdot \mathbf{w} \geq 0$.

Cost function: total distance $R(\mathbf{w})$ of misclassified points from the decision boundary.

Optimization problem: Find $\mathbf{w}$ that minimizes

$$R(w) = \sum_i L(\mathbf{x}_i \cdot \mathbf{w}, y_i) = \sum_{i \in V} -y_i \mathbf{x}_i \cdot \mathbf{w},$$

where V are the misclassified points.

Per-training point loss function

$$L(\text{prediction}_i, y_i) = L(\mathbf{x}_i \cdot \mathbf{w}, y_i) = \begin{cases} 0, & \text{correct}, y_i \mathbf{x}_i \cdot \mathbf{w} \geq 0 \\ -y_i \mathbf{x}_i \cdot \mathbf{w}, & \text{misclassified} \end{cases}$$

Gradient descent: Find w that minimizes $R(w)$.

$$\nabla_w R = \begin{bmatrix} -\sum_i y_i X_{i1} \\ \vdots \\ -\sum_i y_i X_{id} \end{bmatrix}$$

- On each iteration, compute the gradient; update $\mathbf{w}$ by taking a step downhill of size ρ: $\mathbf{w} \leftarrow \mathbf{w} + \rho \sum_{i \in V} y_i \mathbf{x}_i$.
- A misclassified data point far out in dimension j will cause the gradient to have a large component $-\sum_i y_i X_{ij}$ in that dimension.
- $\mathbf{w}$ thus becomes more closely aligned with that axis and the decision boundary.
- Decision boundary therefore becomes more perpendicular to that axis (axis becomes more "important").

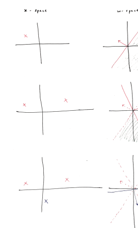
Stochastic gradient descent (Perceptron): on each iteration pick one misclassified point and update $\mathbf{w}$ using gradient for that point: $\mathbf{w} \leftarrow \mathbf{w} + \rho y_{i^*} \mathbf{x}_{i^*}$

Allow decision boundaries that do not pass through origin: add a fictitious dimension so that sample points now lie on the plane $x_{d+1} = 1$ in $(d+1)$ dimensions. Run algorithm as above, just with the new dimensionality.

$$\mathbf{w} \cdot \mathbf{x} + \alpha = 0$$
$$\begin{bmatrix} w_1 \\ w_2 \\ \alpha \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = 0.$$

Optimization in weight space

$\mathbf{x}$-space	$\mathbf{w}$-space
hyperplane	point $\mathbf{w}$ is normal vector to hyperplane
point	hyperplane whose normal vector is the $\mathbf{x}$ point (? don't understand this yet)



Maximum margin classifiers

Margin is distance from hyperplane to nearest sample point.

Previously, in the perceptron, we used the constraint

$$y_i \mathbf{x}_i \cdot \mathbf{w} \geq 0.$$

Now, we demand that there is a non-zero margin between the decision boundary and the points:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + \alpha) \geq 1,$$

The 1 on the RHS is arbitrary; I think $\mathbf{w}$ and α will adapt to make it true for any positive value, so the point is that we're demanding a strictly non-zero margin.

Optimization problem (quadratic program):
Find $\mathbf{w}, \alpha$ that minimize $|\mathbf{w}|^2$ such that $y_i(\mathbf{x}_i \cdot \mathbf{w} + \alpha) \geq 1$ for all points i.

Soft margin SVMs [1] [2]

- Still quadratic program but allow points to violate margin via **slack variables** $\xi_i \geq 0$:
- Constraint is $y_i(\mathbf{x}_i \cdot \mathbf{w} + \alpha) \geq 1 - \xi_i$
- Find non-linear decision boundaries by introducing new features comprising non-linear functions of base features ("lift points into higher-dimensional space").

Optimization problem:

Find w, α, and ξ_i that minimize $|w|^2 + C \sum_{i=1}^n \xi_i$
subject to $y_i(X_i \cdot w + \alpha) \geq 1 - \xi_i$ for all $i \in [1, n]$
$\xi_i \geq 0$ for all $i \in [1, n]$

. . . a quadratic program in $d + n + 1$ dimensions and $2n$ constraints.
[It's a quadratic program because its objective function is quadratic and its constraints are linear inequalities.]

$C > 0$ is a scalar regularization hyperparameter that trades off:

	small C	big C
desire	maximize margin $1/\lvert w \rvert$	keep most slack variables zero or small
danger	underfitting (misclassifies much training data)	overfitting (awesome training, awful test)
outliers	less sensitive	very sensitive
boundary	more "flat"	more sinuous

[1] https://people.eecs.berkeley.edu/~jrs/189/lec/04.pdf
[2] https://www.youtube.com/watch?v=HOZ6ZpPA_Ks

Decision Theory [3] [4]

Suppose there are two possible **classes**: $\{C, D\}$

Decision rule: $r(\mathbf{x}) : \mathbb{R}^d \to \{C, D\}$

Loss function: E.g. 0-1 loss:

$$L(y_i \to \hat{y}_i) = \begin{cases} 0, & \hat{y}_i = y_i \\ 1, & \text{otherwise} \end{cases} \qquad \text{(correct classification)}$$

Risk: Functional $R(r)$: expected loss for rule r, over $\mathrm{p}(X, Y)$. [5]

So what rule function r minimizes the functional R?

Bayes decision rule: Assign $\mathbf{x}$ to class C if

$$\text{(C posterior at } \mathbf{x}) \times \text{(penalty for misclassifing a true C)}$$

is largest for class C. I.e. if

$$\mathrm{p}(C|\mathbf{x}) L(D|C) > \mathrm{p}(D|\mathbf{x}) L(C|D).$$

With 0-1 loss, this is: "assign to class with highest posterior".

With 0-1 loss and two classes, it's: "assign to class with posterior > 0.5".

Empirical risk: Discriminative methods (e.g. logistic regression) lack any model for X. How can we estimate expected loss over $p(X, Y)$? Take the observed sample points as defining a discrete, uniform distribution, in which case

$$\hat{R}(r) = \frac{1}{n} \sum L(r(x_i), y_i).$$

This provides a justification for minimizing the sum/mean of per-sample loss.

[3] https://people.eecs.berkeley.edu/~jrs/189/lec/06.pdf
[4] https://www.youtube.com/watch?v=aXkenQ01qYI
[5]

$$\begin{aligned} R(r) =& \pi(Y = -1)\, \mathbb{E}_{\mathbf{X}}\, L(-1 \to r(X)) + \\ & \pi(Y = +1)\, \mathbb{E}_{\mathbf{X}}\, L(+1 \to r(X)) \qquad \text{over } \mathrm{p}(Y)\, \mathrm{p}(X|Y) \\ =& \sum_X \mathrm{p}(X) \big(\pi(Y = -1) L(-1 \to r(X)) + \\ & \pi(Y = +1) L(+1 \to r(X)) \big) \qquad \text{over } \mathrm{p}(X)\, \mathrm{p}(Y|X) \end{aligned}$$

Statistical justifications

Regression: want to estimate a function f such that $y_i = f(x_i) + \epsilon$, where ϵ has unknown distribution but mean 0. Ideal would be to estimate f with $h(x_i) = \mathbb{E}(Y|x_i)$ since this is equal to $f(x_i)$.

Likelihood justification for linear regression cost function.

Logistic Regression from Maximum Likelihood

Bias-Variance Decomposition

$$\begin{aligned} &= \mathrm{E}[(h(z) - \gamma)^2] \\ &= \mathrm{E}[h(z)^2] + \mathrm{E}[\gamma^2] - 2\,\mathrm{E}[\gamma\, h(z)] \qquad \text{[Observe that } \gamma \text{ and } h(z) \text{ are independent]} \\ &= \mathrm{Var}(h(z)) + \mathrm{E}[h(z)]^2 + \mathrm{Var}(\gamma) + \mathrm{E}[\gamma]^2 - 2\mathrm{E}[\gamma]\,\mathrm{E}[h(z)] \\ &= (\mathrm{E}[h(z)] - \mathrm{E}[\gamma])^2 + \mathrm{Var}(h(z)) + \mathrm{Var}(\gamma) \\ &= \underbrace{(\mathrm{E}[h(z)] - f(z))^2}_{\text{bias}^2 \text{ of method}} + \underbrace{\mathrm{Var}(h(z))}_{\text{variance of method}} + \underbrace{\mathrm{Var}(\epsilon)}_{\text{irreducible error}} \end{aligned}$$

Gaussian discriminant analysis [6] [7]

Anisotropic:

$$\mathrm{p}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \mu)^{\mathbf{T}} \Sigma^{-1} (\mathbf{x} - \mu) \right)$$

Isotropic:

$$\mathrm{p}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} \sigma^d} \exp\left(-\frac{|\mathbf{x} - \mu|^2}{2\sigma} \right)$$

Isotropic Gaussians

Multivariate data $\mathbf{x}$ but features uncorrelated and all features same variance.L

QDA

Fit separate Gaussians to the training data in each class. The likelihood is

$$\mathrm{p}(\mathbf{x}|\text{class } C) = \frac{1}{(2\pi)^{d/2} \sigma_C^d} \exp\left(-\frac{|\mathbf{x} - \mu_C|^2}{\sigma_C^2} \right)$$

and we compare the value of $\mathrm{p}(\mathbf{x}|\text{class } C) \cdot \pi_C \cdot L(D|C)$.

The decision boundaries are where the posterior $\times$ loss are equal. It's easier to compare the log of this:

$$Q_C(\mathbf{x}) = -\frac{|\mathbf{x} - \mu_C|^2}{\sigma_C^2} - d \log \sigma_C + \log \pi_C + \log L(D|C)$$

The posterior probability of class C at point $\mathbf{x}$ is[8]

$$\mathrm{p}(C|\mathbf{x}) = \frac{\pi_C\, \mathrm{p}(\mathbf{x}|C)}{\pi_C\, \mathrm{p}(\mathbf{x}|C) + \pi_D\, \mathrm{p}(\mathbf{x}|D)} = \frac{1}{1 + e^{-(Q_C(\mathbf{x}) - Q_D(\mathbf{x}))}},$$

so logistic in the quadratic expression $Q_C(\mathbf{x}) - Q_D(\mathbf{x})$.

LDA

Estimate separate class means but same variance for all classes. So now

$$\begin{aligned} Q_C(\mathbf{x}) - Q_D(\mathbf{x}) &= \frac{|\mathbf{x} - \mu_D|^2 - |\mathbf{x} - \mu_C|^2}{\sigma^2} + \log \frac{\pi_C}{\pi_D} + \log \frac{L(D|C)}{L(C|D)} \\ &= \frac{(\mathbf{x} - \mu_D) \cdot (\mathbf{x} - \mu_D) - (\mathbf{x} - \mu_C) \cdot (\mathbf{x} - \mu_C)}{\sigma^2} + \log \frac{\pi_C}{\pi_D} + \log \frac{L(D|C)}{L(C|D)} \\ &= \mathbf{x} \cdot \frac{2(\mu_C - \mu_D)}{\sigma^2} + \left(\frac{|\mu_D|^2 - |\mu_C|^2}{\sigma^2} + \log \frac{\pi_C}{\pi_D} + \log \frac{L(D|C)}{L(C|D)} \right) \\ &= \mathbf{x} \cdot \mathbf{w} + \alpha \end{aligned}$$

[6] https://people.eecs.berkeley.edu/~jrs/189/lec/07.pdf
[7] https://www.youtube.com/watch?v=4CefboCXxZs
[8] This is assuming 0-1 loss, so the loss doesn't affect $Q_C(\mathbf{x})$

This means that the decision boundary is linear, and (with 0-1 loss) the posterior is a logistic function which is constant parallel to the decision boundary.

Symmetric matrices, quadratic forms and eigenvectors [9]

Spectral theorem: A symmetric matrix has n **orthogonal** eigenvectors[10][11]

To understand a symmetric matrix $\mathbf{A}$, consider its **quadratic form** $|\mathbf{A}\mathbf{x}|^2 = \mathbf{x}^{\mathbf{T}} \mathbf{A}^2 \mathbf{x}$ (right). Compare this to the graph of $|\mathbf{z}|^2$ (left). The graphs are related by the following changes of coordinates:

$\mathbf{z} \leftarrow \mathbf{A}\mathbf{x}$ changes the elliptical contours into circles; scale by eigenvalues of $\mathbf{A}$.

$\mathbf{A}^{-1}\mathbf{z} \to \mathbf{x}$ changes circles into ellipses; scale by reciprocal of eigenvalues.

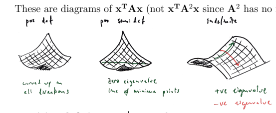
$|\mathbf{A}\mathbf{x}|^2 = 1$ is the equation of an ellipsoid. Its axes are $v_1, \dots, v_n$ and its radii are $\frac{1}{\lambda_1}, \dots, \frac{1}{\lambda_n}$,

Bigger eigenvalue $\iff$ steeper hill.

Alternate interpretation: the ellipsoids are spheres in a space with a different distance metric. The distance metric (metric tensor) is $\mathbf{M} = \mathbf{A}^2$:

$$d(\mathbf{x}, \mathbf{x}') = |\mathbf{A}\mathbf{x}| - |\mathbf{A}\mathbf{x}'| = \sqrt{(\mathbf{x} - \mathbf{x}')\mathbf{A}^2(\mathbf{x} - \mathbf{x}')}$$

These are diagrams of $\mathbf{x}^{\mathbf{T}}\mathbf{A}\mathbf{x}$ (not $\mathbf{x}^{\mathbf{T}}\mathbf{A}^2\mathbf{x}$ since $\mathbf{A}^2$ has no negative eigenvalues):



positive definite	eigenvalues > 0	$\mathbf{x}^{\mathbf{T}}\mathbf{A}\mathbf{x} > 0 \quad \forall \mathbf{x} \neq \mathbf{0}$
positive semidefinite	eigenvalues ≥ 0	$\mathbf{x}^{\mathbf{T}}\mathbf{A}\mathbf{x} >= 0 \quad \forall \mathbf{x}$
indefinite	some positive and some negative eigenvalues	
singular	some zero eigenvalue	

Let Λ be a diagonal matrix containing the eigenvalues and $\mathbf{V}$ contain normalized eigenvectors:

$$V = \begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \\ | & | & & | \end{bmatrix}$$

Note that for an **orthonormal** matrix like this:

1. It rotates / reflects the input vectors, without changing their length.
2. $\mathbf{V}^T \mathbf{V} = \mathbf{I}$, therefore $\mathbf{V}^{-1} = \mathbf{V}^{\mathbf{T}}$.

[9] https://people.eecs.berkeley.edu/~jrs/189/lec/08.pdf
[10] There may be more than n (infinite) eigenvectors, but n orthogonal.
[11] Non-symmetric matrices have non-orthogonal eigenvectors in general.

By the definition of eigenvector we have

$$\mathbf{AV} = \mathbf{V}\Lambda$$

and therefore the **eigendecomposition** of $\mathbf{A}$

$$\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^{\mathbf{T}}.$$

So we can perform $\mathbf{Ax}$ as $\mathbf{V}\Lambda\mathbf{V}^{\mathbf{T}}\mathbf{x}$, and $\mathbf{A}^k\mathbf{x}$ as $\mathbf{V}\Lambda^k\mathbf{V}^{\mathbf{T}}\mathbf{x}$:

1. $\mathbf{V}^{\mathbf{T}} = \mathbf{V}^{-1}$ rotates the input vector into axis-aligned coordinates.
2. Λ scales along different axes.
3. $\mathbf{V}$ returns to the original coordinates.

Λ is said to be the diagonalized version of $\mathbf{A}$.

9 The Anisotropic Multivariate Normal Distribution, QDA, and LDA

Regression

Linear Least Squares Regression

Use fictitious dimension trick, so that $\mathbf{w}$ includes the offset term α and $\mathbf{X}$ is $(n \times (d+1))$.

Find $\mathbf{w}$ that minimizes cost function $J(w)$: sum of squared difference between linear predictor and observed training point.

$$J(w) = |\mathbf{Xw} - \mathbf{y}|^2 = \sum_i (\mathbf{x}_i^{\mathbf{T}}\mathbf{w} - y_i)^2$$

Solve by differentiating and finding the critical point:

$$\begin{aligned}
|\mathbf{Xw} - \mathbf{y}|^2 &= \mathbf{w}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\mathbf{Xw} - 2\mathbf{y}^{\mathbf{T}}\mathbf{Xw} + \mathbf{y}^{\mathbf{T}}\mathbf{y} \\
\nabla_{\mathbf{w}} |\mathbf{Xw} - \mathbf{y}|^2 &= 2\mathbf{X}^{\mathbf{T}}\mathbf{Xw} - 2\mathbf{X}^{\mathbf{T}}\mathbf{y} \\
\mathbf{w}^* &= (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y} =: \mathbf{X}^+\mathbf{y}
\end{aligned}$$

For a new sample point $\mathbf{x}$, the prediction is $\hat{y} = \mathbf{x} \cdot \mathbf{w}^*$.

Related concepts

- **normal equations**: linear system of d equations in unknown $\mathbf{w}$ resulting from setting the gradient equal to zero: $\mathbf{X}^{\mathbf{T}}\mathbf{Xw} - \mathbf{X}^{\mathbf{T}}\mathbf{y} = \mathbf{0}$
- **pseudoinverse**: The matrix $\mathbf{X}^+ = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}$ maps $\mathbf{y}$ to $\mathbf{w}^*$. In general there's no $\mathbf{w}$ that solves $\mathbf{Xw} = \mathbf{y}$, but $\mathbf{w}^* = \mathbf{X}^+\mathbf{y}$ makes the LHS as close as possible to $\mathbf{y}$. So it behaves as a "left inverse" of $\mathbf{X}$, since $\mathbf{X}^+\mathbf{X} = \mathbf{I}$ and left-multiplying by $\mathbf{X}^+$ gives the "solution" to $\mathbf{Xw} = \mathbf{y}$.
- **projection matrix** or **hat matrix**: Still focusing on the training phase, the predictions are $\hat{\mathbf{y}} = \mathbf{Xw}^* = \mathbf{XX}^+\mathbf{y}$. So $\mathbf{XX}^+$ puts that hat on $\mathbf{y}$, or projects $\mathbf{y}$ onto the hyperplane, in the viewpoint described below.

Projection interpretation

Usually we think of n points in $\mathbb{R}^d$. But instead, consider a separate column of the data for each feature: these are d points in $\mathbb{R}^n$. The observed training data $\mathbf{y}$ is also a point in $\mathbb{R}^n$, and so is the prediction $\hat{\mathbf{y}} = \mathbf{Xw}$.

As we vary $\mathbf{w}$, the prediction $\mathbf{Xw}$ describes a hyperplane spanned by the columns of $\mathbf{X}$.

We want to find the $\mathbf{w}^*$ corresponding to the closest point on the hyperplane to $\mathbf{y}$. So $X\mathbf{w}^* - \mathbf{y}$ must be orthogonal to the hyperplane:

$$\mathbf{X}^{\mathbf{T}} \cdot (\mathbf{Xw}^* - \mathbf{y}) = \mathbf{0}.$$

Which are the normal equations (linear system of d equations), derived differently.

Weighted linear regression

Sample point i has weight b_i. Diagonal $n \times n$ matrix $\mathbf{B}$ contains weights.

$$\begin{aligned}
J(\mathbf{w}) &= \sum_i b_i(\mathbf{x}_i^{\mathbf{T}}\mathbf{w} - y_i)^2 \\
&= (\mathbf{Xw} - \mathbf{y})^{\mathbf{T}}\mathbf{B}(\mathbf{Xw} - \mathbf{y}) \\
&= \mathbf{w}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\mathbf{BXw} - 2\mathbf{y}^{\mathbf{T}}\mathbf{BXw} + \mathbf{y}^{\mathbf{T}}\mathbf{y}
\end{aligned}$$

Gradient

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = 2\mathbf{X}^{\mathbf{T}}\mathbf{BXw} - 2\mathbf{X}^{\mathbf{T}}\mathbf{By}$$

Solution

$$\mathbf{w}^* = \left(\mathbf{X}^{\mathbf{T}}\mathbf{BX}\right)^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{By}$$

How to compute the gradient

The cost function is $J(\mathbf{w}) = |\mathbf{Xw} - \mathbf{y}|^2$. We could write this as a dot product and multiply out:

$$\begin{aligned}
J(\mathbf{w}) &= (\mathbf{Xw} - \mathbf{y}) \cdot (\mathbf{Xw} - \mathbf{y}) \\
&= \mathbf{Xw} \cdot \mathbf{Xw} - 2\mathbf{Xw} \cdot \mathbf{y} + \mathbf{y} \cdot \mathbf{y} \\
&= (\mathbf{Xw})^{\mathbf{T}}\mathbf{Xw} - 2(\mathbf{Xw})^{\mathbf{T}}\mathbf{y} + \mathbf{y}^{\mathbf{T}}\mathbf{y} \\
&= \mathbf{w}^T\mathbf{X}^{\mathbf{T}}\mathbf{Xw} - 2\mathbf{w}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\mathbf{y} + \mathbf{y}^{\mathbf{T}}\mathbf{y},
\end{aligned}$$

and then we'd need to differentiate those terms w.r.t. $\mathbf{w}$. However, a better way is to use the chain rule. Define f and g such that $J : \mathbb{R}^d \to \mathbb{R}$ is their composition $J = g \circ f$:

$$\begin{aligned}
f &: \mathbb{R}^d \to \mathbb{R}^n & f(\mathbf{w}) &= \mathbf{Xw} - \mathbf{y} \\
g &: \mathbb{R}^n \to \mathbb{R} & g(\mathbf{z}) &= |\mathbf{z}|^2.
\end{aligned}$$

The chain rule says that $\nabla(g \circ f) = (Df)^{\mathbf{T}}\, \nabla g$, where Df is the derivative of f, i.e. the Jacobian matrix of first partial derivatives[12]. We have $Df(\mathbf{w}) = \mathbf{X}$ and $\nabla g(\mathbf{z}) = 2\mathbf{z}$, so

$$\begin{aligned}
\nabla J(\mathbf{w}) &= 2\mathbf{X}^{\mathbf{T}}(\mathbf{Xw} - \mathbf{y}) \\
&= 2\mathbf{X}^{\mathbf{T}}\mathbf{Xw} - 2\mathbf{X}^{\mathbf{T}}\mathbf{y}.
\end{aligned}$$

Penalized Regression

TODO

Logistic Regression

- Two classes.
- The observations y_i are class labels (or probabilities thereof).

[12] The gradient ∇ applies only to scalar-valued functions.

- The model states that the probability of being in class 1 is given by the usual linear model, mapped onto $(0,1)$ by the logistic function s:

$$\begin{aligned}
y_i &\sim \mathrm{Bern}(s(\mathbf{x}_i^{\mathbf{T}}\mathbf{w})), \\
s(z) &= \frac{1}{1 + e^{-z}}
\end{aligned}$$

Note that $s'(z) = \frac{e^{-z}}{(1+e^{-z})^2} = s(z)(1 - s(z))$.

Likelihood

Let $s_i = s(\mathbf{x}_i^{\mathbf{T}}\mathbf{w})$.

$$\begin{aligned}
\mathcal{L}(\mathbf{w}) &= \prod_i s_i^{y_i} + (1 - s_i)^{(1-y_i)} \\
l(\mathbf{w}) &= \sum_i y_i \log s_i + (1 - y_i)\log(1 - s_i) \\
\nabla l(\mathbf{w}) &= \sum_i \frac{y_i}{s_i}(s_i)(1 - s_i)\mathbf{x}_i + \frac{1 - y_i}{1 - s_i}(-1)(s_i)(1 - s_i)\mathbf{x}_i \\
&= \sum_i \mathbf{x}_i\left(y_i(1 - s_i) - (1 - y_i)s_i\right) \\
&= \sum_i \mathbf{x}_i\left(y_i - s_i\right) \\
&= \mathbf{X}^{\mathbf{T}}\left(\mathbf{y} - \mathbf{s}(\mathbf{Xw})\right) \qquad (d \times 1)
\end{aligned}$$

where $\mathbf{s} : \mathbb{R}^n \to \mathbb{R}^n$ applies s componentwise to the rows.

Optimization problem: Find $\mathbf{w}$ that minimizes the cost function $J(\mathbf{w}) = -l(\mathbf{w})$.

Because the weights $\mathbf{w}$ are tied up inside $s_i = s(\mathbf{x}_i^{\mathbf{T}}\mathbf{w})$ it's not possible to find the minimum $\mathbf{w}^*$ by setting the gradient equal to zero (i.e. by solving a linear system). We can use gradient descent, or Newton's method.

For Newton's method, we need the Hessian of the objective function. This is the $d \times \mathrm{d}$ matrix of partial derivatives of the gradient, i.e. $\mathbf{X}^{\mathbf{T}}$ multiplied by the derivative (Jacobian matrix) of $\mathbf{s}(\mathbf{Xw})$. Define $\mathbf{f}(\mathbf{w}) = \mathbf{Xw}$ so now $\mathbf{s}(\mathbf{Xw}) = (\mathbf{s} \circ \mathbf{f})(\mathbf{w})$.

Function	domain $\to$ range	Jacobian	dim Jacobian
$\mathbf{f}(\mathbf{w}) = \mathbf{Xw}$	$\mathbb{R}^d \to \mathbb{R}^n$	$\mathrm{D}\,\mathbf{f} = \mathbf{X}$	$n \times d$
$\mathbf{s}(\mathbf{z})$	$\mathbb{R}^n \to \mathbb{R}^n$	$\mathrm{D}\,\mathbf{s}(\mathbf{z}) = \mathbf{S}$	$n \times n$

where $\mathbf{S}$ is a diagonal matrix with $\mathbf{S}_{ii} = \mathbf{s}(\mathbf{x}_i^{\mathbf{T}}\mathbf{w})(1 - \mathbf{s}(\mathbf{x}_i^{\mathbf{T}}\mathbf{w}))$. Now by the chain rule,

$$\begin{aligned}
\nabla^2 J(\mathbf{w}) &= \mathbf{X}^{\mathbf{T}}\, \mathrm{D}_{\mathbf{w}}\, \mathbf{s}(\mathbf{Xw}) \\
&= \mathbf{X}^{\mathbf{T}}(\mathrm{D}_{\mathbf{f}}\, \mathbf{s})(\mathrm{D}_w\, \mathbf{f}) \\
&= \mathbf{X}^{\mathbf{T}}\mathbf{SX}.
\end{aligned}$$

1 Change of basis

Suppose person B uses some other basis vectors to describe locations in space. Specifically, in our coordinates, their basis vectors are $\left[\begin{smallmatrix}2\\1\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}-1\\1\end{smallmatrix}\right]$.

When they state a vector, what is it in our coordinates?

If they say $\left[\begin{smallmatrix}-1\\2\end{smallmatrix}\right]$, what is that in our coordinates?

Well, if they say $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$, that's $\left[\begin{smallmatrix}2\\1\end{smallmatrix}\right]$ in our coordinates. And if they say $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$, that's $\left[\begin{smallmatrix}-1\\1\end{smallmatrix}\right]$ in our coordinates. So the matrix containing *their basis vectors expressed using our coordinate system* transforms a point expressed in their coordinate system into one expressed in ours. That last sentence is critical, so hopefully it makes sense! So, the answer is

$$\mathbf{2}{-}111 \begin{bmatrix}-1\\2\end{bmatrix} = \begin{bmatrix}-4\\1\end{bmatrix}.$$

When we state a vector, what is it in their coordinates?

We give the vector $\left[\begin{smallmatrix}3\\2\end{smallmatrix}\right]$. What is that in their coordinate system? By definition, the answer is the weights that scales their basis vectors to hit $\left[\begin{smallmatrix}3\\2\end{smallmatrix}\right]$. So, the solution to

$$\mathbf{2}{-}111 \begin{bmatrix}a\\b\end{bmatrix} = \begin{bmatrix}3\\2\end{bmatrix}.$$

Computationally, we can see that we can get the solution by multiplying both sides by the inverse:

$$\begin{bmatrix}a\\b\end{bmatrix} = \mathbf{2}{-}111^{-1} \begin{bmatrix}3\\2\end{bmatrix}.$$

Conceptually, we have

$$\mathbf{2}{-}111 = \begin{bmatrix}\text{matrix converting their}\\ \text{representation to ours}\end{bmatrix}$$

where "their representation" means the vector expressed using their coordinate system. So the role played by the inverse is

$$\begin{bmatrix}a\\b\end{bmatrix} = \begin{bmatrix}\text{matrix converting our}\\ \text{representation to theirs}\end{bmatrix} \begin{bmatrix}3\\2\end{bmatrix}.$$

When we state a transformation, what is it in their coordinates?

We state a 90 anticlockwise rotation of 2D space:

$$\mathbf{0}{-}110$$

what is that transformation in their coordinates? The answer is

$$\begin{bmatrix}\text{matrix converting our}\\ \text{representation to theirs}\end{bmatrix} \mathbf{0}{-}110 \begin{bmatrix}\text{matrix converting their}\\ \text{representation to ours}\end{bmatrix}$$

since the composition of those three transformations defines a single transformation that takes in a vector expressed in their coordinate system, converts it to our coordinate system, transforms it as requested, and then converts back to theirs.

2 Symmetric matrices

Spectral theorem for symmetric matrices

Symmetric $n \times n$ matrix A (real).

$A^{-1} = A^{\mathbf{T}}$

n orthogonal eigenvectors with real eigenvalues.

Orthonormal matrix U containing normalized eigenvectors.

$A = U\Lambda U^{-1} = U\Lambda U^{\mathbf{T}}$

(Eigenvalues are uniquely determined by matrix. Eigenvalues can be repeated, in which case any linear combination of their eigenvalues is also an eigenvalue.)

Linear and quadratic approximations to a function [13]

We construct first- and second-order approximations to a differentiable function $f : \mathbb{R}^2 \to \mathbb{R}$. The approximation is made at some point $(x_0, y_0) = \mathbf{x}_0 \in \mathbb{R}^2$; we demand that the value of the approximation, and the first and second derivatives, match those of f exactly at that point.

Linear approximation to a function $f(x, y)$ near (x_0, y_0):

$$\begin{aligned}
L(x,y) = {} & f(x_0, y_0) \ + (x - x_0)f_x(x_0, y_0) + (y - y_0)f_y(x_0, y_0) \\
&= f(\mathbf{x}) + (\mathbf{x} - \mathbf{x}_0) \cdot \nabla_f(\mathbf{x}_0)
\end{aligned}$$

Note that, at (x_0, y_0), the first partial derivatives of L are equal to those of f, as they must be. (In fact, we could say that the coefficients are determined by this requirement; see the quadratic case below. But the linear case is obvious without "deriving" the coefficients.)

Quadratic approximation to a function $f(x, y)$ near (x_0, y_0):

First note that the "quadratic form" $ax^2 + 2bxy + cy^2$ can be written as

$$\mathbf{x}^{\mathbf{T}}A\mathbf{x} = \begin{bmatrix}x\\y\end{bmatrix}^{\mathbf{T}} \mathbf{a}bbc \begin{bmatrix}x\\y\end{bmatrix}.$$

[13] khanacademy - Grant Sanderson - second partial derivative test

This is a scalar. In general, a quadratic form for symmetric matric A is

$$\mathbf{x}^{\mathbf{T}}A\mathbf{y} = \sum_{jk} A_{jk}x_jy_k.$$

The j-th component of the gradient of $q(\mathbf{x}) = \mathbf{x}^{\mathbf{T}}A\mathbf{x}$ is $\frac{\partial q}{\partial x_j} = 2\sum_k A_{jk}x_k$, so

$$\nabla\, \mathbf{x}^{\mathbf{T}}\mathbf{Ax} = 2\mathbf{Ax}.$$

$$\begin{aligned}
Q(x,y) = {} & f(\mathbf{x}_0) + (x - x_0)f_x(\mathbf{x}_0) + (y - y_0)f_y(\mathbf{x}_0) + \\
& \frac{1}{2}f_{xx}(\mathbf{x}_0)(x - x_0)^2 + f_{xy}(\mathbf{x}_0)(x - x_0)(y - y_0) + \frac{1}{2}f_{yy}(\mathbf{x}_0)(y - y_0)^2 \\
= {} & f(\mathbf{x}_0) + (\mathbf{x} - \mathbf{x}_0) \cdot \nabla f(\mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^{\mathbf{T}}\nabla^2 f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0),
\end{aligned}$$

where $\nabla^2 f(\mathbf{x}_0)$ is the Hessian matrix $\mathbf{f_{xx}}f_{xy}f_{yx}f_{yy}$ evaluated at $\mathbf{x}_0$.

Second partial derivative test and positive definiteness of Hessian

The second partial derivative test for a function of two variables states that we examine the determinant of the Hessian evaluated at the critical point:

$$D = \det \nabla^2 f(\mathbf{x}_0) = f_{xx}(\mathbf{x}_0)f_{yy}(\mathbf{x}_0) - f_{xy}(\mathbf{x}_0)^2.$$

Notice that $D \geq 0$ implies that the sign of f_{xx} and f_{yy} agree (because we're subtracting the square of the mixed partial f_{xy}, i.e. a positive number).

D	roots	f_{xx}		Hessian
+	no real roots	+	minimum	positive definite
+	no real roots	−	maximum	negative definite
0	one real root	+	minimum	positive semidefinite
0	one real root	−	maximum	negative semidefinite
−	two real roots	n/a	saddle point	-

Explanation

At a critical point $\mathbf{x}_0$, the gradient is zero and the quadratic approximation is therefore

$$Q(x,y) = f(\mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^{\mathbf{T}}\nabla^2 f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0).$$

So if this is a minimum (concave-up paraboloid) then this quadratic form is positive for all $\mathbf{x} \neq \mathbf{x}_0$ (and if it's a maximum then it's negative for all $\mathbf{x} \neq \mathbf{x}_0$).

Basically the argument is that, instead of analyzing the function f itself, we analyze its quadratic approximation at the critical point. So the question comes down to: how do we determine whether a quadratic form is always positive, always negative, or takes positive and negative values?

To answer that, consider a generic quadratic form $ax^2 + 2bxy + cy^2$. Let y be constant at y_0; then we have a quadratic in x, the roots of which are

$$x = \frac{-2by_0 \pm \sqrt{4b^2y_0^2 - 4acy_0^2}}{2a} = y_0\frac{-b \pm \sqrt{b^2 - ac}}{a}.$$

So, whether this is a saddle point or a minimum/maximum depends on whether the quadratic form has real roots. If there are no real roots, then whether it's a minimum or a maximum depends on the sign of f_{xx} (this sign will be the same as that of f_{yy} in the no real roots case).